Qeios

# A Methodological Contribution to Efficient Dynamic Assessment of Reliability Using Satisfiability Approach

David Jaures Fotsa-Mbogne, Guy-de-patience Ftatsi-Mbetmi, Martial Ndje and Markert Benjaulys Tadie-Silatchom

Preprint v1

Oct 4, 2023

https://doi.org/10.32388/5KQP4O

# A methodological contribution to efficient dynamic assessment of reliability using satisfiability approach

David Jaurès Fotsa-Mbogne [a,e,f] *, Guy-de-patience Ftatsi-Mbetmi[b,e,g] , Martial Ndje [c,f] , Markert Benjaulys Tadie-Silatchom [d,e]

[a]Department of Mathematics and Computer Science, ENSAI, The University of Ngaoundere,

[b]Department of Mechanical Engineering, UIT, The University of Ngaoundere,

[c]Department of Electrical Engineering, UIT, The University of Dschang,

[d]Department of Mechanical Engineering, ENSAI, The University of Ngaoundere,

[e]Laboratory of Experimental Mathematics, ENSAI, The University of Ngaoundere,

[f]Laboratory of Electrical, Signal, Image et Automatics, ENSAI, The University of Ngaoundere,

[g]Laboratory of Simulation and Testing, The University Institute of Technology, The University of Ngaoundere,

**Abstract**

This work is concerned with the problem of computing of the reliability for multi-component systems given their dynamic fault trees. The recent literature provided satisfiability methods using the conjunctive normal form (CNF) of the structure function which is usually initially given under its disjunctive normal form (DNF) before using existing satisfiability (SAT) solvers. The conversion of the DNF of the structure function to its CNF is actually very costly. The main contribution of this paper is to propose an efficient method to compute the reliability polynomial using directly the DNF which is itself obtained efficiently from the dynamic fault tree. This involves the representation of logic functions by sets of three-level numeric tuples and the definition of logic operations adapted on these sets. The proposed method ensures polynomial complexity with respect to the number of nodes in the fault tree, the maximum number of inputs of the different logic gates being fixed.
**Keywords** : Reliability polynomial, dynamic fault tree, satisfiability, logic function.
**AMS Classification :** 03-01, 03-08, 03B05, 05-01, 05-08, 68R07, 90B25.

## 1 Introduction

Reliability is an important concept in the daily life of humans. Indeed, in the quest for development, decision-making is very often based on risk assessments of the occurrence of feared events. The latter are linked to the combination of other events considered as minor or elementary. This is particularly the case when studying the reliability of a composite system such as a complex production equipment (Lisnianski

*Corresponding author's Address: email: jauresfotsa@gmail.com, P.O. Box 455, ENSAI, The University of Ngaoundere

and Levitin, 2003; Stapelberg, 2009; Ben-Daya et al., 2012; Limnios and Oprisan, 2012; Bozzano et al., 2013) or a portfolio of financial assets (Connor et al., 2010). Indeed, each component of the equipment or each asset has intrinsic risks to which interactions may be added. To better understand the interactive structure of risks, numerous models have been developed in the literature (Duane, 1964; Howard, 1971; Hoem, 1972; Barlow and Proschan, 1975; Solovyev, 1979; Korolyuk and Turbin, 1982; Bobrowski, 1985; Barlow and Proschan, 1996; Grabski, 2002; Lisnianski and Levitin, 2003; Grabski, 2007; Stapelberg, 2009; Connor et al., 2010; Ben-Daya et al., 2012; Limnios and Oprisan, 2012). Risks being closely related to hazard, many probabilistic tools have been used to evaluate them (Duane, 1964; Howard, 1964; Mine and Osaki, 1970; Korolyuk and Turbin, 1982; Cocozza-Thivent, 1997; Singpurwalla, 2006; Marin and Robert, 2007; Chen et al., 2018; Wang et al., 2020; Yu et al., 2021). Among the existing in the literature there are fault trees, Reliability Block Diagrams, Bayesian networks, Petri nets, fuzzy logic, Markov and semi-Markov processes. Petri nets are particularly used after converting other formalisms, to dynamically evaluate the reliability by simulation (Kordic, 2008; Robidoux et al., 2009; Xing and Robidoux, 2009; Wu et al., 2011; Yang et al., 2011; Yan et al., 2017; Kabir and Papadopoulos, 2019; Rui et al., 2020; Fahmy et al., 2023). The more complex a system becomes (in terms of number of components or subsystems), the more difficult it is to study its reliability, regardless of the modeling formalism. For very complex systems, computational tools have been developed based on logic theory and asymptotic properties of random processes. Thus, using the increasing performance of computer and Monte Carlo existing approaches, it is possible to get very sharp approximations of reliability characteristics for a given system (Hastings, 1970; Gilks et al., 1996; Robert et al., 1999; Dubi, 2001; Marseguerra and Zio, 2002; Paredes et al., 2019). However, to reach a comfortable level of precision the computational cost is very high. In addition to the high computation time, computers consume energy and produce heat, the problems of global warming and rationalization of energy consumption being very actual [1] (Atmanand and Raman, 2009; Grubler et al., 2018; Tyagi et al., 2020; Emil and Diab, 2021). Even the various providers of cloud computing services are interested in reducing computing costs (Wu et al., 2015; Ndamlabin-Mboula et al., 2020; Ndamlabin-Mboula et al., 2021).

Since at least the second half of the twentieth century, scientists have been active in proposing solutions to evaluate the reliability of systems at a lower cost. Although quite old, the use of fault trees remains frequent, even if it must be adapted to the evolution of the needs (Vesely et al., 1981; Bozzano et al., 2013; Limnios, 2013; Tiejun and Shasha, 2020). Indeed, the construction of a fault tree is quite intuitive and easy to understand. The calculation of reliability by fault trees is done by combining the theory of logic and the laws of probabilities of intermediate events. Unfortunately, as mentioned above, the calculations become tedious when the structure under study is large. Several increasingly efficient algorithms have been proposed in the literature for various aspects of reliability calculations. For example, the famous MOCUS algorithm described in the reference (Fussell et al., 1974) is still used for the computation of minimum cuts or path sets. Several recent developments are presented in the book (Limnios, 2013) authored by Limnios and Oprisan. These are generally recursive and truncation methods. More recently, satisfiability techniques have been proposed and applied in references (Brinzei and Aubry, 2015; Aubry and Brinzei, 2016; Duroeulx et al., 2017; Hamaidia et al., 2018; Duroeulx et al., 2019). These works take up the tools of (Limnios, 2013) by integrating the dynamic and parametric aspect of reliability. Moreover, the calculation of the structure function (global reliability) is obtained by a hierarchical calculation. It uses a lattice called Hasse's digram and the minimal cut sets previously calculated. The theoretical computational cost of this approach remains high and its implementation on computer seems a priori difficult.

The main goal of the current work is to propose an efficient and effective computational method to evaluate reliability based on fault trees. The specific goals are (1): Adapt logic operations to a representation of logic functions in sets of three-level numeric tuples, (2): to prune fault tree's leaves and to construct

[1] See the sustainable development goals 7, 11,12 and 13 of the United Nation here https://sdgs.un.org/goals.

equivalent simplified lattice structure similar to Hasse's diagram, and (3): to propose recursive approach of constructing reliability polynomials. The rest of the paper is organized in five sections. A short background with adapted concepts is given in Section 2. Our contributions stand from Section 3 to Section 5 following to the announced specific goals. We end the main contain of the paper with a conclusion.

# 2 Background on satisfiability approach for computing reliability

In this section, we present a synthetic review on the concept of fault tree. We also recall a quite recent method for determination of the probability of an event using fault tree and the known probabilities of possible causes.

## 2.1 Definition and representation of fault tree

There is a wide literature on fault trees. Most of the definitions are more descriptive than formal. But the recurrent idea refers to a graph having a tree structure. Here we suggest a systematic and general definition.

**Definition 2.1 (Fault tree)** *Consider a sequence* $\mathscr{N} = \{\varphi_k\}_{k\in I\subseteq\mathbb{N}}$ *of boolean functions, each function* $\varphi_k$ *having* 0 *or* $n_k \in \mathbb{N}^\star$ *ordered entries labeled from* 1 *to* $n_k$. *Consider also a subset of* $\mathscr{E} \subseteq \mathscr{N}\times\mathscr{N}$ *and a map* $\mathscr{I} : \mathscr{E} \to \mathbb{N}^\star$ *such that* $\mathscr{I}(\varphi_{k_1}, \varphi_{k_2}) \in \{1, \ldots, n_{k_2}\}$ *and given* $\varphi_{k_2}$, *the function* $\mathscr{I}(\bullet, \varphi_{k_2})$ *is injective. Then a fault tree is valued directed graph* $(\mathscr{N}, \mathscr{E}, \mathscr{I})$ *without circuit and having exactly one output. The elements of* $\mathscr{N}$ *are called events. Any edge* $(\varphi_{k_1}, \varphi_{k_2}) \in \mathscr{E}$ *is called a causality relationship having* $\varphi_{k_1}$ *as origin and the entry* $\mathscr{I}(\varphi_{k_1}, \varphi_{k_2})$ *of* $\varphi_{k_2}$ *as extremity. An event without entry is said basic while an event which is never an origin of an edge is a top even. Other events are called intermediate.*

Usually in a fault trees theory, an event is at most the origin of one edge but we think that a this restriction is not necessary since an event can impact several other considered events. If one replaces each edge in Definition 2.1 by its inverse assuming that each event was at most the origin of one edge, then a classical structure of tree is recovered with the top event as the root and basic events as leaves.

The most common representation of fault trees is graphic. This is maybe because the charts are more intuitive and easy to understand. The graphical representation of fault trees is therefore that of a set of events interconnected by logic gates. The basic logic gates are **NOT**, **AND** and **OR** but dynamic aspects are progressively introduced through new logic gates such as **DELAY**, **PAND**, **SEQ**, **SPARE** and **FDEP**. We refer to (Fussell et al., 1974; Vesely et al., 1981; Limnios, 2013; Hamaidia et al., 2018) for further details on the graphical construction of fault tree. In order to take into consideration both the dynamic and the parametric aspect of fault events the authors in (Tiejun and Shasha, 2020) have introduced the concept of space fault tree. As we will see when defining dynamic logic gates later, a dynamic fault tree can be seen as increasing sequence of classical fault trees : past events are input for future events.

As common in graph theory, a fault tree can be represented with a matrix notation, $M_{i,j} = \ell$ meaning that $\mathscr{I}\left(\varphi_i, \varphi_j\right) = \ell$ if $\ell \neq 0$ and $\left(\varphi_i, \varphi_j\right) \notin \mathscr{E}$ otherwise. The matrix representation allows easier automatic computations on fault tree. Moving from the basic events to the top event is a successive composition of boolean functions leading to the structure function of the studied system (see Chapter 4 in (Limnios, 2013)). To efficiently provide a simplest expression of the structure function has always been an issue in literature. Regardless to the used method, the main tool to get the structure function remains boolean calculus. In the following, we recall the aim of the satisfiability approach and we present relatively recent methods for its implementation.

## 2.2 Description of the satisfiability approach

The satisfiability approach consists in getting a polynomial expression of the reliability of a given system where the variables are the reliability values of its components (Wang and Williams, 1991; Kamath et al., 1993; Limnios, 2013; Brinzei and Aubry, 2015; Aubry and Brinzei, 2016; Duroeulx et al., 2017; Duroeulx et al., 2019). That polynomial expression is obtained via another function called structure function. Such a function is a boolean expression depending on boolean state variables of the components or failure events in the system. Hence, a multi-state of the system with $n \in \mathbb{N}^*$ components is given as a boolean vector in $\{0,1\}^n$. By the structure of a fault tree it is relatively easy to get the structure function. However, the conversion of the structure function to the reliability polynomial if more difficult, especially if the structure function is not reduced to a minimal disjunctive normal form known as "sum of minimal ties". A tie or path set is a multi-state ensuring that the system is functioning. Introducing the partial order $\preceq$ defined in $\{0,1\}^n$ by $x \preceq y$ if $x_i \leq y_i$, $\forall i = 1,\ldots,n$, the set $\{0,1\}^n$ has a lattice structure. Thus, the expression "minimal tie" get sense and it is possible to organize the ties in a hierarchical diagram called Hasse's diagram. The latter allows to definitely compute the reliability (or satisfiability) polynomial (Aubry and Brinzei, 2016; Duroeulx et al., 2017). The authors in (Duroeulx et al., 2017) propose two algorithms to determine the minimal set of tie. The first one consists in browsing the set of ties and identifying minimal ties set through successive comparison. The set of ties is obtained by putting the structure function under its disjunctive normal form (sum of minterms). The second algorithm uses the minimal cuts set in order to get using Hasse's diagram principle, a reduced set of ties containing minimal ties set. The procedure of extracting minimal ties set is similar with the first algorithm. We recall that a cut is a multi-state ensuring that the system fails. Cuts are more natural to get using the disjunctive normal form of negation of structure function given by the fault tree. Unfortunately, all those algorithms are greedy and can display high complexity either for putting the structure function in a normal form or extracting the minimal set of ties. For example, if we assume that a $n$-component system is functioning when at least $k$ components are safe then the size of the set of ties is $(n!/(k!(n-k)!))$ and the complexity order in time of extracting minimal ties set is $(n!/(k!(n-k)!))^2$. The complexity order in memory for representing all the multi-states is $2^n$.

# 3 Representation and operations in logic functions spaces

In this section, we define logic operations in the set of sets of n-tuple of three level logic operations (negation, tautology and identity). A subset of $\{-1,0,1\}^n$ represents a sum of minterms. For example, the sets $\{(1,-1)\}$ and $\{(0,1),(1,-1)\}$ are equivalent to logic functions given respectively by $(p,q) \mapsto p \wedge \neg q$ and $(p,q) \mapsto q \vee (p \wedge \neg q)$. The sets $\{\}$ and $\{(0,0)\}$ denote the functions $(p,q) \mapsto False$ and $(p,q) \mapsto True$. Notice that a $n$-tuple containing zeros is a compact notation of a larger set of boolean vectors. For example $\{(0,1)\}$ is a compact notation of $\{(-1,1),(1,1)\}$. This is and advantage in the determination of minimal cuts set and minimal ties set.

As said before, we consider the following three elementary logic functions: (1) $-1$ : Negation, (2) $0$ : Tautology or True, and (3) $+1$ : Identity. In the following we state some definitions and provide illustrative examples.

**Definition 3.1** *Consider sequences $x, y \in \{-1,0,1\}^n$. We say that $x \geq y$ if $\forall i = 1,\ldots,n$, $x_i \geq y_i$ and $\sum_{i=1}^{n}(x_i - y_i) \leq 1$.*

**Example 3.1**

1. $(0,1) \geq (0,-1)$ *and* $(0,1) \geq (0,0)$.

2. $(0,1) \geq (-1,1)$ *and* $(-1,1) \geq (-1,-1)$*, but* $(0,1) \not\geq (-1,-1)$ *and* $(0,1) \not\leq (-1,-1)$.

3. $(0,1) \not\leq (-1,0)$ *and* $(0,1) \not\geq (-1,0)$.

The example 3.1 shows that the binary relation $\leq$ is reflexive, antisymmetric but not transitive. So $\leq$ is not an order. However, $\leq$ is important to check possibilities to reduce sums of minterms.

**Definition 3.2 (Disjunction: OR)** *Consider sequences* $x, y \in \{-1,0,1\}^n$*. We define the disjunction* $\{x\} \vee \{y\} = \{y\} \vee \{x\} \subseteq \{0,1\}^n$ *as the set satisfying* $\{x\} \vee \{y\} = \{x,y\}$ *if* $\sum_{i=1}^{n}(x_i - y_i) > 1$*, otherwise* $\{x\} \vee \{y\} = \{z\}$ *with* $z_i = x_i \mathbb{1}_{\{0\}}(x_i - y_i)$*. If* $A, B \subseteq \{-1,0,1\}^n$ *then* $A \vee B = \underset{(x,y)\in A\times B}{\cup} \{x\} \vee \{y\}$.

**Example 3.2**

1. $\{(0,1)\} \vee \{(0,-1)\} = \{(0,0)\}$.

2. $\{(0,1)\} \vee \{(-1,0)\} = \{(0,1),(-1,0)\}$.

3. $\{(0,1),(-1,0)\} \vee \{(-1,-1),(0,0)\} = \{(0,0),(-1,0),(0,1),(-1,-1)\}$.

**Proposition 3.1**
*According to Definition 3.2, the complexity in time for computing a disjunction* $A \vee B$ *is* $\Theta(n|A||B|)$ *when* $A, B \subseteq \{-1,0,1\}^n$.

**Proof 3.1** *The proof is based on the fact that the computation of* $A \vee B$ *is strongly related to the enumeration of the Cartesian product* $A \times B$*. The evaluation of each element* $\{x\} \vee \{y\}$ *pass through n comparisons.*

**Definition 3.3 (Conjunction: AND)** *Consider sequences* $x, y \in \{-1,0,1\}^n$*. We define the conjunction* $\{x\} \wedge \{y\} = \{y\} \wedge \{x\} \subseteq \{0,1\}^n$ *as the set satisfying* $\{x\} \wedge \{y\} = \{\}$ *if* $\min_{i=1}^{n} x_i y_i = -1$*, otherwise* $\{x\} \wedge \{y\} = \{z\}$ *with* $z_i = \max(-1, \min(1, x_i + y_i))$*. If* $A, B \subseteq \{-1,0,1\}^n$ *then* $A \wedge B = \underset{(x,y)\in A\times B}{\cup} \{x\} \wedge \{y\}$.

**Example 3.3**

1. $\{(0,1)\} \wedge \{(0,-1)\} = \{\}$.

2. $\{(0,1)\} \wedge \{(-1,0)\} = \{(-1,1)\}$.

3. $\{(0,1),(-1,0)\} \wedge \{(-1,-1),(0,0)\} = \{(-1,0),(0,1),(-1,-1)\}$.

**Proposition 3.2**
*According to Definition 3.3, the complexity in time for computing a conjunction* $A \wedge B$ *is* $\Theta(n|A||B|)$ *when* $A, B \subseteq \{-1,0,1\}^n$.

**Proof 3.2** *The proof is based on the fact that the computation of* $A \wedge B$ *is strongly related to the enumeration of the Cartesian product* $A \times B$*. The evaluation of each element* $\{x\} \wedge \{y\}$ *pass through n comparisons.*

**Definition 3.4 (Negation: NOT)** *Consider a sequence* $x = (x_1, \ldots, x_n) \in \{-1,0,1\}^n$*. We define its negation such as*

$$\neg\{x\} = \cup_{i=1, x_i \neq 0}^{n} \{(0, \ldots, 0, -x_i, 0, \ldots, 0)\} \tag{1}$$

*If* $A \subseteq \{-1,0,1\}^n$ *then* $\neg A = \underset{x\in A}{\wedge} \neg\{x\}$.

**Example 3.4**

1. $\neg\{(0,0)\} = \{\}$.
2. $\neg\{(0,1,-1)\} = \{(0,-1,0),(0,0,1)\}$.
3. $\neg\{(0,1),(-1,0)\} = \{(1,-1)\}$.

**Proposition 3.3**

*According to Definition 3.4, the complexity in time for computing a negation* $\neg A$ *is* $\Theta\left(n|A| + \frac{n^3\left(n^{|A|-1}-1\right)}{n-1}\right)$ *when* $A \subseteq \{-1,0,1\}^n$.

**Proof 3.3** *The proof is based on the fact that the negation is an iterative conjunction of elementary disjunctions.*

**Proposition 3.4**
*Let* $A, B, C \subseteq \{-1,0,1\}^n$. *Then the following statements hold.*

*(i)* $\neg(\neg A) = A$.

*(ii)* $A \vee B = B \vee A$ *and* $A \wedge B = B \wedge A$.

*(iii)* $(A \vee B) \vee C = A \vee (B \vee C)$ *and* $(A \wedge B) \wedge C = A \wedge (B \wedge C)$.

*(iv)* $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ *and* $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$.

*(v)* $\neg(A \vee B) = \neg A \wedge \neg B$ *and* $\neg(A \wedge B) = \neg A \vee \neg B$.

**Proof 3.4** *Since logic operations are equivalent to operations on sets, the announced properties holds. Indeed, negation is equivalent to passage to the complementary, disjunction is equivalent to union and conjunction is equivalent to intersection.*

**Definition 3.5 (Conjunction with priority: PAND)**
*Let* $\left\{A_{i,j}\right\}_{i,j=1}^{m} \subseteq \{-1,0,1\}^n$ *and* $\{P_i\}_{i=1}^{m} \subseteq \mathbb{R}^n$ *with* $P_{i_{(1)}} \leq P_{i_{(2)}} \leq \ldots, P_{i_{(m)}}$. *We define*

$$PAND\left(\left\{A_i, A_j\right\}, \left\{B_i, B_j\right\}, \left\{P_i, P_j\right\}\right) = \begin{cases} A_i \wedge \neg A_j \wedge B_i \wedge B_j, \text{ if } P_i < P_j \\ \neg A_i \wedge A_j \wedge B_i \wedge B_j, \text{ if } P_i > P_j \\ \left(\left(A_i \wedge A_j\right) \vee \left(\neg A_i \wedge \neg A_j\right)\right) \wedge B_i \wedge B_j, \text{ if } P_i = P_j \end{cases}, \tag{2}$$

$$\begin{aligned} PAND\left(\{A_i\}_{i=1}^{m}, \{B_i\}_{i=1}^{m}, \{P_i\}_{i=1}^{m}\right) &= \wedge_{i,j=1}^{m} PAND\left(\left\{A_i, A_j\right\}, \left\{B_i, B_j\right\}, \left\{P_i, P_j\right\}\right) \\ &= \wedge_{k=1}^{m-1} PAND\left(\left\{A_{i_{(k)}}, A_{i_{(k+1)}}\right\}, \left\{B_{i_{(k)}}, B_{i_{(k+1)}}\right\}, \left\{P_{i_{(k)}}, P_{i_{(k+1)}}\right\}\right). \end{aligned} \tag{3}$$

**Definition 3.6 (Conjunction with sequence: SEQ)**
*Let* $\left\{A_{i,j}\right\}_{i,j=1}^{m} \subseteq \{-1,0,1\}^n$ *and* $\{P_i\}_{i=1}^{m} \subseteq \mathbb{R}^n$ *with* $P_{i_{(1)}} \leq P_{i_{(2)}} \leq \ldots, P_{i_{(m)}}$. *We define*

$$SEQ\left(\{A_{1,1}, A_{1,2}, A_{2,1}, A_{2,2}\}, \{P_1, P_2, \}\right) = PAND\left(\{A_{1,1}, A_{1,1}\}, \{A_{2,1}, A_{2,2}\}, \{P_1, P_2\}\right) \tag{4}$$

$$SEQ\left(\left\{A_{i,j}\right\}_{i,j=1}^{m}, \{P_i\}_{i=1}^{m}\right) = \wedge_{k=1}^{m-1}\left(\left(\wedge_{\ell=1}^{k} A_{k,i_{(\ell)}}\right) \wedge \left(\wedge_{\ell=k+1}^{m}\left(P_{i_{(k)}} = P_{i_{(\ell)}}\right) \vee \neg A_{k,i_{(\ell)}}\right)\right). \tag{5}$$

**Definition 3.7 (Standby or SPARE)** *Let* $\{A_i\}_i^m, \{B_i\}_i^m \subset \{-1, 0, 1\}^n$. *We define*

$$SPARE\left(\{A_i\}_{i=1}^m, \{B_i\}_{i=1}^m\right) = \left(\vee_{i=1}^m \left(A_i \wedge \left(\wedge_{j=1, j\neq i}^m \neg A_j\right)\right)\right) \wedge \left(\wedge_{i=1}^m B_j\right) \tag{6}$$

The PAND, SEQ and SPARE gates correspond to their classical definition in terms of dynamic gates is the inputs $A_i$s denote the state of entries at a given time $t$, the inputs $B_i$s denote the state of entries at a given next time $t + dt$, and the inputs $P_i$s denote the decremental priorities in the sequence of failures. The description of existing logic gates is complete here because the possible definition of gate is not exhaustive when basic gate are already defined [2].

# 4 Hierarchical calculus of minimal cuts set

In this section we show how the fault tree is converted in a lumped tree and how the minimal set of cuts is computed hierarchical. Indeed, the nodes of the lumped fault tree are outputs or the gates appearing in the initial fault tree. Using notations and operations defined in Section 3, it is possible to get a set containing all the minimal cuts. Assuming that the considered system has $n$ components, the procedure of getting the lumped fault tree is given by the Algorithm 1.

**Algorithm 1** : Lumped Fault Tree

Consecutively number the nodes of the fault tree (components or gates) in such way that each node has an order greater than its son;
Set N as the number of node;
**for** $(i \in \{1, \ldots, N - n\})$ **do**
  Create a node numbered $i$;
  Assign a code $(id_i, x_{i+n}, y_{i+n}) \in IdSet \times \{0, 1\}^{2N} \times \mathbb{R}^N$ to the gate numbered $i + n$ depending on its identifier $id_i$, its input selection $(x_i)$ and its priorities $(y_i)$;
**end for**
**for** $(i \in \{1, \ldots, N - n\})$ **do**
  **for** $(j \in \{1, \ldots, N - n\})$ **do**
    **if** $(x_{i+n,j+n} = 1)$ **then**
      Generate directed edges $(i, j)$;
    **end if**
  **end for**
**end for**

Following Algorithm 1, the lumped fault tree has $N - n$ nodes and necessarily a lower number of edges.

**Example 4.1**
*Consider a dynamic fault tree given by the Figure 1. The considered system has three components. The circles in blue color represent the state of components since the last event in the system while the circles in red color represent the current state of components. All the gates are represented with black color.*

*The reduced associated fault tree is given in Figure 2.*

[2] This justifies according to (Fussell et al., 1974), why MOCUS program only allows AND and OR logic gates

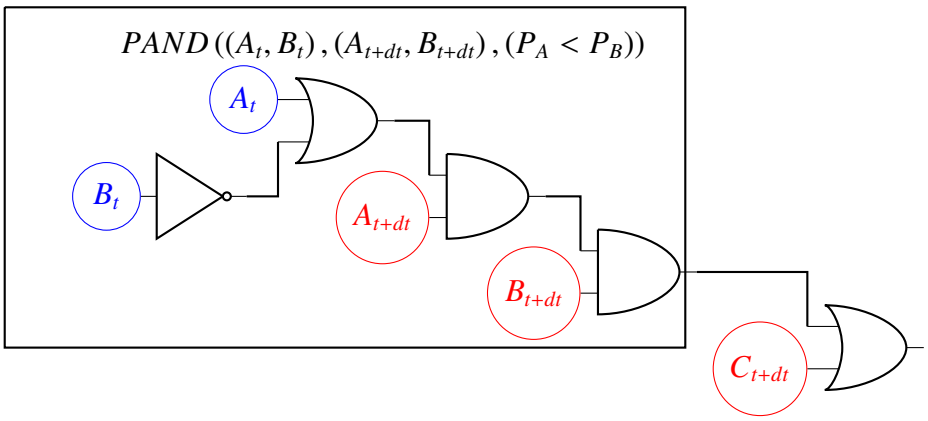


Figure 1: A dynamic fault tree

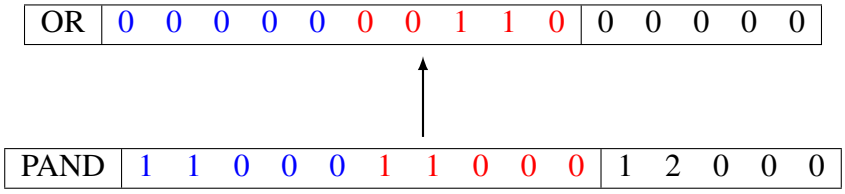


Figure 2: Reduced fault tree with coded nodes

*Figure 2 corresponds to the Table 1.*

Table 1: Logic gates

| **Order** | **Gate Id** | **Inputs** | | **Priorities** |
|---|---|---|---|---|
| | | **Previous state** | **Current state** | |
| 4 | PAND | (1, 1, 0, 0, 0) | (1, 1, 0, 0, 0) | (1, 2, 0, 0, 0) |
| 5 | OR | (0, 0, 0, 0, 0) | (0, 0, 1, 1, 0) | (0, 0, 0, 0, 0) |

Instead of computing directly the reliability one can first compute the unreliability function given the minimal cuts set. Determining cuts consists in successive evaluation of logic sets according to the gates appearing in the fault tree. The corresponding procedure is given in the Algorithm 2.

**Example 4.2** *Consider the fault tree with the table of gates given in Example 4.1.*

$$
\begin{aligned}
V_9 =& evaluateGate\,(PAND, \{\{(1,0,0,0,0,0,0,0,0,0)\}, \{(0,1,0,0,0,0,0,0,0,0)\},\\
&\{(0,0,0,0,0,1,0,0,0,0)\}, \{(0,0,0,0,0,0,1,0,0,0)\}\}, (1,2,0,0,0))\\
=& PAND\,(\{\{(1,0,0,0,0,0,0,0,0,0)\}, \{(0,1,0,0,0,0,0,0,0,0)\}\},\\
&\{\{(0,0,0,0,0,1,0,0,0,0)\}, \{(0,0,0,0,0,0,1,0,0,0)\}\}, (1,2))\\
=& \{(1,0,0,0,0,1,1,0,0,0), (0,-1,0,0,0,1,1,0,0,0)\}
\end{aligned}
$$

$$
\begin{aligned}
V_{10} =& evaluate\,(OR, \{V_9, \{(0,0,0,0,0,0,0,0,1,0)\}\}, (0,0,0,0,0))\\
=& V_9 \vee \{(0,0,0,0,0,0,0,0,1,0)\}\\
=& \{(0,0,0,0,0,0,0,1,0,0), (1,0,0,0,0,1,1,0,0,0), (0,-1,0,0,0,1,1,0,0,0)\}.
\end{aligned}
$$

**Algorithm 2** : Determination of cuts

**for** $(i \in \{1, \ldots, 2N\})$ **do**
  **if** $(i \leq N + n)$ **then**
    $V_i = \left\{\left(0, \ldots, 0, x_j, 0, \ldots, 0\right)\right\}$;
  **else**
    Evaluate $V_i \subseteq \{-1, 0, 1\}^{2N}$ according to the code $(Id_i, x_i, y_i)$ :

$$V_i = evaluateGate\left(id_i, \left\{V_j; x_{i,j} = 1\right\}, y_i\right);$$

  **end if**
**end for**

**Proposition 4.1** *The set of cuts obtained after applying Algorithm 2 contains all the minimal cuts.*

**Proof 4.1** *The principle of disjunction defined in Definition 3.2 allows when possible to obtained reduced logic vectors by evaluating all the couples of the corresponding Cartesian product. This reduction is made possible thanks to the consideration of the elementary contingency value* 0.

# 5 Recursive calculus of reliability polynomial

In this section we show how to use the lumped fault tree to recursively compute the reliability polynomial. Depending on the reliabilities $R_i$s of the components, we aim at computing the unreliability function $U = 1 - R$ depending on the unreliabilities $U_i = 1 - R_i$ $i = 1, \ldots, n$. This is possible through computing the minimal cuts set and applying the Hasse's diagram algorithm described in (Aubry and Brinzei, 2016; Duroeulx et al., 2017) (see Figure 5 and Figure 1 respectively). We provide here an alternative method to compute the unreliability polynomial using a set of cuts.

**Proposition 5.1** *Consider the function* $\tau_i : \{-1, 0, 1\} \times \mathbb{R} \to [0, 1]$ *satisfying* $\tau_i(0) = 1$, $\tau_i(-1) = R_i = 1 - U_i$ *and* $\tau_i(1) = U_i$. *The unreliability polynomial of the empty set is* 1. *The unreliability polynomial of a singleton* $\{(x_1, \ldots, x_n)\}$ *is* $U(\{(x_1, \ldots, x_n)\}) = \prod_{i=1}^{n} \tau_i(x_i)$. *The unreliability polynomial of a conjunction* $A \vee B$ *is* $U(A \vee B) = U(A) + U(B) - U(A \wedge B)$. *Precisely, if* $x \in A \subseteq \{-1, 0, 1\}^n$ *and* $B = A \setminus \{x\}$ *then* $U(A) = U(A) + U(\{x\}) - U(\{x\} \wedge B)\}$. *Moreover, the complexity in time for computing the unreliability polynomial of* $A$ *is* $\Theta\left(n\left(\left(|A| + 1\right) 2^{|A|-1} - |A| - \sum_{\ell=1}^{|A|-2} \ell 2^{\ell}\right)\right)$.

**Proof 5.1** *As said earlier logic operations are equivalent to operations on sets. Moreover, either reliability or unreliability are probabilities and using the inclusion-exclusion development*[3]*, one has*

$$\mathbb{P}\left(\cup_{i=1}^{n} A_i\right) = \sum_{k=1}^{n} \sum_{i_1, i_2, \ldots, i_k} (-1)^{k-1} \mathbb{P}\left(\cap_{j=1}^{k} A_{i_j}\right). \tag{7}$$

*If* $\Theta(C)$ *is the complexity for computing* $U(B)$ *then the complexity for computing* $U(A)$ *is* $\Theta(2C + n|A|)$. *Indeed, the complexity of computing* $\{x\} \wedge B$, $U(\{x\})$ *and* $U(\{x\} \wedge B)$ *are respectively* $\Theta(n(|A| - 1))$, $\Theta(n)$ *and* $C$ . *By recurrence, the announced complexity is obtained.*

[3] See the book (Limnios, 2013).

**Example 5.1** *Consider the fault tree with the table of gates given in Example 4.1. Based on the set of cuts*

$$V_{10} = \{(0,0,0,0,0,0,0,1,0,0),(1,0,0,0,0,1,1,0,0,0),(0,-1,0,0,0,1,1,0,0,0)\}$$

*and letting $L(t)$ denotes the time of the last change in the system before t, the unreliability polynomial is*

$$\begin{aligned}
U(t) =& U(V_{10}) \\
=& U(\{(0,0,0,0,0,0,0,1,0,0)\}) + U(\{(1,0,0,0,0,1,1,0,0,0)\}) + U(\{(0,-1,0,0,0,1,1,0,0,0)\}) \\
&- U(\{(0,0,0,0,0,0,0,1,0,0)\} \wedge \{(1,0,0,0,0,1,1,0,0,0)\}) \\
&- U(\{(0,0,0,0,0,0,0,1,0,0)\} \wedge \{(0,-1,0,0,0,1,1,0,0,0)\}) \\
&- U(\{(1,0,0,0,0,1,1,0,0,0)\} \wedge \{(0,-1,0,0,0,1,1,0,0,0)\}) \\
&+ U(\{(0,0,0,0,0,0,0,1,0,0)\} \wedge \{(1,0,0,0,0,1,1,0,0,0)\} \wedge \{(0,-1,0,0,0,1,1,0,0,0)\}).
\end{aligned}$$

*This leads to the reliability polynomial*

$$\begin{aligned}
R(t) =& (1-(1-R_1(L(t))(1-R_2(L(t))))(1-R_6(t))(1-R_7(t)))R_8(t) \\
&(1-(1-R_1(L(t))(1-R_2(L(t))))(1-R_1(t-L(t)))(1-R_2(t-L(t))))R_3(t-L(t)). \qquad (8)
\end{aligned}$$

As one can observe in Algorithm 2 and Proposition 5.1 the complexity of computing the unreliability polynomial is exponential either in time or in memory. In terms of time, the complexity mainly depends on the size of the set which can be of the order of $4^N$ since the codification of logic vectors is under $2N$ bits. The complexity in memory is related to the storage of those logic vector sets. We claim that evaluation recursively the unreliability polynomial at each node of the lumped fault tree significantly reduces the complexity. We then propose the Algorithm 3.

**Algorithm 3** : Recursive determination of unreliability polynomial

**for** $(i \in \{1,\dots,2N\})$ **do**
  **if** $(i \leq n+N)$ **then**
    Evaluate $U_i$;
  **else**
    Set $Id_i$ as the identifier of gate numbered $i-N$;
    Set $x_i$ as the set of the identifiers of the inputs to the gate numbered $i-N$;
    Set $y_i$ as the set of the priorities of the inputs to the gate numbered $i-N$;
    Evaluate $V_i \subseteq \{-1,0,1\}^{|x_i|}$ according to $(Id_i, x_i, y_i)$ :

$$V_i = evaluateGate\left(id_i, \left\{\left\{z \in \{0,1\}^{|x_i|}\right\}; \sum_{j=1}^{|x_i|} |z_j| = 1\right\}, y_i\right);$$

    Evaluate $U_i$ according to $V_i$ and $\left\{U_j; j \in x_i\right\}$;
  **end if**
**end for**

**Example 5.2** *Consider again the fault tree with the table of gates given in Example 4.1.*

- $i \in \{1,2,3,4,5,6,7,8\}$ *:* $U_i = 1 - R_i$
- $i = 9$ *:*

- $x_9 = \{1, 2, 6, 7\}$
- $y_9 = \{1, 2\}$
- $V_9$ :

$$\begin{aligned} V_9 &= PAND\left(\{(1,0,0,0),(0,1,0,0)\},\{(0,0,1,0),(0,0,0,1)\},(1,2)\right) \\ &= \{(1,0,1,1),(0,-1,1,1)\} \end{aligned}$$

- $U_9 = U_1U_6U_7 + R_2U_6U_7 - U_1R_2U_6U_7 = (U_1 + R_1R_2)\,U_6U_7 = (1 - R_1\,(1 - R_2))\,(1 - R_6)\,(1 - R_7)$

- $i = 10$ :
  - $x_{10} = \{8, 9\}$
  - $y_{10} = \{\}$
  - $V_{10}$ :

$$\begin{aligned} V_{10} &= OR\left(\{(1,0)\},\{(0,1)\}\right) \\ &= \{(1,0),(0,1)\} \end{aligned}$$

  - $U_{10} = U_8 + U_9 - U_8U_9 = 1 - R_8R_9 = 1 - (1 - R_1\,(1 - R_2))\,(1 - R_6)\,(1 - R_7)\,R_8$

*It is then straightforward to get* $R_{10} = R = (1 - R_1\,(1 - R_2))\,(1 - R_6)\,(1 - R_7)\,R_8$ *as given previously in Example 5.1.*

**Theorem 5.1** *Assume that the lumped fault tree obtained in Algorithm 1 has m nodes each having n sons. Then the complexity in time for computing the unreliability polynomial is* $\Theta\left(mn\left((n+1)\,2^{n-1} - n - \sum_{\ell=1}^{n-2}\ell 2^{\ell}\right)\right)$.

**Proof 5.2** *The proof is based on Proposition 5.1, replacing at each node the term* $|A|$ *by n. Cumulating the complexities at each of the m nodes, we get the announced result.*

The Theorem 5.1 shows that if the fault tree is rearranged so that the number of entry of each gate is as small as possible then, although the number of gates ($m$) increases, the complexity of computing the unreliability polynomial becomes polynomial and not exponential as it is initially.

# 6 Conclusion

The main goal of this work was to propose an efficient and effective computational method to evaluate reliability based on fault trees. Indeed, the recent satisfiability method is costly and seems difficult to practically implement. Hence, the specific objectives of this work were (1): to define new logic operations on sets of "boolean vectors", (2): to prune fault tree's leaves and to construct equivalent simplified lattice structure similar to Hasse's diagram, and (3): to propose recursive approach of constructing reliability polynomials.

We successfully achieved the construction of adapted logic operations on sets of three-level numeric tuples representing logic functions. Those logic operators are equivalent to symbolic calculus of proposition, but it is easier and synthetic to use them. The proposed logic operators are natural to implement on computers since they operate on tuples having their components in $\{-1, 0, 1\}$. We also succeed in stating two algorithms permitting respectively to prune given fault tree and to evaluate the minimal set of cuts. Our last achievement was to provide an algorithm for computing recursively the reliability polynomial having in

mind that it also to 1 minus the reliability polynomial. It is shown conditionally upon the number of entries of all logic gates is bounded, that applying the last algorithm to the pruned fault tree makes the complexity to be polynomial instead of being exponential.

As the immediate perspective of this work, the authors are actively coding a python package for computing reliability polynomial based on dynamic fault trees given as entries. The latter named "DFT2RP" will be available very soon on https://pypi.org/ or as an Application Program Interface (API), and it will be exhaustively documented on https://github.com/.

## Acknowledgment

We thank the Cameroonian Association for the Promotion of Research and Development of Innovative Solutions for Sustainable Development (ARDIS4SD) for the enriching discussions related to this paper and the facilities offered to the first author. The authors are also grateful to the reviewers for their sound advice.

## Competing interests

The authors declare there is no competing interest.

## References

Atmanand, A. K. G. and Raman, R. (2009). Energy and sustainable development-an indian perspective. *World Academy of Science, Engineering and Technology*, 30:2009.

Aubry, J.-F. and Brinzei, N. (2016). Calcul direct de la fiabilité d'un système par son graphe ordonné. *Congrès Lambda Mu 20 de Maîtrise des Risques et de Sûreté de Fonctionnement, 11-13 Octobre 2016, Saint Malo, France*.

Barlow, R. and Proschan, F. (1975). International series in decision processes, statistical theory of reliability and life testing.

Barlow, R. E. and Proschan, F. (1996). *Mathematical theory of reliability*, volume 17. Siam.

Ben-Daya, M., Duffuaa, S. O., and Raouf, A. (2012). *Maintenance, modeling and optimization*. Springer Science & Business Media.

Bobrowski, D. (1985). Models and mathematical methods in reliability theory. *WNT, Warszawa*.

Bozzano, M., Cimatti, A., and Mattarei, C. (2013). Efficient analysis of reliability architectures via predicate abstraction. In *Hardware and Software: Verification and Testing: 9th International Haifa Verification Conference, HVC 2013, Haifa, Israel, November 5-7, 2013, Proceedings 9*, pages 279–294. Springer.

Brinzei, N. and Aubry, J.-F. (2015). An approach of reliability assessment of systems based on graphs models. In *European Safety and Reliability Conference ESREL 2015*, pages 1485–1493. CRC Press/Balkema, Taylor & Francis Group.

Chen, C., Liu, X., Chen, H.-H., Li, M., and Zhao, L. (2018). A rear-end collision risk evaluation and control scheme using a bayesian network model. *IEEE Transactions on Intelligent Transportation Systems*, 20(1):264–284.

Cocozza-Thivent, C. (1997). *Processus stochastiques et fiabilité des systèmes*, volume 28. Springer Science & Business Media.

Connor, G., Goldberg, L. R., and Korajczyk, R. A. (2010). *Portfolio risk analysis*. Princeton University Press.

Duane, J. (1964). Learning curve approach to reliability monitoring. *IEEE transactions on Aerospace*, 2(2):563–566.

Dubi, A. (2001). Modeling of realistic system with the monte carlo method: A unified system engineering approach. In *Proceedings of the Annual Reliability and Maintainability Symposium*.

Duroeulx, M., Brinzei, N., Duflot, M., and Merz, S. (2017). Satisfiability techniques for computing minimal tie sets in reliability assessment. *hal-01518920*.

Duroeulx, M., Brinzei, N., Duflot, M., and Merz, S. (2019). Integrating satisfiability solving in the assessment of system reliability modeled by dynamic fault trees. In *29th European Safety and Reliability Conference, ESREL 2019*. Research Publishing Services.

Emil, F. and Diab, A. (2021). Energy rationalization for an educational building in egypt: Towards a zero energy building. *Journal of Building Engineering*, 44:103247.

Fahmy, M., Ahmed, A., and Khalil, M. (2023). On reliability: A mathematical fault tree. *Reliability: Theory & Applications*, 18(1 (72)):78–85.

Fussell, J., Henry, E., and Marshall, N. (1974). Mocus: A computer program to obtain minimal sets from fault trees. Technical report, Aerojet Nuclear Co., Idaho Falls, Idaho (USA).

Gilks, W., Richardson, S., and Spiegelhalter, D. (1996). Markov chain monte carlo in practice.

Grabski, F. (2002). Semi-markov models of reliability and operation. *IBS PAN Warsaw*.

Grabski, F. (2007). Applications of semi-markov processes in reliability. *Journal of Polish Safety and Reliability Association*, 1.

Grubler, A., Wilson, C., Bento, N., Boza-Kiss, B., Krey, V., McCollum, D. L., Rao, N. D., Riahi, K., Rogelj, J., De Stercke, S., et al. (2018). A low energy demand scenario for meeting the 1.5 c target and sustainable development goals without negative emission technologies. *Nature energy*, 3(6):515–527.

Hamaidia, M., Kara, M., and Innal, F. (2018). Probability and frequency derivation using dynamic fault trees. *Process Safety Progress*, 37(4):535–552.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

Hoem, J. M. (1972). Inhomogeneous semi-markov processes, select actuarial tables, and duration-dependence in demography. In *Population dynamics*, pages 251–296. Elsevier.

Howard, R. A. (1964). Research in semi-markovian decision structures. *J. Oper. Res. Soc. Japan*, 6(4):163–199.

Howard, R. A. (1971). Dynamic probabilistic systems, volume 2: Semi-markov and decision processes.

Kabir, S. and Papadopoulos, Y. (2019). Applications of bayesian networks and petri nets in safety, reliability, and risk assessments: A review. *Safety science*, 115:154–175.

Kamath, A., Karmarkar, N., Ramakrishnan, K., and Resende, M. (1993). An interior point approach to boolean vector function synthesis. In *Proceedings of 36th Midwest Symposium on Circuits and Systems*, pages 185–189. IEEE.

Kordic, V. (2008). *Petri Net, Theory and Applications*. IntechOpen.

Korolyuk, V. and Turbin, A. (1982). Markov renewal processes in problems of systems reliability. *Naukova Dumka*.

Limnios, N. (2013). *Fault trees*. John Wiley & Sons.

Limnios, N. and Oprisan, G. (2012). *Semi-Markov processes and reliability*. Springer Science & Business Media.

Lisnianski, A. and Levitin, G. (2003). *Multi-state system reliability: assessment, optimization and applications*, volume 6. World Scientific Publishing Company.

Marin, J.-M. and Robert, C. (2007). *Bayesian core: a practical approach to computational Bayesian statistics*. Springer Science & Business Media.

Marseguerra, M. and Zio, E. (2002). Basics of the monte carlo method with application to system reliability.

Mine, H. and Osaki, S. (1970). Markovian decision processes.

Ndamlabin-Mboula, J. E., Kamla, V. C., and Djamegni, C. T. (2020). Cost-time trade-off efficient workflow scheduling in cloud. *Simulation Modelling Practice and Theory*, 103:102107.

Ndamlabin-Mboula, J. E., Kamla, V. C., and Tayou Djamégni, C. (2021). Dynamic provisioning with structure inspired selection and limitation of vms based cost-time efficient workflow scheduling in the cloud. *Cluster Computing*, 24(3):2697–2721.

Paredes, R., Dueñas-Osorio, L., Meel, K. S., and Vardi, M. Y. (2019). Principled network reliability approximation: A counting-based approach. *Reliability Engineering & System Safety*, 191:106472.

Robert, C. P., Casella, G., and Casella, G. (1999). *Monte Carlo statistical methods*, volume 2. Springer.

Robidoux, R., Xu, H., Xing, L., and Zhou, M. (2009). Automated modeling of dynamic reliability block diagrams using colored petri nets. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(2):337–351.

Rui, L., Chen, X., Gao, Z., Li, W., Qiu, X., and Meng, L. (2020). Petri net-based reliability assessment and migration optimization strategy of sfc. *IEEE Transactions on Network and Service Management*, 18(1):167–181.

Singpurwalla, N. D. (2006). *Reliability and risk: a Bayesian perspective*. John Wiley & Sons.

Solovyev, A. (1979). Analytical methods of reliability theory. *Warsaw: WN-T*.

Stapelberg, R. F. (2009). *Handbook of reliability, availability, maintainability and safety in engineering design*. Springer Science & Business Media.

Tiejun, C. and Shasha, L. (2020). *Space fault tree theory and system reliability analysis*. edp sciences.

Tyagi, R., Vishwakarma, S., Singh, K. K., and Syan, C. (2020). Low-cost energy conservation measures and behavioral change for sustainable energy goal. *Affordable and clean energy. Encyclopedia of the UN Sustainable Development Goals. Springer, Cham. https://doi. org/10.1007/978-3-319-71057-0_155-1*.

Vesely, W. E., Goldberg, F. F., Roberts, N. H., and Haasl, D. F. (1981). Fault tree handbook. Technical report, Nuclear Regulatory Commission Washington DC.

Wang, C. and Williams, A. (1991). The threshold order of a boolean function. *Discrete Applied Mathematics*, 31(1):51–69.

Wang, Y., Li, J., Wu, Z., Chen, J., Yin, C., and Bian, K. (2020). Dynamic risk evaluation and early warning of crest cracking for high earth-rockfill dams through bayesian parameter updating. *Applied Sciences*, 10(21):7627.

Wu, F., Wu, Q., and Tan, Y. (2015). Workflow scheduling in cloud: a survey. *The Journal of Supercomputing*, 71:3373–3418.

Wu, J., Yan, S., and Xie, L. (2011). Reliability analysis method of a solar array by using fault tree analysis and fuzzy reasoning petri net. *Acta Astronautica*, 69(11-12):960–968.

Xing, L. and Robidoux, R. (2009). Drbd: Dynamic reliability block diagrams for system reliability modelling h. xu. *International Journal of Computers and Applications*, 31(2).

Yan, R., Jackson, L. M., and Dunnett, S. J. (2017). Automated guided vehicle mission reliability modelling using a combined fault tree and petri net approach. *The International Journal of Advanced Manufacturing Technology*, 92(5-8):1825–1837.

Yang, X., Li, J., Liu, W., and Guo, P. (2011). Petri net model and reliability evaluation for wind turbine hydraulic variable pitch systems. *Energies*, 4(6):978–997.

Yu, Q., Liu, K., Yang, Z., Wang, H., and Yang, Z. (2021). Geometrical risk evaluation of the collisions between ships and offshore installations using rule-based bayesian reasoning. *Reliability Engineering & System Safety*, 210:107474.