Intrinsic ordering, combinatorial numbers and reliability engineering

Luis González*

Research Institute of Intelligent Systems & Numerical Applications in Engineering, Department of Mathematics, University of Las Palmas de Gran Canaria, Campus de Tafira, 35017 Las Palmas de Gran Canaria, Spain

Abstract

A new algorithm for evaluating the top event probability of large fault trees (FTs) is presented. This algorithm does not require any previous qualitative analysis of the FT. Indeed, its efficiency is independent of the FT logic, and it only depends on the number n of basic system components and on their failure probabilities. Our method provides exact lower and upper bounds on the top event probability by using new properties of the intrinsic order relation between binary strings. The intrinsic order enables one to select binary n-tuples with large occurrence probabilities without necessity to evaluate them. This drastically reduces the complexity of the problem from exponential (2^n binary n-tuples) to linear (n Boolean variables). Our algorithm is mainly based on a recursive formula for rapidly computing the sum of the occurrence probabilities of all binary n-tuples with weight m whose 1s are placed among the k right-most positions. This formula, as well as the balance between accuracy and computational cost, is closely related to the famous Pascal's triangle.

Keywords: Reliability engineering modelling; fault tree analysis; top event probability; intrinsic order; intrinsic order graph; Hamming weight

*Tel.: +34 928458832
Email address: luisglez@dma.ulpgc.es (Luis González)

1. Introduction

One of the most common techniques for analyzing system safety and reliability is the fault tree analysis (FTA), an extensively used method worldwide, which is mainly based on probability theory and Boolean algebra [1, 2, 3]. Large, complex systems where FTA has been widely applied can be found in many different scientific or engineering areas, such as aeronautics, chemistry and materials, energy resources, mechanics, meteorology and climatology, nuclear physics, robotics, etc. (see, e.g., [4] for an exhaustive list of such subject categories).

FTA was first conceived in 1961 by H. A. Watson of Bell Telephone Laboratories to perform a safety evaluation of the Minuteman Launch Control System [5]. The basic idea of FTA is the translation of the failure behavior of a technical system into a visual diagram involving logic gates, i.e., the fault tree (FT) [6]. A FT is a logical representation of the manner in which combinations of failures of the basic components of a system lead to the undesired state of the system, the so-called top event [7, 8].

One of the main topics in FTA is to estimate the failure probability of the whole system –also called the system unavailability, or the top event probability. Many different techniques have been proposed in the literature to evaluate the system unavailability; see, e.g., [6, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21] and the references contained therein.

Throughout this paper, we consider complex systems depending on an arbitrarily large number n of statistically independent random Boolean variables $x_1, \ldots, x_n$. The usual convention is to assign the value 1 or 0 to variable x_i if component i fails or works, respectively. So, each one of the binary n-tuples of 0s and 1s, $u = (u_1, \ldots, u_n) \in \{0,1\}^n$, describes the current situation of the n basic components of the system (failing or working) and, in the following, we shall refer to these binary n-tuples as binary strings, or as system elementary states.

Then, using a more precise algebraic terminology, the FT can be described by a stochastic Boolean function

$$\begin{gathered}\Phi : \{0,1\}^n \longrightarrow \{0,1\} \\ (x_1, \ldots, x_n) \mapsto \Phi(x_1, \ldots, x_n)\end{gathered}$$

depending on the n basic variables x_i of the system. Assuming that $\Phi = 1$ if the system fails, $\Phi = 0$ otherwise, then the failure probability of the whole system can be evaluated by computing the probability $\Pr\{\Phi = 1\}$.

Let us recall that a fault tree is said to be coherent if its structure function Φ satisfies the following two conditions (see, e.g., [22, 23, 24])

(i) Relevance. Each component x_i is relevant, i.e., it contributes to the system state. Formally, for all $i = 1, 2, \ldots, n$

$$\Phi(0_i, x) \neq \Phi(1_i, x), \quad \text{for some vector } x,$$

where

$$\begin{aligned} \Phi(0_i, x) &= \Phi(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n), \\ \Phi(1_i, x) &= \Phi(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n). \end{aligned}$$

(ii) Monotonicity. The system neither transits from a failed state to a good state by the failure of a component, nor transits from a good state to a failed state by the repair of a component. Formally, the structure function Φ is non-decreasing in each variable, i.e.,

$$\Phi(x) \geq \Phi(y) \quad \text{if } \; x > y,$$

where $x, y \in \{0,1\}^n$, and $x > y$ means that $x_i \geq y_i$ for every component $i = 1, 2, \ldots, n$, with $x_i > y_i$ for some i.

When at least one of the two above conditions is not fulfilled, the system is non-coherent.

Most of FTA methods for evaluating the top event probability $\Pr\{\Phi = 1\}$ are based on the minimal cut set approach [8]. A (minimal) cut set is a (minimal) combination of component failures that leads to the system failure (top event). However, for large, complex FTs it is not possible, in general, to enumerate all its minimal cut sets due to high memory requirements and long computing time, since the number of potential minimal cut sets exponentially increases, in general, with the size of the FTs, that is, with the numbers of their basic events and gates [13].

The main goal of this paper is to present a new algorithm for obtaining exact lower and upper bounds on the top event probability (for both coherent and non-coherent systems), with no need to find the minimal cut sets. Moreover, indeed our method does not require any previous qualitative analysis of the FT to estimate the system unavailability.

At the end of this paper, we will explain in detail the characteristics and advantages of our proposed method. At this moment, let us mention that its main advantage, compared with many other techniques proposed in the literature to obtain exact lower and upper bounds on system unavailability, is the following. Our algorithm does not require any knowledge or information (often required by other techniques) about the Boolean (structure) function of the FT. Moreover, we shall prove that it is possible (and easy) to assure a priori the maximum admissible error in the estimation, since this error is independent of the FT to be evaluated, and it only depends on the occurrence probabilities of certain selected binary n-tuples.

The bounds on system unavailability can be obtained from the occurrence probabilities of any subset of selected system elementary states. However, the main point is that the accuracy in the estimation (difference between upper and lower bounds) of $\Pr\{\Phi = 1\}$ improves at the same time as the sum of the occurrence probabilities of all the selected elementary states increases. Hence, in order to get a good compromise between accuracy and computational cost, we need to select as few binary n-tuples u as possible, with occurrence probabilities $\Pr\{u\}$ as large as possible.

For selecting the binary strings u with large occurrence probabilities, we use a theorem that states new properties related to the intrinsic order criterion (IOC). IOC is a simple, positional criterion that allows us to compare the occurrence probabilities $\Pr\{u\}$, $\Pr\{v\}$ of two given binary n-tuples u, v with no need to evaluate them, simply looking at the relative positions of their 0s and 1s. IOC was first described in [25]: a theoretical paper where the intrinsic ordering is introduced in the context of the evaluation of stochastic Boolean functions.

Other theoretical results and practical applications of IOC can be found in [26, 27, 28, 29, 30, 31, 32]. Let us briefly describe the main ideas or techniques presented in these works. In [26], different characterizations as well as necessary conditions and sufficient conditions for the intrinsic order are derived. In [28, 30], we provide two different algorithms for counting and generating all the binary n-tuples which are always more probable (less probable) than a fixed n-tuple $u \in \{0,1\}^n$. Based on these algorithms, in [31] we determine, for any fixed binary n-tuple u, all its possible "ranks" (positions) in the list of all the 2^n binary n-tuples, arranged in decreasing order of their occurrence probabilities. The usual representation for the intrinsic order, the so-called intrinsic order graph, is constructed in [29], and some order-theoretic and graph-theoretic properties of this graph are derived in [32].

All the above commented papers are mainly theoretical, and they do not contain algorithms for evaluating the unavailability of technical systems. An

algorithm for estimating the top event probability is given in [27], and it is also based on IOC. The main differences between the algorithm performed in [27] and the one proposed here are the following:
(i) The former selects a proper set of binary n-tuples, basically by taking into account the weight (number of 1-bits in the bitstring) and the lexicographic (truth-table) order between the bitstrings. The latter improves this selection strategy by using new properties of the intrinsic order instead of the lexicographic order. This leads to a more efficient algorithm, since the occurrence probabilities of the selected bitstrings are now larger, and thus the number of binary n-tuple probabilities that need to be computed is reduced.
(ii) For computing the occurrence probability of each selected binary n-tuple with weight m, the former needs to multiply n factors, while the latter only needs to multiply $m + 1$ factors. This is especially useful, for computational purposes, when the selected binary n-tuples have small weights, which is in general the case in the algorithm proposed in this paper.
(iii) For computing the maximum error in the estimation of the system unavailability (so that we can assure the required accuracy), both algorithms evaluate the total sum of the selected binary n-tuple probabilities. However, while the former uses a non-recursive formula for this purpose, the latter uses a recurrence relation, which is computationally more efficient.

Moreover, the balance between accuracy and computational cost in our new algorithm is based on the above mentioned recursive formula, which is closely related to the famous Pascal's triangle.

This paper has been organized as follows. Section 2 is devoted to set the assumptions and basic notations used in this work. In Section 3, we present all the required background about the above mentioned deterministic method for estimating the system unavailability and about the intrinsic order relation. Beginning with our new, unpublished results, in Section 4, we state some new properties of the intrinsic order. Section 5 is devoted to present a simple recurrence relation, closely related to the Pascal's formula, for computing the sums of the selected binary string probabilities. In Section 6, we present our algorithm for estimating the system unavailability, and we illustrate it with a real-life example. Finally, in Section 7 we present our conclusions.

2. Assumptions and notation

2.1. Assumptions

(i) The components and system have binary states.

(ii) The basic components are non-repairable.
(iii) The FT contains only static gates; it is not a dynamic FT.
(iv) The basic events are assumed to be mutually statistically independent.
(v) The probability of failure of each basic event is given.

In addition to the above assumptions, let us mention that the number of basic components and the number of gates of the FT to be solved may be arbitrarily large. Moreover, any event, basic or gate, may appear at multiple locations in the FT (i.e., repeated events are allowed). Finally, our method can be applied to both coherent and non-coherent FTs, and the FT logic may be arbitrarily complex.

2.2. Nomenclature and notation

n	number of basic system components
$\{0,1\}^n$	set of all the 2^n binary n-tuples
x_i	i-th basic Boolean variable of the system: $x_i = 1$ if component i fails, $x_i = 0$ otherwise
$\overline{x_i}$	negation of variable x_i
p_i	basic (failure) probability of component i: $p_i = \Pr\{x_i = 1\}$, $1 - p_i = \Pr\{x_i = 0\}$
q_i	a quotient: $q_i = p_i/(1 - p_i)$, $1 \leq i \leq n$
$u = (u_1, \ldots, u_n)$	binary n-tuple describing a system elementary state
M_v^u	the $(2 \times n)$-matrix whose first and second row are the binary n-tuples u and v, respectively
$\Pr\{u\}$	occurrence probability of system elementary state u
Φ	Boolean (structure) function describing the FT: $\Phi = 1$ if the system fails, $\Phi = 0$ otherwise
$\Pr\{\Phi = 1\}$	top event probability or system unavailability
C_1, C_0	subsets of the sets of binary n-tuples for which $\Phi = 1, 0$, respectively: $C_1 \subseteq \{u \in \{0,1\}^n \mid \Phi(u) = 1\}$ $C_0 \subseteq \{u \in \{0,1\}^n \mid \Phi(u) = 0\}$
L, U	lower and upper bounds on system unavailability
$w_H(u)$	Hamming weight of u, i.e., its number of 1-bits: $w_H(u) = \sum_{i=1}^{n} u_i$
$u_{(10}$	decimal numbering of binary n-tuple u:

	$u_{(10} = \sum_{i=1}^{n} 2^{n-i} u_i$
$\preceq$	intrinsic order relation between binary n-tuples
$\triangleright$	covering relation associated to the intrinsic order
I_n	the partially ordered set $(\{0,1\}^n, \preceq)$
S_0	occurrence probability of the binary n-tuple $(0,\ldots,0)$: $S_0 = \Pr\{(0,\ldots,0)\} = \prod_{i=1}^{n}(1-p_i)$
C_m^k	set of all binary n-tuples with weight m whose 1s are placed among the k right-most positions
S_m^k	sum of probabilities of all binary n-tuples of C_m^k
ϵ	required accuracy to estimate system unavailability
C	set of selected binary n-tuples used in the algorithm
T	total number of binary n-tuples used in the algorithm
S	sum of occurrence probabilities of all binary n-tuples used in the algorithm
$A - B$	set difference: $A - B = \{x \in A \mid x \notin B\}$
$u + C$	arithmetic sum of element u and set C: $u + C = \{u + v \mid v \in C\}$
$\lvert\cdot\rvert$	cardinality of a set

---

3. Preliminary results

As commented in Section 1, our algorithm will provide exact lower and upper bounds on the failure probability $\Pr\{\Phi = 1\}$ of an n-component system. In the next subsection, we explain how such bounds can be obtained from any arbitrary subset of the set $\{0,1\}^n$ of all system elementary states.

3.1. Bounds on system unavailability

Denoting by p_i the failure probability of the i-th system component, i.e.,

$$\Pr\{x_i = 1\} = p_i, \quad \Pr\{x_i = 0\} = 1 - p_i, \quad 1 \le i \le n,$$

then, due to the statistical independence between basic components, the occurrence probability of each system elementary state u is given by the expression

$$\Pr\{u\} = \Pr\{(u_1,\ldots,u_n)\} = \prod_{i=1}^{n} p_i^{u_i}(1-p_i)^{1-u_i} \quad \text{for all } u \in \{0,1\}^n. \quad (3.1)$$

In other words, Eq. (3.1) means that $\Pr\{\,u\}$ can be simply computed as the product of factors p_i or $1-p_i$, if $u_i = 1$ or $u_i = 0$, respectively.

Now, it is well known that the probability of a Boolean function Φ taking value 1 can be exactly computed as the sum of the occurrence probabilities of all system elementary states for which $\Phi = 1$ or, alternatively, as the remainder to 1 of the sum of the occurrence probabilities of all system elementary states for which $\Phi = 0$ [33]. That is,

$$\sum_{\substack{u\in\{0,1\}^n\\ \Phi(u)=1}} \Pr\{u\} = \Pr\{\Phi = 1\} \equiv 1 - \Pr\{\Phi = 0\} = 1 - \sum_{\substack{u\in\{0,1\}^n\\ \Phi(u)=0}} \Pr\{u\}\,. \tag{3.2}$$

For small values of the number n of basic system components, Eq. (3.2) enables one to exactly compute the failure probability, $\Pr\{\Phi = 1\}$, of the system. However, for large values of n this procedure is not feasible in practice because of the exponential nature of the problem. Just think that there are 2^n binary n-tuples of 0s and 1s. To overcome this obstacle, we can obtain lower and upper bounds L, U on the top event probability, $\Pr\{\Phi = 1\}$, as follows.

Let $C \subseteq \{0,1\}^n$ be an arbitrary subset of binary n-tuples (system elementary states), and let C_1 and C_0 be the two subsets of C for which $\Phi = 1$ and $\Phi = 0$, respectively, i.e.,

$$C_1 = \{u \in C \mid \Phi(u) = 1\}, \qquad C_0 = \{u \in C \mid \Phi(u) = 0\}.$$

Then using Eq. (3.2) we have

$$\sum_{u\in C_1} \Pr\{u\} \le \sum_{\substack{u\in\{0,1\}^n\\ \Phi(u)=1}} \Pr\{u\} = \Pr\{\Phi = 1\}, \tag{3.3}$$

$$\sum_{u\in C_0} \Pr\{u\} \le \sum_{\substack{u\in\{0,1\}^n\\ \Phi(u)=0}} \Pr\{u\} = \Pr\{\Phi = 0\} \tag{3.4}$$

and from (3.3) and (3.4) we get the following lower and upper bounds L, U

$$L = \sum_{u\in C_1} \Pr\{u\} \le \Pr\{\Phi = 1\} \equiv 1 - \Pr\{\Phi = 0\} \le 1 - \sum_{u\in C_0} \Pr\{u\} = U. \tag{3.5}$$

Note that, for a fixed set C of binary n-tuples, the above bounds L and U depend on the Boolean function Φ, because the same happens with the set partition

$$C = \{u \in C \mid \Phi(u) = 0\} \cup \{u \in C \mid \Phi(u) = 1\} = C_0 \cup C_1.$$

However, the maximum error in the estimation (difference between upper and lower bounds)

$$U - L = \left(1 - \sum_{u \in C_0} \Pr\{u\}\right) - \sum_{u \in C_1} \Pr\{u\} = 1 - \sum_{u \in C_0 \,\cup\, C_1 = C} \Pr\{u\} \qquad (3.6)$$

is completely independent of the FT logic (i.e., it is the same for any Boolean function Φ), and it only depends on the total sum, $\sum_{u \in C} \Pr\{u\}$, of the occurrence probabilities of all the selected binary n-tuples $u \in C$. More precisely, the accuracy in the estimate of the system unavailability, $\Pr\{\Phi = 1\}$, improves at the same time as the maximum error $U - L$ decreases, i.e., at the same time as this total sum, $\sum_{u \in C} \Pr\{u\}$, increases, as Eq. (3.6) has shown [27].

Consequently, the main question is how to select the minor number of system elementary states, $u \in C$, with occurrence probabilities, $\Pr\{u\}$, as large as possible, in order to minimize the computational cost –by increasing the last sum in Eq. (3.6)– when estimating the system unavailability with an acceptable accuracy, by using Eq. (3.5).

Note that the simplest answer to this question, namely ordering the 2^n binary n-tuple probabilities is not valid, because of the exponential nature of the problem and the high computational costs in sorting algorithms. To avoid this obstacle, in [25] we have established a simple, positional criterion that allows us to compare two given elementary state probabilities, $\Pr\{u\}, \Pr\{v\}$, without computing them, simply looking at the positions of the 0s and 1s in the n-tuples u, v. This criterion is explained in detail below.

3.2. The intrinsic order

In the following, we indistinctly denote the n-tuple $u \in \{0,1\}^n$ by its binary representation $(u_1, \ldots, u_n)$ or by its decimal representation, denoted here by $u_{(10}$, and we use the symbol "$\equiv$" to indicate the conversion between them, i.e.,

$$u = (u_1, \ldots, u_n) \equiv u_{(10} = \sum_{i=1}^{n} 2^{n-i} u_i,$$

e.g., for $n = 7$

$$u = (1, 0, 0, 1, 0, 1, 1) \equiv u_{(10} = 2^0 + 2^1 + 2^3 + 2^6 = 75.$$

3.2.1. The intrinsic order criterion

Given two system elementary states $u, v \in \{0,1\}^n$, the ordering between their occurrence probabilities $\Pr(u)$, $\Pr(v)$ obviously depends on the basic failure probabilities p_i, as the following simple example shows.

Example 3.1. Let $n = 3$, $u = 3 \equiv (0, 1, 1)$ and $v = 4 \equiv (1, 0, 0)$. Using Eq. (3.1), we have

$$p_1 = 0.1,\ p_2 = 0.2,\ p_3 = 0.3 : \Pr\{(0, 1, 1)\} = 0.054 < \Pr\{(1, 0, 0)\} = 0.056,$$

$$p_1 = 0.2,\ p_2 = 0.3,\ p_3 = 0.4 : \Pr\{(0, 1, 1)\} = 0.096 > \Pr\{(1, 0, 0)\} = 0.084.$$

However, under some adequate assumptions, for some pairs of binary strings the ordering between their occurrence probabilities is independent of the basic probabilities p_i, and it only depends on the relative positions of their 0s and 1s. More precisely, the following theorem [25, 27] provides us with an intrinsic, positional order criterion to compare the occurrence probabilities of two given binary n-tuples without computing them.

Theorem 3.1 (The intrinsic order theorem). *Let $n \geq 1$. Let $x_1, \dots, x_n$ be the n basic components of an n-component system. Assume that they are statistically independent and that their basic probabilities $p_i = \Pr\{x_i = 1\}$ satisfy*

$$0 < p_1 \leq p_2 \leq \cdots \leq p_n \leq \frac{1}{2}. \tag{3.7}$$

Then the probability of the n-tuple $v = (v_1, \dots, v_n) \in \{0,1\}^n$ is intrinsically less than or equal to the probability of the n-tuple $u = (u_1, \dots, u_n) \in \{0,1\}^n$ (that is, for all set of basic probabilities $\{p_i\}_{i=1}^n$ satisfying (3.7)) if and only if the matrix

$$M_v^u = \begin{pmatrix} u_1 & \dots & u_n \\ v_1 & \dots & v_n \end{pmatrix}$$

either has no $\binom{1}{0}$ column, or for each $\binom{1}{0}$ column in M_v^u there exists (at least) one corresponding preceding $\binom{0}{1}$ column (IOC).

Remark 3.1. In the following, we assume that the basic probabilities p_i always satisfy condition (3.7). Note that this hypothesis is not restrictive for practical applications because, if for some $i : p_i > \frac{1}{2}$, then we only need to consider the variable $\overline{x_i} = 1 - x_i$, instead of x_i. Next, we order the n new Boolean variables by increasing order of their probabilities.

Remark 3.2. The main computational advantage of Theorem 3.1 is that, instead of computing and ordering the 2^n binary n-tuple probabilities $\Pr\{u\}$, we only need to order the n basic probabilities p_i, as shown in Eq. (3.7). In other words, IOC reduces the complexity of the problem from exponential to linear!

Remark 3.3. The $\binom{0}{1}$ column preceding to each $\binom{1}{0}$ column is not required to be necessarily placed at the immediately previous position, but just at previous position.

Remark 3.4. The term *corresponding*, used in Theorem 3.1, has the following meaning: For each two $\binom{1}{0}$ columns in matrix M_v^u, there must exist (at least) two *different* $\binom{0}{1}$ columns preceding to each other. In other words: For each $\binom{1}{0}$ column in matrix M_v^u, the number of preceding $\binom{0}{1}$ columns must be strictly greater than the number of preceding $\binom{1}{0}$ columns.

The matrix condition IOC, stated by Theorem 3.1, is called the *intrinsic order criterion*, because it is independent of the basic probabilities p_i and it only (i.e., *intrinsically*) depends on the relative positions of the 0s and 1s in the binary strings u and v. Theorem 3.1 naturally leads to the following partial order relation on the set $\{0,1\}^n$ [25, 26]. The so-called intrinsic order will be denoted by "$\preceq$", and when $v \preceq u$ we say that v *is intrinsically less than or equal to* u.

Definition 3.1. For all $u, v \in \{0,1\}^n$

$$v \preceq u \ \text{ iff } \ \Pr\{v\} \leq \Pr\{u\} \ \text{ for all set of basic probabilities } \{p_i\}_{i=1}^n \text{ s.t. (3.7)}$$

$$\text{iff matrix } \ M_v^u \text{ satisfies IOC.}$$

Throughout this paper, the partially ordered set (poset, for short) for n variables $(\{0,1\}^n, \preceq)$ will be denoted by I_n; see [34] for more details about posets.

Example 3.2. For $n = 3$: $3 \equiv (0,1,1) \npreceq 4 \equiv (1,0,0)$ and also, we have that $4 \equiv (1,0,0) \npreceq 3 \equiv (0,1,1)$ because

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

do not satisfy IOC (Remark 3.4). Therefore, $(0,1,1)$ and $(1,0,0)$ are incomparable by intrinsic order, i.e., the ordering between $\Pr\{(0,1,1)\}$ and $\Pr\{(1,0,0)\}$ depends on the basic probabilities p_i, as Example 3.1 has shown.

Example 3.3. For $n = 4$: $12 \equiv (1,1,0,0) \preceq 3 \equiv (0,0,1,1)$ because

$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

satisfies IOC (Remark 3.3). Therefore, for all $0 < p_1 \leq p_2 \leq p_3 \leq p_4 \leq \frac{1}{2}$

$$\Pr\{(1,1,0,0)\} \leq \Pr\{(0,0,1,1)\}.$$

Many different properties of the intrinsic order can be immediately derived from its simple matrix description IOC. We refer the reader to [25, 26, 27, 28, 29, 30, 31, 32] for both theoretical consequences and practical applications of IOC.

3.2.2. The intrinsic order graph

To finish this section, we present the graphical representation of the poset $I_n = (\{0,1\}^n, \preceq)$. The usual representation of a poset is its Hasse diagram (see [34] for more details about these diagrams). Specifically, for our poset I_n, its Hasse diagram is a directed graph (digraph, for short) whose vertices are the 2^n binary n-tuples of 0s and 1s, and whose edges go upward from v to u whenever u covers v, denoted by $u \rhd v$. This means that u is intrinsically greater than v and there are no other elements between them, i.e.,

$$u \rhd v \quad \Leftrightarrow \quad u \succ v \text{ and there is no } w \in \{0,1\}^n \text{ s.t. } u \succ w \succ v.$$

The Hasse diagram of the poset I_n will be also called the *intrinsic order graph* for n variables. For small values of n, the Hasse diagram of I_n can be constructed by direct application of IOC. For instance, the Hasse diagram of $I_1 = (\{0,1\}, \preceq)$ is given in Fig. 1.

Indeed, note that I_1 has a downward edge from 0 to 1 because $0 \succ 1$, since matrix $\binom{0}{1}$ satisfies IOC (it has no $\binom{1}{0}$ columns; see Theorem 3.1). This is in accordance with the obvious fact that

$$\Pr\{0\} = 1 - p_1 \geq p_1 = \Pr\{1\}\,, \text{ since } \; p_1 \leq 1/2 \text{ due to Eq. (3.7).}$$

$$\begin{tikzcd} 0 \arrow[d, dash] \\ 1 \end{tikzcd}$$

Fig. 1. The intrinsic order graph for $n = 1$ using decimal representation.

However, for large values of n we need a more efficient method. For this purpose, in [29] we have developed an algorithm for iteratively building up, for all $n \geq 2$, the digraph of I_n from the digraph of I_1 (depicted in Fig. 1). The next theorem states this algorithm, using the decimal representation $u_{(10}$ of the binary strings $u = (u_1, \ldots, u_n) \in \{0,1\}^n$. See [29] for the proof and for additional properties of the intrinsic order graph.

Theorem 3.2 (Building up I_n from I_1). *For all $n > 1$, the digraph of $I_n = \{0, \ldots, 2^n - 1\}$ can be drawn simply by adding to the digraph of $I_{n-1} = \{0, \ldots, 2^{n-1} - 1\}$ its isomorphic copy $2^{n-1} + I_{n-1} = \{2^{n-1}, \ldots, 2^n - 1\}$. This addition must be performed placing the powers of 2 at consecutive levels of the Hasse diagram of I_n. Finally, the edges connecting one vertex u of I_{n-1} with the other vertex v of $2^{n-1} + I_{n-1}$ are given by the set of vertex pairs*

$$\left\{(u,v) \equiv \left(u_{(10}, 2^{n-2} + u_{(10}\right) \;\middle|\; 2^{n-2} \leq u_{(10} \leq 2^{n-1} - 1\right\}.$$

Basically, Theorem 3.2 affirms that to construct I_n, we first add to I_{n-1} its isomorphic copy $2^{n-1} + I_{n-1}$, and then we connect one-to-one the nodes of "the second half of the first half" to the nodes of "the first half of the second half": A nice fractal property of I_n!

In Fig. 2, the algorithm described by Theorem 3.2 is illustrated with the intrinsic order graph for $n = 1, 2, 3, 4$ using the decimal numbering instead of the binary representation of their 2^n nodes, for a more comfortable and simpler notation. An example of intrinsic order graph using the binary representation of its nodes, namely the digraph of I_4, is depicted in Fig. 3, in the next section.

The intrinsic order graph of I_n displays all the binary n-tuples (i.e., its 2^n nodes) from top to bottom in decreasing order of their occurrence probabilities, as explained below.

Each pair (u, v) of vertices connected in the digraph of I_n either by one edge or by a longer descending path (consisting of more than one edge) from u to v, means that u is intrinsically greater than v, i.e., $u \succ v$. For instance, looking at the Hasse diagram of I_4, the right-most one in Fig. 2, we observe that $3 \equiv (0, 0, 1, 1) \succ 12 \equiv (1, 1, 0, 0)$, in accordance with Example 3.3.

On the contrary, each pair (u, v) of non-connected vertices in the digraph of I_n either by one edge or by a longer descending path, means that u and v are incomparable by intrinsic order, i.e., $u \nsucc v$ and $v \nsucc u$. For instance, looking at the Hasse diagram of I_3, the third one from left to right in Fig. 2, we observe that $3 \equiv (0, 1, 1)$ and $4 \equiv (1, 0, 0)$ are incomparable by intrinsic order, in accordance with Examples 3.1 and 3.2.

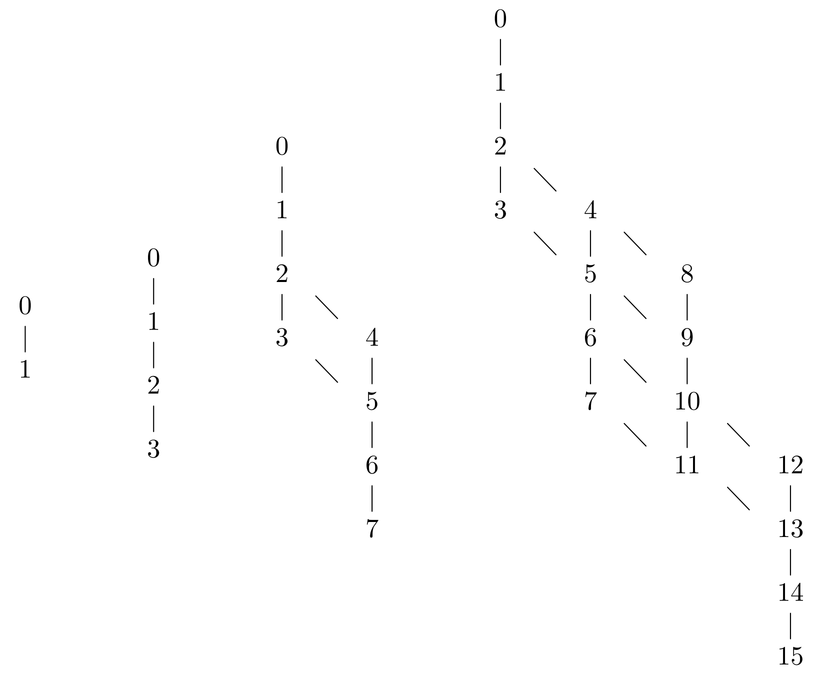

Fig. 2. The intrinsic order graph for $n = 1, 2, 3, 4$ using decimal representation.

4. New properties of the intrinsic order

Once we have presented, in Section 3, all the required background for making this paper self-contained, from now on the rest of the paper is devoted to our new results.

The following theorem states new properties of the intrinsic order relation, that will be used in the proposed algorithm. First, we need to set the following notation.

For every binary n-tuple $u = (u_1, \ldots, u_n)$, the Hamming weight –or simply the weight– of u, i.e., the number of 1-bits in u, will be denoted by

$$w_H(u) = \sum_{i=1}^{n} u_i.$$

The occurrence probability of the zero n-tuple will be denoted by S_0, i.e., according to Eq. (3.1),

$$S_0 = \Pr\left\{\left(0, \overbrace{\ldots}^{n}, 0\right)\right\} = \prod_{i=1}^{n} (1 - p_i) \,. \tag{4.1}$$

For the nonzero n-tuples –that is, for those with weight greater than or equal to 1–, we shall denote by C_m^k (S_m^k, respectively) the set (the sum of the probabilities, respectively) of the binary n-tuples with weight m whose 1s are placed among the k right-most positions ($1 \leq m \leq k \leq n$), i.e.,

$$C_m^k = \left\{u \in \{0,1\}^n \mid w_H(u) = m,\ u_i = 1 \Rightarrow i \in \{n-k+1, \ldots, n\}\right\},$$

$$S_m^k = \sum_{u \in C_m^k} \Pr\{u\} \,.$$

Note that, for the extreme cases $k = m$ and $k = n$, we have, on one hand

$$C_m^m = \left\{\left(0, \overbrace{\ldots}^{n-m}, 0, 1, \overbrace{\ldots}^{m}, 1\right)\right\}, \quad S_m^m = \Pr\left\{\left(0, \overbrace{\ldots}^{n-m}, 0, 1, \overbrace{\ldots}^{m}, 1\right)\right\}$$

and, on the other hand

$$C_m^n = \left\{u \in \{0,1\}^n \mid w_H(u) = m\right\}, \quad S_m^n = \sum_{u \in \{0,1\}^n,\, w_H(u) = m} \Pr\{u\} \,, \tag{4.2}$$

that is, C_m^n (S_m^n, respectively) is simply the set (the sum of the probabilities, respectively) of all binary n-tuples with Hamming weight m.

Obviously, if u is a binary n-tuple with weight m whose 1s are placed among the $k-1$ right-most positions, then u is also a binary n-tuple with weight m whose 1s are placed among the k right-most positions, i.e.,

$$C_m^m \subset C_m^{m+1} \subset \cdots \subset C_m^{k-1} \subset C_m^k \subset \cdots \subset C_m^{n-1} \subset C_m^n,$$

and thus

$$S_m^m < S_m^{m+1} < \cdots < S_m^{k-1} < S_m^k < \cdots < S_m^{n-1} < S_m^n.$$

Theorem 4.1. *Let $n \geq 1$. Then*
(i) For every binary n-tuple v with weight $m > 0$, there exists, at least, one binary n-tuple u with weight $m-1$ such that $u \succ v$.
(ii) For every binary n-tuple $v \in C_m^k - C_m^{k-1}$, such that $0 < m < k \leq n$, there exists, at least, one binary n-tuple $u \in C_m^{k-1}$ such that $u \succ v$.

Proof. We give a constructive proof explicitly defining the binary strings u.
(i) Let v be a binary n-tuple with weight $m > 0$. Choose any index $j \in \{1, \ldots, n\}$ such that $v_j = 1$. Define u as follows

$$u_i = \begin{cases} 0 & \text{if} \quad i = j, \\ v_i & \text{if} \quad i \neq j. \end{cases}$$

In this way, $w_H(u) = w_H(v) - 1 = m - 1$. Moreover, matrix M_v^u has exactly one $\binom{0}{1}$ column (its j-th one), while its remaining $n-1$ columns are either $\binom{0}{0}$ or $\binom{1}{1}$. Hence, M_v^u has no $\binom{1}{0}$ columns and then, by Theorem 3.1, it satisfies IOC. Therefore, by Definition 3.1, $u \succ v$.
(ii) Let $v \in C_m^k - C_m^{k-1}$ $(m < k \leq n)$. Then v is a binary n-tuple with weight m whose 1s are placed among the k right-most positions, but not among the $k-1$ right-most positions. Hence, since v_{n-k+1} is the bit placed at the k-th position from right to left in the n-tuple v, we have

$$v_{n-k+1} = 1 \quad \text{and} \quad \exists\, j > n-k+1 \text{ s.t. } v_j = 0.$$

Define u as follows

$$u_i = \begin{cases} 0 & \text{if} \quad i = n-k+1, \\ 1 & \text{if} \quad i = j, \\ v_i & \text{if} \quad i \neq n-k+1, j. \end{cases}$$

In this way, $w_H(u) = w_H(v) = m$, and $u \in C_m^{k-1}$. Moreover, matrix M_v^u has exactly one $\binom{1}{0}$ column (its j-th one) and exactly one $\binom{0}{1}$ column (its $(n-k+1)$-th one), while its remaining $n-2$ columns are either $\binom{0}{0}$ or $\binom{1}{1}$. Hence, the only $\binom{1}{0}$ column of M_v^u is preceded by a corresponding $\binom{0}{1}$ column (since $n-k+1 < j$), and thus, by Theorem 3.1, M_v^u satisfies IOC. Therefore, by Definition 3.1, $u \succ v$. $\square$

Theorem 4.1 and its proof can be illustrated by Fig. 3, where the digraph of I_4 (the right-most one in Fig. 2) is depicted using now the binary representation of their nodes, instead of their decimal equivalents. For instance, consider the binary 4-tuple $v = (1,1,0,0) \in \{0,1\}^4$. On one hand, since $m = w_H(v) = 2$ then, according to Theorem 4.1-(i), we can choose, e.g., $u = (1,0,0,0)$, so that $w_H(u) = 1$ and $u \succ v$. On the other hand, since $v = (1,1,0,0) \in C_2^4 - C_2^3$ then, according to Theorem 4.1-(ii), we can choose, e.g., $u = (0,1,1,0)$, so that $w_H(u) = 2$, $u \in C_2^3$ and $u \succ v$.

$$\begin{tikzcd}
(0,0,0,0) \arrow[d, dash] & & & \\
(0,0,0,1) \arrow[d, dash] & & & \\
(0,0,1,0) \arrow[d, dash] \arrow[dr, dash] & & & \\
(0,0,1,1) \arrow[dr, dash] & (0,1,0,0) \arrow[d, dash] \arrow[dr, dash] & & \\
 & (0,1,0,1) \arrow[d, dash] \arrow[dr, dash] & (1,0,0,0) \arrow[d, dash] & \\
 & (0,1,1,0) \arrow[d, dash] \arrow[dr, dash] & (1,0,0,1) \arrow[d, dash] & \\
 & (0,1,1,1) \arrow[dr, dash] & (1,0,1,0) \arrow[d, dash] \arrow[dr, dash] & \\
 & & (1,0,1,1) \arrow[dr, dash] & (1,1,0,0) \arrow[d, dash] \\
 & & & (1,1,0,1) \arrow[d, dash] \\
 & & & (1,1,1,0) \arrow[d, dash] \\
 & & & (1,1,1,1)
\end{tikzcd}$$

Fig. 3. The intrinsic order graph for $n = 4$ using binary representation.

5. The sums S_m^k and the Pascal's triangle

In this section, we present a simple recurrence relation, closely related to the Pascal's triangle, for rapidly computing the sums S_m^k defined in the previous section. This relation is given in the next theorem. Due to its recursiveness, our formula is very adequate for computational purposes and, indeed, it is the main result underlying our algorithm for estimating the system unavailability –presented in the next section.

First, we associate to each one of the basic probabilities p_i $(1 \le i \le n)$, the corresponding quotient

$$q_i = \frac{p_i}{1-p_i} \quad \text{for all } i = 1, 2, \dots, n.$$

Note that, due to our hypothesis (3.7) and to the increasing character of function $y = \frac{x}{1-x}$, we have

$$0 < q_1 \le q_2 \le \cdots \le q_n \le 1.$$

Remark 5.1. We must highlight here that the above quotients q_i can be used to reduce the computational cost, when computing the occurrence probabilities of the binary n-tuples $u = (u_1, \dots, u_n)$. Instead of using Eq. (3.1), taking advantage of the occurrence probability S_0 of the zero n-tuple (computed by Eq. (4.1)), we have

$$\begin{aligned} \Pr\{u\} = \Pr\{(u_1,\dots,u_n)\} &= \prod_{i=1}^{n} p_i^{u_i}(1-p_i)^{1-u_i} = \prod_{\substack{i=1\\u_i=1}}^{n} p_i \prod_{\substack{i=1\\u_i=0}}^{n} (1-p_i) \\ &= \prod_{\substack{i=1\\u_i=1}}^{n} \frac{p_i}{1-p_i} \prod_{i=1}^{n} (1-p_i) = \Pr\left\{\left(0, \overbrace{\dots}^{n}, 0\right)\right\} \prod_{\substack{i=1\\u_i=1}}^{n} q_i = S_0 \prod_{\substack{i=1\\u_i=1}}^{n} q_i, \end{aligned}$$

so that

$$\Pr\{u\} = \Pr\{(u_1,\dots,u_n)\} = S_0 \prod_{\substack{i=1\\u_i=1}}^{n} q_i. \tag{5.1}$$

The above strategy for deriving Eq. (5.1) (which computes the binary string probabilities) has also been used in [27] for a different purpose, namely for obtaining an explicit, non-recursive formula to compute the maximum error in the estimation of the system unavailability.

We stress out the fact that Eq. (5.1) is especially useful for reducing the computational cost, when the Hamming weight of u (i.e., the number of bits $u_i = 1$) is small, which is in general the case for the bitstrings used in the algorithm that we propose in the next section. Indeed, suppose that we want to compute the occurrence probability of a binary n-tuple $u = (u_1, \ldots, u_n)$ with a small weight m. If we use Eq. (3.1), we need to multiply the n factors

$$p_i^{u_i} (1 - p_i)^{1-u_i} = \begin{cases} \quad p_i & \text{if} \quad u_i = 1, \\ 1 - p_i & \text{if} \quad u_i = 0 \end{cases} \qquad (1 \le i \le n) .$$

Otherwise, using Eq. (5.1), we only need to multiply the $m + 1$ factors $S_0, q_{i_1}, \ldots, q_{i_m}$, where $i_1, \ldots, i_m$ are the positions of the m 1-bits in u. For instance, for

$$n = 9, \quad u = (0, 0, 1, 0, 0, 0, 0, 1, 1) , \quad w_H (u) = 3,$$

using Eq. (3.1), we get

$$\Pr \{u\} = (1 - p_1) (1 - p_2) \, p_3 \, (1 - p_4) (1 - p_5) (1 - p_6) (1 - p_7) \; p_8 \, p_9,$$

while, using Eq. (5.1), we get

$$\Pr \{u\} = S_0 \; q_3 \; q_8 \; q_9.$$

Theorem 5.1. *For all $n \ge 1$ and for all m, k such that $1 \le m \le k \le n$*

$$S_m^k = S_m^{k-1} + q_{n-k+1} S_{m-1}^{k-1}, \tag{5.2}$$

where we adopt the conventions that $S_0^k = S_0$ for all $k = 0, 1, \ldots, n - 1$ and $S_m^{m-1} = 0$ for all $m = 1, 2, \ldots, n$.

Proof. First, note that, according to the definition of the sums S_m^k, the expressions S_0^k $(0 \le k \le n - 1)$ and S_m^{m-1} $(1 \le m \le n)$, to which the above conventions are referred, do not make sense. We distinguish the following two cases.
(a) For $k = m$, we have

$$\begin{aligned} S_m^m &= \Pr \left\{ \left(0, \overbrace{\cdots}^{n-m}, 0, 1, \overbrace{\cdots}^{m}, 1 \right) \right\} \\ &= \frac{p_{n-m+1}}{1 - p_{n-m+1}} \Pr \left\{ \left(0, \overbrace{\cdots}^{n-m+1}, 0, 1, \overbrace{\cdots}^{m-1}, 1 \right) \right\} = q_{n-m+1} S_{m-1}^{m-1} \end{aligned}$$

and this is exactly Eq. (5.2) for $k = m$, since we have adopted the convention that $S_m^{m-1} = 0$ $(1 \leq m \leq n)$.
(b) For $k > m$, we give a combinatorial proof. The set C_m^k of the binary n-tuples $(u_1, \dots, u_n)$ with weight m whose 1s are placed among the k right-most positions, namely $n-k+1, \dots, n$, can be partitioned into the following two subsets:

(b.1) The subset of n-tuples $u \in C_m^k$ such that $u_{n-k+1} = 0$.
(b.2) The subset of n-tuples $u \in C_m^k$ such that $u_{n-k+1} = 1$.

The first subset is exactly the set of binary n-tuples with weight m whose m 1s are placed among the $k-1$ right-most positions, i.e., the set C_m^{k-1}. The second subset is exactly the set of binary n-tuples with weight m such that $u_{n-k+1} = 1$ and whose remaining $m-1$ 1-bits are placed among the $k-1$ right-most positions, i.e., the set

$$\left(0, \overset{n-k}{\overbrace{\cdots}}, 0, 1, 0, \overset{k-1}{\overbrace{\cdots}}, 0\right) + C_{m-1}^{k-1}$$
$$= \left\{ \left(0, \overset{n-k}{\overbrace{\cdots}}, 0, 1, 0, \overset{k-1}{\overbrace{\cdots}}, 0\right) + v \;\middle|\; v \in C_{m-1}^{k-1} \right\}.$$

So, we have the set partition

$$C_m^k = C_m^{k-1} \cup \left[\left(0, \overset{n-k}{\overbrace{\cdots}}, 0, 1, 0, \overset{k-1}{\overbrace{\cdots}}, 0\right) + C_{m-1}^{k-1} \right] \tag{5.3}$$

and thus we get
$$S_m^k = S_m^{k-1} + q_{n-k+1} S_{m-1}^{k-1},$$
since S_m^k, S_m^{k-1} and $q_{n-k+1} S_{m-1}^{k-1}$ are obviously the sums of the occurrence probabilities of all the n-tuples belonging to the sets

$$C_m^k, \qquad C_m^{k-1} \qquad \text{and} \qquad \left(0, \overset{n-k}{\overbrace{\cdots}}, 0, 1, 0, \overset{k-1}{\overbrace{\cdots}}, 0\right) + C_{m-1}^{k-1},$$

respectively. □

The above combinatorial proof suggests a tight connection between our model and the famous Pascal's triangle. Indeed, note that each sum S_m^k is taken over the corresponding set C_m^k, whose cardinality is the combinatorial number $\binom{k}{m}$, since m 1s must be placed among k positions. That is,

$$S_m^k = \sum_{u \in C_m^k} \Pr\{u\} \quad \text{and} \quad \left|C_m^k\right| = \binom{k}{m}, \quad 1 \leq m \leq k \leq n. \tag{5.4}$$

In Fig. 4, we illustrate this connection. The left triangle contains the sums S_m^k $(1 \leq m \leq k \leq 4)$ of system elementary state probabilities. The right triangle contains the binomial coefficients $\binom{k}{m}$ $(1 \leq m \leq k \leq 4)$, that is, it is the Pascal's triangle where its left-most diagonal $\left\{\binom{k}{0}\right\}_{k \geq 0}$ has been deleted in order to avoid the repetition of the zero n-tuple probability S_0. Clearly, Eq. (5.4) shows that each one of the sums S_m^k is taken over exactly $\binom{k}{m}$ binary strings. The interpretation of Fig. 4, for the relationship between accuracy and computational cost, is the following. The error estimate of the system unavailability, $\Pr\{\Phi = 1\}$, is reduced by the quantity S_m^k at the same time as the Boolean function Φ, describing the FT logic, needs to be evaluated at $\binom{k}{m}$ binary strings.

$$\begin{array}{ccccccc}
 & & & S_1^1 & & & \\
 & & S_1^2 & & S_2^2 & & \\
 & S_1^3 & & S_2^3 & & S_3^3 & \\
S_1^4 & & S_2^4 & & S_3^4 & & S_4^4
\end{array}
\qquad
\begin{array}{ccccccc}
 & & & \binom{1}{1} & & & \\
 & & \binom{2}{1} & & \binom{2}{2} & & \\
 & \binom{3}{1} & & \binom{3}{2} & & \binom{3}{3} & \\
\binom{4}{1} & & \binom{4}{2} & & \binom{4}{3} & & \binom{4}{4}
\end{array}$$

Fig. 4. Accuracy/computational cost ratio and the Pascal's triangle.

Moreover, for the case $m < k$, counting the binary n-tuples involved in each of the three sums of formula (5.2) or, equivalently, counting the binary n-tuples belonging to each of the three sets of partition (5.3), we get

$$\left|C_m^k\right| = \left|C_m^{k-1}\right| + \left|C_{m-1}^{k-1}\right|,$$

and then, using Eq. (5.4), we obtain the famous Pascal's formula

$$\binom{k}{m} = \binom{k-1}{m} + \binom{k-1}{m-1}.$$

6. The algorithm

Based on the previous ideas and propositions, in this section we present a new algorithm for evaluating the system unavailability. We call it the

FTA via Intrinsic Ordering & Pascal's Triangle Algorithm (*FTAIOPT* - Algorithm, for short) because the FTA of the current system is based on an adequate selection of binary n-tuples (i.e., with large occurrence probabilities) tightly connected to both the intrinsic ordering and the Pascal's triangle. Next, we illustrate our approach on a real-world analysis problem taken from [35].

6.1. The FTAIOPT-Algorithm

For each given maximum admissible error ε, the FTAIOPT-Algorithm provides lower and upper bounds on the failure probability $\Pr\{\Phi = 1\}$ of a (coherent or non-coherent) system depending on n mutually independent basic components. The set $\{p_i\}_{i=1}^{n}$ of basic probabilities must satisfy the (non-restrictive) hypothesis (3.7). If this assumption is not satisfied by the current system, then we proceed as explained in Remark 3.1 and next, we rewrite the Boolean function Φ using the new Boolean variables.

The following are the main ideas of the FTAIOPT-Algorithm:
(i) Theorem 4.1-(i) has stated that for every nonzero binary n-tuple v of weight m we can always find a binary n-tuple u of weight $m-1$, s.t. $\Pr\{u\} > \Pr\{v\}$. Theorem 4.1-(ii) has stated that for every nonzero binary n-tuple v of weight m whose 1s are placed among the k right-most positions, but not among the $k-1$ right-most positions, we can always find a binary n-tuple u of weight m whose 1s are placed among the $k-1$ right-most positions, s.t. $\Pr\{u\} > \Pr\{v\}$. Hence, we select the sets C_m^k of binary n-tuples in increasing order of the weight m (*for* $m = 1$ *to* n), and for each fixed weight m, in increasing order of k (*for* $k = m$ *to* n).
(ii) To compute the sums S_m^k, we use the recurrence formula (5.2) stated by Theorem 5.1.
(iii) In order to get the required accuracy (maximum admissible error ε), we select the binary n-tuples until we assure that the sum of their occurrence probabilities is greater than or equal to $1-\varepsilon$. In this way, according to Eq. (3.6), we have

$$\sum_{u\in C} \Pr\{u\} \geq 1-\varepsilon \Leftrightarrow U - L = 1 - \sum_{u\in C} \Pr\{u\} \leq \varepsilon. \tag{6.1}$$

(iv) For each one of the selected binary n-tuples, $u \in C$, we compute its occurrence probability, $\Pr\{u\}$, by Eq. (3.1) and evaluate the Boolean function Φ. According to Eq. (3.5), we get the lower and upper bounds on system unavailability

$$L = \sum_{u \in C,\ \Phi(u)=1} \Pr\{u\} \leq \Pr\{\Phi = 1\} \leq 1 - \sum_{u \in C,\ \Phi(u)=0} \Pr\{u\} = U,$$

satisfying, as desired,

$$U - L \leq \varepsilon.$$

Based on these ideas, our algorithm can be described as follows.
The FTAIOPT-Algorithm

```
Step 1. For $i = 1$ to $n$ compute $q_i = \frac{p_i}{1-p_i}$.
Step 2. Compute $S_0 = \Pr\left\{\left(0, \overbrace{\cdots}^{n}, 0\right)\right\}$ (using Eq. (4.1)).
Step 3. If $S_0 \geq 1 - \epsilon$ (the zero $n$-tuple is enough!) then goto Step 9 (due to Eq. (6.1))
Step 4. For $k = 0, 1, \ldots, n-1$ : define $S_0^k = S_0$ (by convention, Theorem 5.1).
Step 5. For $m = 1, 2, \ldots, n$ :    define $S_m^{m-1} = 0$ (by convention, Theorem 5.1).
Step 6. Call $S = S_0$.
Step 7. For $m = 1$ to $n$ (due to Theorem 4.1)
            For $k = m$ to $n$ (due to Theorem 4.1)
                compute $S_m^k = S_m^{k-1} + q_{n-k+1} S_{m-1}^{k-1}$ (by Theorem 5.1)
                if $S + S_m^k \geq 1 - \epsilon$ then goto Step 8 (due to Eq. (6.1))
            Next $k$
        Compute $S = S + S_m^n$ ($S$ is the sum of the probabilities of all
        $n$-tuples with weights $0, \ldots, m$, due to Eq. (4.2))
    Next $m$
Step 8. For all nonzero selected $n$-tuples $u \in C_1^n \cup \cdots \cup C_{m-1}^n \cup C_m^k$ compute $\Pr\{u\}$ (using Eq. (5.1)).
Step 9. For all nonzero selected $n$-tuples $u \in C_1^n \cup \cdots \cup C_{m-1}^n \cup C_m^k$ and for the zero $n$-tuple do:
            If $\Phi(u) = 1$ then $L = L + \Pr\{u\}$,
            If $\Phi(u) = 0$ then $V = V + \Pr\{u\}$.
Step 10. Compute $U = 1 - V$ and write (due to Eq. (3.5))
            $L \leq \Pr\{\Phi = 1\} \leq U$.
```

6.2. A real-world example

Our real-life example corresponds to the accumulator system (ACC) of a pressured water reactor (PWR) in a nuclear power plant, which has been taken from [35]. The ACC is a passive injection system that is a part of the PWR emergency core coolant injection system, where one accumulator is attached to the cold leg of each loop of the reactor coolant system to provide a source of emergency make-up water for that loop if a loss of coolant accident (LOCA) occurs. The accumulator is simply a pressure vessel partially filled with borated water, and pressurized with nitrogen gas. When pressure in the cold leg drops below 650 psig, the check valves open, and borated water is forced into the reactor coolant system. The PWR has 3 loops, hence there are three accumulators, one attached to cold leg of each loop. The contents of 2 accumulators are required to successfully reflood the core following a large LOCA. If a LOCA occurs in a cold leg, the content of the corresponding accumulator is lost through the break; thus both of the remaining 2 accumulators must successfully discharge their contents into the reactor coolant system to protect the core. Therefore, success for the accumulator discharge to the reactor coolant system for the postulated cold leg LOCA requires both accumulators. The total CPU time required to solve this numerical example was less than one second, using a 2GHz Pentium IV.

The ACC failure behavior is represented by a coherent fault tree with 83 basic components and 42 logic gates. For a maximum admissible error $\epsilon = 10^{-6}$, it was enough to use the zero 83-tuple (with weight $m = 0$); all the 83 binary 83-tuples of weight $m = 1$; and the $\binom{34}{2} = 561$ binary 83-tuples of weight $m = 2$, whose 1s are placed among the $k = 34$ right-most positions. That is, the set of selected binary 83-tuples was

$$C = \{0\} \cup C_1^{83} \cup C_2^{34}.$$

Then, using a total number of (see Eq. (5.4) and Fig. 4)

$$T = |\{0\}| + \left|C_1^{83}\right| + \left|C_2^{34}\right| = \binom{83}{0} + \binom{83}{1} + \binom{34}{2} = 645$$

binary 83-tuples, the sum of the occurrence probabilities of all selected bitstrings was (see Eqs. (4.1) & (5.4))

$$\sum_{u \in C} \Pr\{u\} = S = S_0 + S_1^{83} + S_2^{34} = 9999.99065 \cdot 10^{-4} \geq 9999.99 \cdot 10^{-4} = 1 - \epsilon.$$

In this way, we have obtained the following lower and upper bounds on the ACC unavailability

$$L = 3.74552 \cdot 10^{-4} \leq \Pr\{\Phi = 1\} \leq 3.75487 \cdot 10^{-4} = U,$$

satisfying, as desired (see Eq. (6.1) and Step 7),

$$U - L = 9.35 \cdot 10^{-7} < 10^{-6} = \epsilon.$$

Note that Steps 1-7 of the algorithm have determined that the above binary 83-tuples are enough for estimating the ACC unavailability with the required accuracy $\epsilon = 10^{-6}$. These steps only involve the basic probabilities p_i and the required accuracy ϵ. Thus, the same 645 binary strings are also enough to estimate the top event probability of any other coherent or non-coherent 83-system with the same set $\{p_i\}_{i=1}^{83}$ of basic probabilities as the ACC system, and for the same maximum admissible error $\epsilon = 10^{-6}$.

Next, in Step 8 the occurrence probabilities of these 645 selected binary strings are computed. Of course, due to the way in which Step 7 selects the binary n-tuples (in increasing order of their Hamming weights), these calculations can be carried out, with a very low computational cost (taking advantage of the value S_0 computed in Step 2), simply by using Eq. (5.1). See Remark 5.1 and Step 8.

Note that the Boolean structure (function) of the system has not yet used, i.e., the results obtained by Steps 1-8 are valid with independence of the FT logic! In particular, it is also irrelevant whether the current system is coherent or non-coherent.

Then, for each given structure function Φ, we only need to evaluate the current Boolean function Φ on the 645 selected binary 83-tuples for obtaining the corresponding exact lower and upper bounds L, U for each given maximum admissible error (Steps 9-10).

Under quite general hypotheses, our algorithm provides exact lower and upper bounds on system unavailability for a given accuracy. The main characteristics and advantages of the proposed method are the following:
(i) Our algorithm, based on Eq. (3.5) and on the IOC, does not require any knowledge or information (that other techniques often require) about the FT structure function. This is the main advantage, compared with many other methods. (ii) Because of Eqs. (3.5) and (3.6), the accuracy in the estimation is independent of the FT logic, and it only depends on the sum of the occurrence probabilities of the selected binary n-tuples. (iii) Due to Eqs. (3.5)

and (6.1), one can assure a priori (i.e., with no need to use the Boolean structure function) the required accuracy, simply by computing the sum of the selected binary string probabilities. (iv) For this reason, the set of bitstrings that the algorithm has selected (using the IOC) for evaluating a given FT with a maximum admissible error, would be also valid for evaluating, with the same accuracy, any other FT with the same number of basic events and with the same basic probabilities. That is, only the lower and upper bounds depend on the FT logic, but not its difference (accuracy). (v) As explained in Section 5, the balance between accuracy and computational cost is closely related to the Pascal's formula.

Finally, we would like to highlight, with an unquestionable argument, the relevance (in the context of our model) of the intrinsic ordering for automatically selecting binary n-tuples with large occurrence probabilities. For large values of the number n of basic components (say, for instance, $n = 203$, a reasonable number in practice) computing and sorting all the 2^n binary n-tuples in decreasing order of their occurrence probabilities would be not only computationally expensive, but indeed physically impossible! Just think that nowadays physicists estimate that the age of the Universe (from the Big-Bang to present) is roughly 2^{203} Planck times. Recall that the Planck time is the smallest possible measurement of time that has any physical meaning (1 Planck time $\cong 5.391 \cdot 10^{-44}$ seconds; 1 second $\cong 1.855 \cdot 10^{43}$ Planck times).

7. Conclusion

In this paper, we have presented a new, easily implementable algorithm for obtaining exact lower and upper bounds on the system unavailability of complex technical systems depending on an arbitrarily large number n of statistically independent basic components. Our algorithm does not require any previous qualitative analysis of the FT, its efficiency is independent of the complexity of the FT logic, and it is based on the adequate selection of system elementary states with large occurrence probabilities.

For selecting such system elementary states, we have derived new theoretical properties of the intrinsic ordering based on IOC: A matrix, positional criterion that allows us to automatically compare binary n-tuple probabilities without computing them, simply looking at the relative positions of their 0s and 1s.

For assuring a priori the required accuracy in the estimation, we have presented a recurrence relation, tightly related to the famous Pascal's formula, for rapidly computing the sum S_m^k of the occurrence probabilities of all binary n-tuples with weight m whose 1s are placed among the k rightmost positions. The accuracy improves by the quantity S_m^k at the same time as the computational cost increases by the number $\binom{k}{m}$ of bitstrings involved in that sum, and used in the algorithm.

Finally, for future research one can think on improving the efficiency of the proposed algorithm by finding alternative ways for automatically selecting system elementary states with large occurrence probabilities. Such new selection strategies would be based on new theoretical properties of the intrinsic order relation, obtained from IOC.

Acknowledgements

The author thanks the anonymous referees for their valuable comments and suggestions, which have improved the earlier version of this paper. This work was partially supported by the "Ministerio de Economía y Competitividad" (Spanish Government), and FEDER; Grant contract: CGL2011-29396-C03-01.

References

[1] N.G. Leveson, Safeware: System Safety and Computers, Addison-Wesley, Reading, MA, 1995.

[2] T. Bedford, R.M. Cooke, Probabilistic Risk Analysis: Foundations and Methods, Cambridge University Press, Cambridge, 2001.

[3] M. Rausand, A. Hoyland, System Reliability Theory: Models, Statistical Methods and Applications, second ed., Wiley Interscience, New York, 2004.

[4] National Aeronautics and Space Administration, Fault Tree Analysis–A Bibliography, Technical Report NASA/SP-2000-6111, 2000.

[5] D.F. Haasl, Advanced concepts in fault tree analysis, in: Proceedings of System Safety Symposium, Seattle, WA, 1965.

[6] C.A. Ericson, Fault tree analysis–a history, in: Proceedings of the 17th International System Safety Conference, Orlando, 1999.

[7] W.E. Vesely, F.F. Goldberg, N.H. Roberts, D.F. Haasl, Fault Tree Handbook, NUREG-0492, US Nuclear Regulatory Commission, Washington, DC, 1981.

[8] W.E. Vesely, J. Dugan, J. Fragola, J. Minarick, J. Railsback, Fault Tree Handbook with Aerospace Applications, National Aeronautics and Space Administration, NASA, 2002.

[9] W.S. Lee, D.L. Grosh, F.A. Tillman, C.H. Lie, Fault tree analysis, methods, and applications–a review, IEEE Trans. Reliab. 34 (1985) 194-203.

[10] N.D. Singpurwalla, Foundational issues in reliability and risk analysis, SIAM Rev. 30 (1988) 264-282.

[11] A. Rauzy, New algorithms for fault tree analysis, Reliab. Eng. Syst. Safety 40 (1993) 203-211.

[12] O. Coudert, J.C. Madre, Metaprime: An interactive fault-tree analyzer, IEEE Trans. Reliab. 43 (1994) 121-127.

[13] J.D. Andrews, T.R. Moss, Reliability and Risk Assessment, second ed., Professional Engineering Publishing, London, 2002.

[14] A. Rauzy, Binary decision diagrams for reliability studies, in: K.B. Misra (Ed.), Handbook of Performability Engineering, Springer, London, 2008, pp. 381-396.

[15] R. Remenyte-Prescott, J.D. Andrews, An efficient real-time method of analysis for non-coherent fault trees, Qual. Reliab. Eng. Int. 25 (2009) 129-150.

[16] C. Ibáñez-LLano, A. Rauzy, E. Meléndez, F. Nieto, A reduction approach to improve the quantification of linked fault trees through binary decision diagrams, Reliab. Eng. Syst. Safety 95 (2010) 1314-1323.

[17] S. Reed, J.D. Andrews, S.J. Dunnett, Improved efficiency in the analysis of phased mission systems with multiple failure mode components, IEEE Trans. Reliab. 60 (2011) 70-79.

[18] A. Mentes, I.H. Helvacioglu, An application of fuzzy fault tree analysis for spread mooring systems, Ocean Eng. 38 (2011) 285-294.

[19] J. Wu, S. Yan, L. Xie, Reliability analysis method of a solar array by using fault tree analysis and fuzzy reasoning Petri net, Acta Astronautica 69 (2011) 960-968.

[20] S.H. Han, H.-G. Lim, Top event probability evaluation of a fault tree having circular logics by using Monte Carlo method, Nuclear Eng. Design 243 (2012) 336-340.

[21] Y. Wang, Q. Li, M. Chang, H. Chen, G. Zang, Research on fault diagnosis expert system based on the neural network and the fault tree technology, Procedia Eng. 31 (2012) 1206-1210.

[22] R.E. Barlow, F. Proschan, Statistical Theory of Reliability and Life Testing: Probability Models, To Begin With, Silver Spring, MD, 1981.

[23] S. Beeson, J. Andrews, Calculating the failure intensity of a non-coherent fault tree using the BDD technique, Qual. Reliab. Eng. Int. 20 (2004) 225-235.

[24] T.L. Chu, G. Apostolakis, Methods for probabilistic analysis of noncoherent fault trees, IEEE Trans. Reliab. R-29 (1980) 354-360.

[25] L. González, A new method for ordering binary states probabilities in reliability and risk analysis, Lecture Notes Comput. Sci. 2329 (2002) 137-146.

[26] L. González, N-tuples of 0s and 1s: Necessary and sufficient conditions for intrinsic order, Lecture Notes Comput. Sci. 2667 (2003) 937-946.

[27] L. González, D. García, B. Galván, An intrinsic order criterion to evaluate large, complex fault trees, IEEE Trans. Reliab. 53 (2004) 297-305.

[28] L. González, A new algorithm for complex stochastic Boolean systems, Lecture Notes Comput. Sci. 3980 (2006) 633-643.

[29] L. González, A picture for complex stochastic Boolean systems: The intrinsic order graph, Lecture Notes Comput. Sci. 3993 (2006) 305-312.

[30] L. González, Algorithm comparing binary string probabilities in complex stochastic Boolean systems using intrinsic order graph, Adv. Complex Syst. 10 (2007) 111-143.

[31] L. González, Ranking intervals in complex stochastic Boolean systems using intrinsic ordering, in: B.B. Rieger, M.A. Amouzegar, S.-I. Ao (Eds.), Machine Learning and Systems Engineering, Springer, New York, 2010, pp. 397-410.

[32] L. González, Edges, chains, shadows, neighbors and subgraphs in the intrinsic order graph, IAENG Int. J. Appl. Math. 42 (2012) 66-73.

[33] W.G. Schneeweiss, Boolean Functions with Engineering Applications and Computer Programs, Springer, Heidelberg, New York, 1989.

[34] R.P. Stanley, Enumerative Combinatorics, Vol. 1, Cambridge University Press, Cambridge, 1997.

[35] US Nuclear Regulatory Commission, Reactor Safety Study: An Assessment of Accident Risks in U.S. Commercial Nuclear Power Plants, Technical Report NUREG-75/014: WASH-1400, 1975.