Integrating several formalisms in order to increase Fault Trees' modeling power

Daniele Codetta-Raiteri
Dipartimento di Informatica, Università del Piemonte Orientale, Italy
e-mail: *raiteri@mfn.unipmn.it*

Abstract

The Fault Tree (FT) is a widespread model in the field of Reliability, but its modeling power is very limited. Therefore several FT extensions have been proposed in the literature, each introducing particular modeling primitives, but in a separate way. In this paper, we integrate the primitives coming from three relevant FT extensions (Parametric, Dynamic, and Repairable FT), into the formalism called Generalized FT (GFT). We define each primitive in such a way that it can be combined with any other one. This allows to compactly represent redundancies and symmetries of the system structure, set several kinds of dependency among the events, and model repair processes, in the same model. The paper provides also an analysis process for GFT models based on the modular approach. In particular, we provide the conditions to detect modules, considering the presence of all the primitives. Besides modules, we exploit the parametric form also at the solution level, with the aim of reducing the cost of analysis.

Keywords: Fault Trees, Boolean gates, dynamic gates, repair, parametric form, modules.

1 Introduction

The Fault Tree (FT)[1] [1] is a widespread model in the Reliability field and can be analyzed by generating the *Binary Decision Diagram* (BDD) [2] representing the same Boolean formula incorporated by the FT. Then, the unreliability (and other results) can be efficiently computed on the BDD [2]. However, the FT modeling power is rather limited: Basic Events (BE) are assumed to be binary and independent, while the combinations of BEs leading to the Top Event (TE), can only be expressed by means of *Boolean gates*. In order to overcome such limits and assumptions, several FT extensions have been proposed in the literature. Each of them inherited only the elements of the original FT formalism (events and Boolean gates), and introduced new modeling capabilities. So, the particular features of a FT extension are not available in the other extensions. For this

[1]The list of all the acronyms mentioned in this paper is available in Tab. 1.

Acronym	Meaning	Acronym	Meaning
BDD	Binary Decision Diagram	IE	Intermediate Event
BE	Basic Event	k:n	k out of n
BRE	Basic Replicator Event	PAND	Priority AND gate
Cov_{BE}	basic Coverage set	pBDD	Parametric Binary Decision Diagram
Cov_E	Coverage set	pBox	Parametric Box
CSM	Combinatorial Solution Module	pFDEP	Probabilistic Functional DEPendency gate
CTMC	Continuous Time Markov Chain	PFT	Parametric Fault Tree
DBN	Dynamic Bayesian Networks	RB	Repair Box
DFT	Dynamic Fault Tree	RE	Replicator Event
FDEP	Functional DEPendency gate	RFT	Repairable Fault Tree
FT	Fault Tree	SEQ	SEQuence enforcing gate
GFT	Generalized Fault Tree	SP	SPare gate
GRT	Global Repair Time policy	SSM	State space Solution Module
GSPN	Generalized Stochastic Petri Net	SWN	Stochastic Well-formed Net

Table 1: The list of the acronyms.

reason, in this paper, we integrate the modeling primitives present in three relevant FT extensions (Sec. 2), into a single formalism called *Generalized Fault Tree* (GFT) (Sec. 4). This determines a relevant increase of the modeling power because in a GFT model, we can exploit in an isolated or combined way, all the recent modeling capabilities such as the compact modeling of redundancies, the dependencies among the events, the repair of components or subsystems. However, the integration of different formalisms into a single one, is not so straightforward. In order to use in a combined way, primitives coming from different FT extensions, we must provide for each primitive: **1)** a notation such that the primitive becomes compatible with any other one; **2)** a semantics considering the influence of the other primitives.

We do not limit our attention to the definition of a generalized formalism, but we provide also an analysis process for GFT models (Sec. 5), oriented to reduce the computing costs of the analysis; to this aim, we exploit: **1)** the modular approach [3] based on the detection and on the analysis in isolation of the *modules* which are the subtrees independent from the rest of the GFT model. This avoids the analysis of the whole model in a single complex solution step. We extend the modules detection technique in such a way to take into account the presence and the role of all the primitives. **2)** The parametric form [4] consisting of the compact representation of symmetries and redundancies, and exploited at both the modeling and the analysis level, to reduce the model size. In particular, the analysis of modules in isolation is performed by resorting to the parametric version of *Stochastic Petri Nets* [1] and BDDs.

The case study described in Sec. 3 and the corresponding GFT model (Fig. 5) support the description of the unified formalism and of the analysis process.

2 Fault Tree extensions

This section provides basic notions about the modeling primitives and the analysis techniques introduced in each of the FT extensions that will be concentrated in the GFT formalism (Sec. 4).

Parametric Fault Trees. One of the ways to improve the Reliability of a system, consists of replicating its critical components or subsystems; in this cases, the construction and the analysis of the FT may become quite unpractical because the model will be composed by several identical (large) subtrees representing the replicated parts in the system. *Parametric Fault Trees* (PFT) [4] were proposed with the purpose of providing the compact modeling of such parts. Using PFTs, identical subtrees are folded into a single parametric subtree, while the identity of each replica is maintained through the possible values of the parameters. From a PFT, the equivalent *Parametric Binary Decision Diagrams* (pBDD) [5] can be generated; the pBDD maintains the parametric form which is exploited in the analysis, together with the BDD advantages. Applications of the PFT formalism are described in [6].

Dynamic Fault Trees (DFT) [7] introduce *dynamic gates* representing several kinds of dependency between events: functional dependencies, dependencies concerning the order of events, and the presence of spare components. Due to the presence of dependencies in the model, DFTs need the state space solution; this means generating all the possible system states and stochastic transitions between states. In other words, we need to obtain a *Continuous Time Markov Chain* (CTMC) [1]. However, the state space analysis suffers from very high computing costs if the system is complex. For this reason, a modular approach to analyze DFTs, was proposed in [3]: first, the DFT *modules* (Sec. 1) are detected. Each module is analyzed in isolation with the proper technique: CTMC (if it contains dynamic gates) or BDD (if it contains only Boolean gates). In this way, the state space analysis is limited to the first class of modules. The software tool *Galileo* [8] exploits this technique.

Repairable Fault Trees (RFT) [9] introduce a new primitive called *Repair Box* (RB) representing the presence of a repair process involving a certain set of components, and activated by the occurrence of a specific failure event concerning a component or a subsystem. This establishes some dependencies among the failure and the repair events, so the state space analysis is required, but only by the subtrees where RBs are present. Therefore the modular approach can be applied to RFTs as well. The state space analysis of a RFT module can be performed by conversion into a *Generalized Stochastic Petri Net* (GSPN) [1] and by exploiting the available GSPN solution techniques [9]. Applications of the RFT formalism are available in [10].

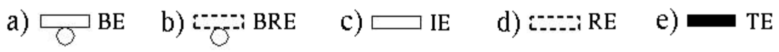

Figure 1: Types of event: a) Basic Event. b) Basic Replicator Event. c) Intermediate Event. d) Replicator Event. e) Top Event.

3 The case study

This section provides the case study of a multiprocessor computing system inspired from [4] and adapted in such a way to use all the GFT modeling primitives (Sec. 4) in the corresponding model (Fig. 5). The system is composed by two main subsystems (Fig. 4.a): the computing module (CM) and the disk access (DA). The failure of any of them causes the system failure.

CM is composed by three redundant processing units ($PU1$, $PU2$, $PU3$), so CM fails if all of them are failed. Each processing unit is composed by one processor and one internal memory; in the case of $PU1$, they are $P1$ and $M1$ respectively. CM contains also two spare (Sec. 4.4) memories ($R1$, $R2$) connected to all the processing units by means of the bus B. $R1$ or $R2$ can replace any internal memory if it fails. A processing unit fails if its processor fails, or if its internal memory fails and no spare memories are available to replace it. A spare memory is available if it is not failed, it is not already replacing another internal memory, and B has not failed. As soon as DA fails, a repair process recovering $D1$ and $DBUS$ is activated.

The subsystem DA contains the primary hard disk $D1$, the backup disk $D2$, and a particular device named BD performing the periodical update of $D2$. Initially, the processing units access $D1$, while $D2$ is only periodically accessed by BD for the update operations; so we assume that in this situation, $D2$ can not fail. If $D1$ fails, the processing units access $D2$ instead of $D1$, and from this moment, $D2$ may fail. Both $D1$ and $D2$ can be accessed through the bus $DBUS$. The failure of DA happens if $DBUS$ fails, if both $D1$ and $D2$ are failed, or if $D2$ is not updated when $D1$ fails. The third condition requires that both $D1$ and BD are failed and the failure of BD occurred before the failure of $D1$. If instead $D1$ fails before BD, this has no relevance because the update operation is no more necessary. The failure of a any processing unit activates a repair process acting on the corresponding processor and internal memory.

4 The generalized formalism

We now present the GFT formalism including all the primitives introduced in FT, PFT, DFT and RFT (Sec. 2). The description of their notation, semantics, parametric form, and integration, is supported by the GFT model of the case study (Fig. 5).

4.1 Events

Events concern the failure of particular parts of the system. We can consider an event as a Boolean variable: it is initially $false$ to represent the working state, and it may become $true$ to represent the failed state.

Atomic events. ***Basic Events*** (BE) (Fig. 1.a) model the failure of the elementary components of the system; a BE turns from $false$ to $true$ after a random time ruled by a probability distribution; in this paper we assume the negative exponential one; so, a *failure rate* (λ) is associated with the BE. ***Intermediate Events*** (IE) (Fig. 1.c) represent the failure of subsystems; their Boolean value is the output of a gate. Finally, we have a unique event called ***Top Event*** (TE) (Fig. 1.e) modeling the failure of the whole system; the TE must be the output of a gate and can not be the input of any gate.
Case study. The events $DBUS$ and DA in the GFT in Fig. 5 are examples of BE and IE respectively[2].

Replicator events. Using the parametric form, the subtrees with the same structure and the same failure rates (replicated subtrees) can be folded into a single parametric subtree, while their identities are maintained through the possible values of a parameter. Such values range over the *type* [4] of the parameter; the cardinality of the

[2]The events in the model in Fig. 5 are listed in Tab. 2 providing their names and their meaning.

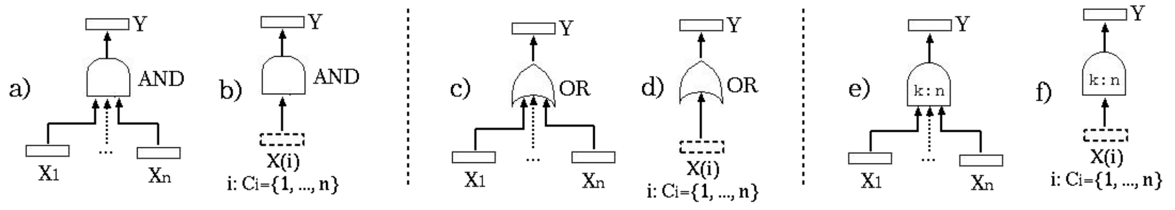

Figure 2: Boolean gates: a,b) AND. c,d) OR. e,f) k out of n.

type establishes the number of replicated subtrees folded into the parametric subtree. The root of a parametric subtree is a ***Replicator Event*** (RE) (Fig. 1.d); a parameter and the corresponding type are declared when the RE is instanced. Such parameter is associated also with the events in the parametric subtree if an instance of them is present in each of the replicated subtrees represented in compact form by the parametric subtree.

A parametric subtree may contain inner parametric subtrees. In this case, more than one parameter may be associated with the events in the inner parametric subtrees. The rules about the use of multiple parameters are detailed in [4]. In general, if a RE has several parameters, the relative parametric subtree folds as many replicated subtrees as the possible combinations of the parameters' values.

The ***Basic Replicator Event*** (BRE) (Fig. 1.b) folds several BEs with the same failure rate and connected to the same gate. To this aim, a parameter and the associated type are declared in a BRE which may have other parameters if it is nested in one or more parametric subtrees.

Subtree. We indicate with $\widehat{A}$ the subpart of the model composed by the IE or RE A and all the events A' (and the corresponding gates) such that a path exists between A' and A according to the arcs orientation. In other words, $\widehat{A}$ is the subtree "rooted" in A.

Case study. The DFT model of the case study (Fig. 4.b) is characterized by the presence of the following replicated subtrees: $\widehat{PU1}$={$P1$, $MEM1$, $M1$, $R1$, $R2$, B}; $\widehat{PU2}$={$P2$, $MEM2$, $M2$, $R1$, $R2$, B}; $\widehat{PU3}$={$P3$, $MEM3$, $M3$, $R1$, $R2$, B}. Because of their symmetry, they are folded in $\widehat{PU(i)}$={$P(i)$, $MEM(i)$, $M(i)$, $R(j)$, B} inside the GFT model in Fig. 5. The root is the RE $PU(i)$ where the parameter i of type $C_i = \{1, 2, 3\}$ is declared. Such parameter identifies the replicated subtrees folded in $\widehat{PU(i)}$, and is associated also with the BEs $P(i)$, $M(i)$, and the IE $MEM(i)$ because distinct instances of them are present in each of the replicated subtrees. The BRE $R(j)$ with the parameter j of type $C_j = \{1, 2\}$, folds the BEs $R1$ and $R2$. The parameter i is not associated with $R(j)$ and with the BE B, because such events belong to all the subtrees folded in $\widehat{PU(i)}$.

4.2 Boolean gates

Notation. If g is a Boolean gate, g is connected to the input events $X_1, \ldots, X_n$ ($n \geq 2$), and to the output event Y; arcs respect the logic circuit orientation (Fig. 2.a, Fig. 2.c, Fig. 2.e).

Parametric form. $X(i)$ is connected to g as input event (Fig. 2.b, Fig. 2.d, Fig. 2.f). If $X(i)$ is a BRE, then it folds n BEs with the same failure rate; if instead $X(i)$ is a RE, then it is the root of a parametric subtree folding n replicated subtrees (Sec. 4.1). The parameter i ranges over its type $C_i = \{1, \ldots, n\}$.

Semantics. $X_1, \ldots, X_n$ (possibly folded in $X(i)$) are independent. The value of Y is given by the Boolean function corresponding to the gate, applied on the values of $X_1, \ldots, X_n$:

- **AND** gate (Fig. 2.a and Fig. 2.b): Y is $true$ (failed) if all $X_1, \ldots, X_n$ (possibly folded in $X(i)$) are $true$, and is $false$ (working) otherwise.
- **OR** gate (Fig. 2.c and Fig. 2.d): Y is $false$ if all $X_1, \ldots, X_n$ (possibly folded in $X(i)$) are $false$, and is $true$ otherwise.
- **k out of n** (k:n) gate (Fig. 2.e and Fig. 2.f): Y is $true$ if at least k ($1 < k < n$) among $X_1, \ldots, X_n$ (possibly folded in $X(i)$) are $true$, and is $false$ otherwise.

Case study. In Fig. 5, because of AND gates, MS is failed if both $D1$ and $D2$ are failed, while CM is failed if all the events folded in $PU(i)$ are failed. Because of OR gates, DA fails if at least one among $DBUS$, UPD, MS fails, while $PU(i)$ for a certain value of i, fails if $P(i)$ or $MEM(i)$ (for the same value of i) fails.

4.3 The Repair Box (RB)

A RB indicates the presence of a repair process involving a subsystem, graphically appears as a wrench inside a square, and is connected by means of oriented arcs to the events in the GFT.
Trigger. The event T connected to the RB b by means of the oriented arc (T, b) is called trigger event; its aim is twofold: its occurrence enables the repair action modeled by b, and it is the "root" of the subtree whose events are influenced by the repair action. The event triggering a RB can be of any type.
Parametric form. If a parameter is associated with T, then b models the presence of several repair processes, each acting on one of the symmetric subsystems represented in compact form by $\widehat{T}$.
Basic coverage. A RB b is connected to a set of BEs or BREs representing the components of the subsystem to be repaired; this set is called the basic coverage set of b ($Cov_{BE}(b)$); given the event A belonging to $Cov_{BE}(b)$, b is connected to A by means of the oriented arc (b, A). If A is a BE, the effect of the RB b is setting the value of A to $false$ (working), if it is currently $true$ (failed). If instead A is a BRE, the effect of b is setting to $false$ the value of any BE folded in A.
Coverage. The effect of the RB is not limited to Cov_{BE}: setting the value of the event A in $Cov_{BE}(b)$ to $true$, may determine the change of the value of an IE or RE A' such that a path exists from A to A' according to the arcs orientation. So, the coverage set of the RB b ($Cov_E(b)$) is composed by all the events whose value is influenced by the action of b in a direct or indirect way.
Case study. In Fig. 5, $Cov_{BE}(REP1) = \{DBUS, D1\}$. So, when the trigger event DA occurs, a repair process involving $DBUS$ and $D1$ starts. This has actually effect on $Cov_E(REP1) = Cov_{BE}(REP1) \cup \{UPD, MS, DA, TE\}$. In the case of $REP2$, when the trigger event $PU(i)$ occurs for a certain value of i, a repair process starts and involves $Cov_{BE}(REP2) = \{P(i), M(i)\}$, for the same value of i. This actually influences $Cov_E(REP2) = Cov_{BE}(REP2) \cup \{MEM(i), PU(i), CM, TE\}$.
Policy. A RB is characterized also by a *repair policy* that specifies each aspect of the repair process. In Fig. 5 we assume the *Global Repair Time* (GRT) policy [9]: the detection of the trigger is immediate, there is no limit to the number of components under repair at the same time, and their repair is considered as an atomic action happening after a random period of time beginning when the trigger event occurs. Because of the last feature, the time to repair is ruled by a global repair rate (μ) associated with the RB, according to the negative exponential distribution.

4.4 Dynamic gates

Dynamic gates establish several kinds of dependency among the events, and differ from Boolean gates also for other aspects: **1)** they may have trigger and dependent events, instead of input and output events; **2)** the input, trigger, or dependent events may be ordered. In particular, the arcs connecting the input and output events to the dynamic gate respect the logic circuit orientation. The trigger event T is connected to the gate g by means of the arc (T, g), while the arc (g, D) connects the dependent event D. The description of the notation and the semantics of every dynamic gate follows.

Probabilistic Functional Dependency Gate (pFDEP).
Notation. A pFDEP gate g is connected to one trigger event T and to a set of dependent events D_1, ..., D_n ($n \geq 1$) (Fig. 3.a).
Semantics. If T becomes $true$ (failed), D_1, ..., D_n becomes $true$ as well, with probability p ($0 < p \leq 1$). pFDEP [11] extends the original FDEP [7] where $p = 1$.
Parametric form. g can be connected to the dependent BRE or RE $D(i)$ (Fig. 3.b). If so, $C_i = \{1, \ldots, n\}$ (the type of i, with $n \geq 2$) establishes the number of events functionally dependent on T, that are folded in $D(i)$. The trigger event T must not be a BRE or a RE, otherwise this would not be consistent with the original notation of this gate [7] stating that the trigger has to be a single atomic event.
Case study. In Fig. 5, the failure of B determines the failure of the components represented by $R(j)$, with probability $p = 1$.
Integration with RB. The presence of a RB b may influence the semantics of the dynamic gates. In the case of pFDEP, let us suppose that $D_1, \ldots, D_n$ (possibly folded in $D(i)$), or a subset of them, belong to $Cov_E(b)$. If T fails, $D_1, \ldots, D_n$ fail as well. If later they are repaired, has T still effect on them? We can distinguish between **1)** *instantaneous* effect: T determines the failure of $D_1, \ldots, D_n$ only in the instant when it becomes failed, so

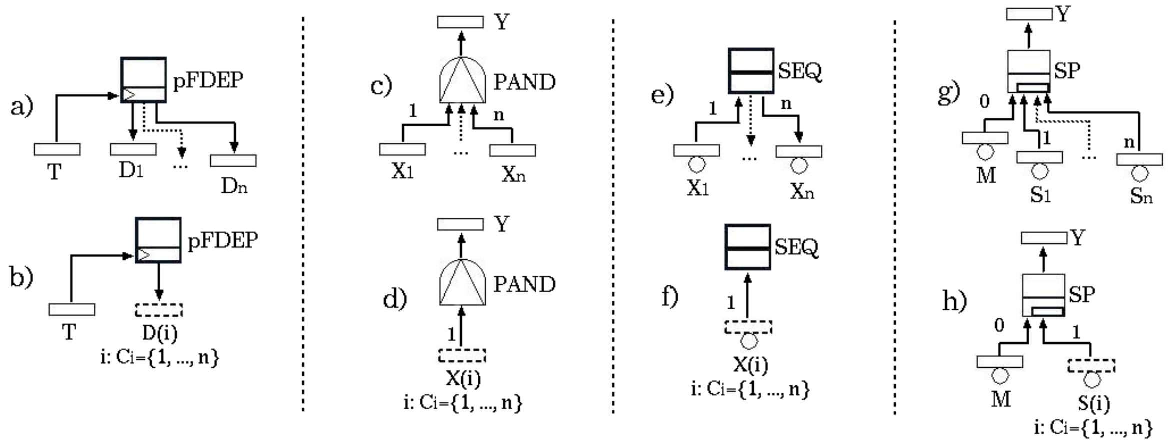

Figure 3: Dynamic gates: a,b) Probabilistic Functional Dependency. c,d) Priority AND. e,f) Sequence Enforcing. g,h) Spare gate.

Event	type	Component / Subsystem	Event	type	Component / Subsystem
B	BE	Memory Bus	$M(i)$	BE	Internal Memory of the Processing Unit i
BD	BE	Device for the Backup Disk update	$MEM(i)$	IE	Memory access of the Processing Unit i
CM	IE	Computing module	MS	IE	Mass Storage
$D1$	BE	Primary Disk	$P(i)$	BE	Processor of the Processing Unit i
$D2$	BE	Backup Disk	$PU(i)$	RE	Processing Unit i
DA	IE	Disk Access	$R(j)$	BRE	Spare Memory j
$DBUS$	BE	Disk Bus	UPD	IE	Backup disk Update

Table 2: The list of the events in the GFT in Fig. 5.

if later $D_1, \ldots, D_n$ are repaired, they actually become working. **2)** *Permanent* effect: T causes the failure of $D_1, \ldots, D_n$ while it is in the failed state; therefore if we repair $D_1, \ldots, D_n$ they immediately turn again failed because of the permanent effect of T. In Fig. 5, we assume the permanent effect.

Priority And (PAND).
Notation. A PAND gate g is connected to the input events X_1, ..., X_n ($n \geq 2$) by means of ordered arcs, and to the output event Y (Fig. 3.c).
Semantics. The event Y is $true$ (failed) if all the input events are $true$ and they occurred in this order: $X_1 \prec X_2 \prec \ldots \prec X_n$. Y is $false$ (working) in any other case.
Parametric form. The input event of a PAND gate can be the RE or BRE $X(i)$ (Fig. 3.d). If the type of i is $C_i = \{1, \ldots, n\}$ ($n \geq 2$), then the value of Y is $true$ if all the events folded in $X(i)$ are $true$ and occurred respecting the increasing order of the elements of C_i.
Case study. In Fig. 5, UPD fails if both BD and $D1$ are failed, and BD failed before $D1$.
Contemporary input events. A semantic problem arises if we take into account the possibility of input events occurring in the same instant. In this sense, two versions of the PAND gate can be defined: one where contemporary input events determine the $true$ value of the output event, and one where they determine its $false$ value. In Fig. 5, we assume the first version of the gate.
Integration with RB. In Fig. 5, $D1$ belongs to $Cov_{BE}(REP1)$, so $D1$ is repairable: the value of $D1$ may change repeatedly from $false$ to $true$, and vice-versa, during time. Let us suppose that $D1$ fails and is repaired; then, BD fails. In this case, has the failure order been respected or not? Actually, in the moment when BD fails, $D1$ is not failed; this can be interpreted as the respect of the failure order. However, in the moment when BD fails, $D1$ had previously been in the failed state; from this point of view, the failure order has not been respected. In other words, when we deal with a gate of type PAND having some repairable input event X, we have to decide if the first or the last failure of X must be taken into account in the verification of the failure order of the input events. In Fig. 5, we consider the first failure of the repairable component.

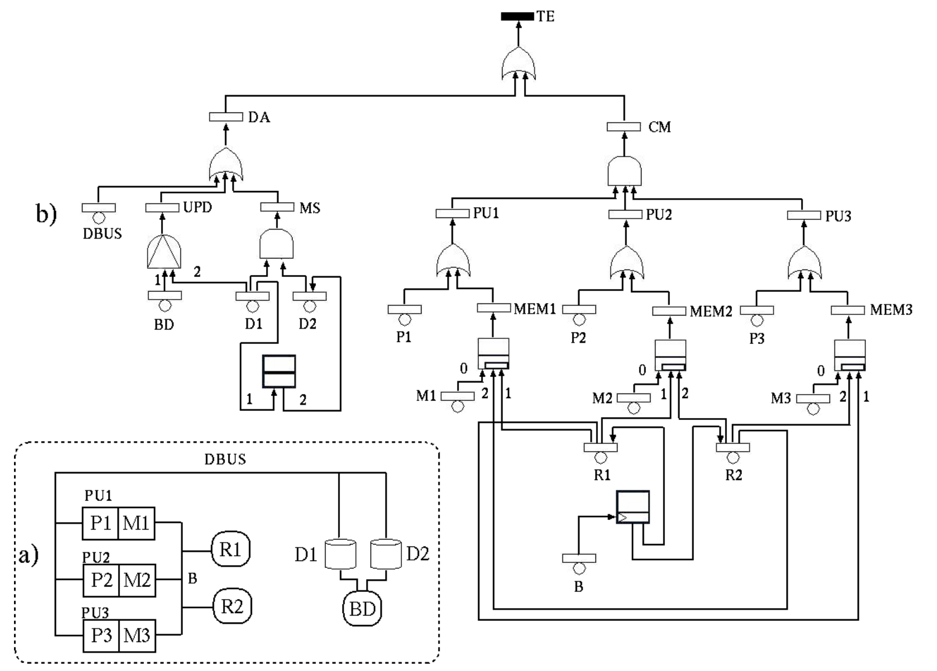

Figure 4: a) The scheme of the case study. b) The DFT equivalent to the GFT model in Fig. 5 assuming no repair.

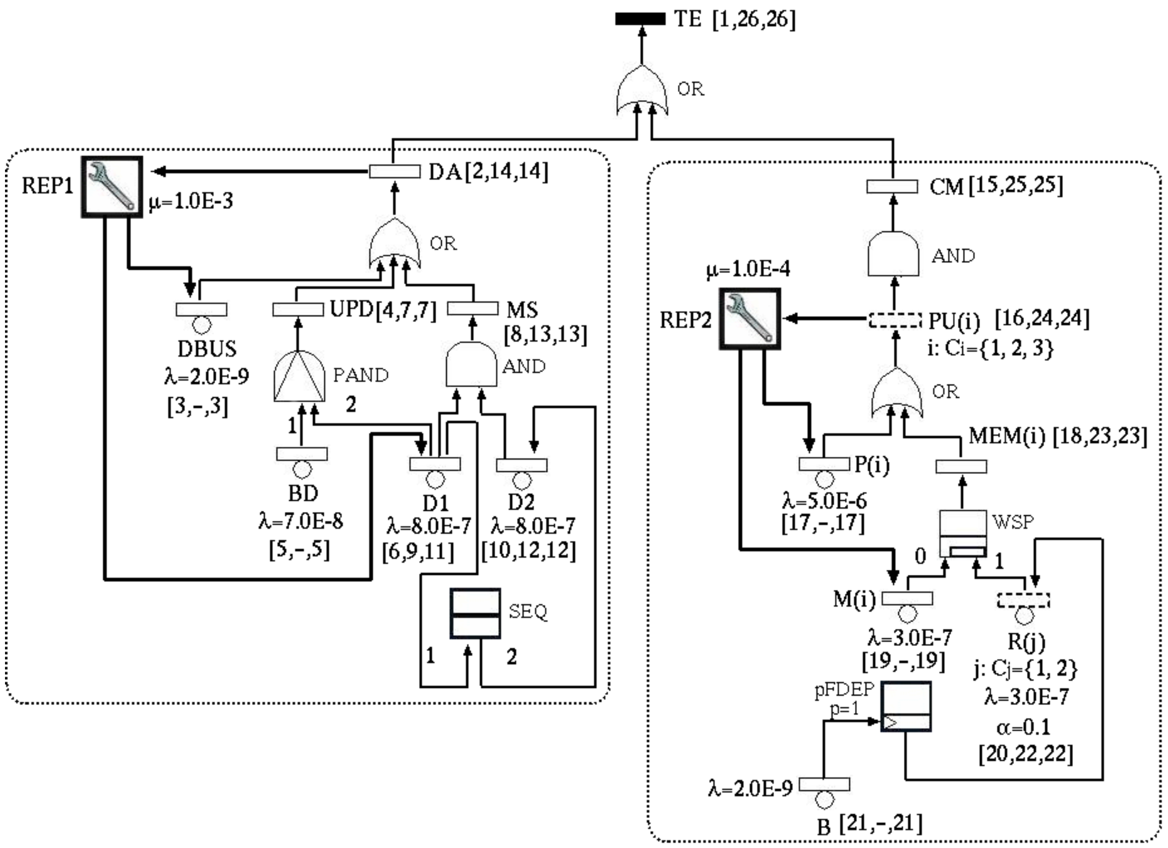

Figure 5: The GFT model of the case study: the minimal SSMs $\widehat{DA}$ and $\widehat{CM}$ are put in evidence. The list of events and their meaning is available in Tab. 2.

Sequence Enforcing Gate (SEQ).
Notation. A SEQ gate g is connected to the trigger BE X_1, and to the dependent BEs $X_2, \ldots, X_n$ ($n \geq 2$) by means of ordered arcs (Fig. 3.e).
Semantics. $X_1, \ldots, X_n$ are forced to turn to the $true$ value respecting a specific order: $X_1 \prec X_2 \prec \ldots \prec X_n$. In other words, X_{j+1} may become $true$ only after that X_j has become $true$. X_1 acts as the trigger of the gate because it is independent (X_1 is the only one initially enabled to occur).
Parametric form. g can be connected to the BRE $X(i)$ (Fig. 3.f). If the type of i is $C_i = \{1, \ldots, n\}$ ($n \geq 2$), the BEs folded in $X(i)$ are forced to occur respecting the increasing order of the elements of C_i.
Case study. In Fig. 5, $D2$ may fail only after the failure of $D1$.
Integration with RB. In Fig. 5, $D1$ belongs to $Cov_{BE}(REP1)$. Let us suppose that $D1$ fails and is repaired; if $D2$ has not yet failed, is it now possible the failure of $D2$? We can establish that $D2$ can not fail now because $D1$ is not currently failed, or we can say that $D2$ can fail because $D1$ failed in the past. In other words, when we deal with a SEQ gate acting on some repairable component X_j, we have to decide if the successor X_{j+1} may fail if X_j is currently failed, or if X_j failed at least once in the past. In Fig. 5, we assume that the first interpretation holds.

Spare Gate (SP).
Notation. A SP gate is connected to the input BEs $M, S_1, \ldots, S_n$ ($n \geq 1$) by means of ordered arcs, and to the output event Y. M is distinguished from the other input event by assigning an order number 0 to the corresponding arc (Fig. 3.g).
Semantics. This gate models the presence of a main component M and a set of spare components $S_1, \ldots, S_n$ ($n \geq 1$) with the aim of replacing M in its function if M fails. $S_1, \ldots, S_n$ are not binary events, but they can be instead in three states: dormant (or stand-by), working, failed. A spare is initially dormant and it turns to the working state if it has to replace the main component; a spare may fail both in the dormant and in the working state. The spare failure rate changes depending on its current state: if the failure rate of the spare S_j is λ_{S_j} in the working state, $\alpha_{S_j}\lambda_{S_j}$ is its failure rate in the dormant state, with $0 \leq \alpha_{S_j} \leq 1$ (α_{S_j} is called dormancy factor). If the working spare S_j fails, M will be replaced by the spare S_{j+1} (if any). The order of activation of the spares is given by the order of the corresponding arcs. The output event Y is $true$ if M is $true$ and all $S_1, \ldots, S_n$ are $true$.
Parametric form. The input events of the gate are the BE M (main component) and the BRE $S(i)$. $S(i)$ represents a set of spare components with the same failure rate and the same dormancy factor (Fig. 3.h). If $C_i = \{1, \ldots, n\}$ ($n \geq 2$) is the type of the parameter i, the order of activation of the spares is given by the increasing order of the elements of C_i.
Integration with RB. If M is repairable, the effect of its repair is twofold: M turns from the failed to the working state, and the spare component S_j currently replacing the main one, turns from the working state to the dormant state (stand-by). If later, M fails again, S_j is still available to replace the main one.
Case study. In Fig. 5, $M(i)$ belongs to $Cov_{BE}(REP2)$; therefore the main component $M(i)$ (for any value of the parameter i) may undergo repair and be replaced by one of the spare $R(j)$.

5 Module based GFT analysis

The analysis of a GFT model requires the generation of its *state space* (Sec. 2); this is due to the presence of dynamic gates and RBs establishing dependencies among the events in the model. The state space analysis of the whole GFT model is typically computationally expensive, or even impracticable, since the number of states tends to grow exponentially with the number of components. A way to reduce the cost of the solution, consists of generating and analyzing in isolation the state space of each subtree containing dynamic gates or RBs. Such subtrees must be selected in such a way that their analysis in isolation does not compromise the analysis of the rest of the model. So they must be *modules* (Sec. 1); in Sec. 5.1, we introduce the conditions that a subtree in a GFT must satisfy in order to be classified as a module. A GFT module requiring the state space analysis can be converted into *Stochastic Well-formed Net* (SWN) [12] according to the rules described in [13]. SWNs extend GSPNs by introducing colored tokens which can be exploited to maintain the parametric form at the state space level: from the SWN, a symbolic (or lumped) state space can be generated [12]: its number of states is reduced with respect to the ordinary state space, proportionally to the degree of redundancy and symmetry characterizing the system and its model, as shown in Sec. 5.4. In this way, instead of analyzing the state space of the whole GFT model, we deal with several state spaces whose dimensions are reduced in a relevant way because they concern subparts of the model, and they are symbolic.

After the state space analysis of modules in isolation, the resulting GFT does not contain any dynamic gate or RB, but it may still contain parametric subtrees. Therefore it does not need the state space analysis, but it can be solved in form of BDD or pBDD (Sec. 2). In Sec. 5.3, the steps of the GFT analysis process are described in details and applied to the GFT model in Fig. 5.

5.1 Modules detection

In a GFT, a subtree (Sec. 4) $\widehat{M}$ is a module if it satisfies all the following conditions:

Structural independence. An event may belong to two or more subtrees such that each of them is not included in the other ones. For instance, in Fig. 5, $D1$ belongs to $\widehat{UPD}$ and $\widehat{MS}$.
Definition. A subtree $\widehat{M}$ is structurally independent if all the events in $\widehat{M}$ belong to subtrees included in $\widehat{M}$ or to subtrees including $\widehat{M}$.
If we ignore the RBs and their arcs, the structural independent subtrees in a GFT, can be identified by applying the modules detection algorithm developed for FTs [14]. This algorithm consists of a depth-first left-most visit of the model, starting from the TE and assuming the opposite orientation of the arcs. The arc orientation chosen in this paper for Boolean and dynamic gates (Fig. 2 and Fig. 3) is compatible with the algorithm. The structural independent subtrees are identified according to the variables $[t_1, t_2, t_l]$ indicating the steps of the visit where each event is visited for the first, second and last time, as shown in Fig. 5.
Case study. In Fig. 5, $\widehat{UPD}$ is not a structurally independent because it contains the event $D1$ that belongs also to $\widehat{MS}$, but $\widehat{MS}$ is not contained in $\widehat{UPD}$. The subtree $\widehat{DA}$ is instead structurally independent: its events belong to subtrees included in $\widehat{DA}$ and belong to $\widehat{TE}$ including $\widehat{DA}$. This holds also for $\widehat{CM}$.

Parametric independence. A parametric subtree folds several replicated subtrees, as explained in Sec. 4.1. A parametric subtree may contain an inner subtree that is *shared* by all the replicated subtrees. For instance, in Fig. 5, the subtree $\widehat{R(j)}$ belong to all the replicated subtrees folded in $\widehat{PU(i)}$ (Sec. 4.1).
Definition. A subtree is parametrically independent if it contains no parametric shared subtrees.
This condition can be verified by observing the parameter set of the events: $\widehat{M}$ is parametrically independent if all the parameters associated with M are associated with all the events belonging to $\widehat{M}$.
Case study. In Fig. 5, $\widehat{MEM(i)}$ and $\widehat{PU(i)}$ are not parametrically independent because they contain the events $R(j)$ and B where the parameter i is not present. They would be parametrically independent if they instead contained $R(i, j)$ and $B(i)$. The subtree $\widehat{CM}$ is parametrically independent because the parameter set of CM is empty, so it is contained in the parameter set of all the events belonging to $\widehat{CM}$.

Independence from dynamic gates. The input or dependent events of dynamic gates may be IEs or REs, which are the root of subtrees. So, the dynamic gate may establish some dependency also on the subtrees.
Definition. A subtree $\widehat{M}$ is independent from dynamic gates if no paths exist from M to an event A such that A is the input or the dependent event of a dynamic gate.
Case study. In Fig. 5 all the dynamic gates act on BEs or BREs, so all the subtrees rooted in IEs or REs are independent from dynamic gates.

Independence from repair. If T is the trigger event of a RB, the values of the events in the subtree $\widehat{T}$ depend on the occurrence of T and on the action of the RB.
Definition. A subtree $\widehat{M}$ is independent from repair if it is not contained in a larger subtree $\widehat{T}$ such that T is the trigger of a RB (such condition does not avoid M to be the trigger of a RB).
Case study. In Fig. 5, $\widehat{MEM(i)}$ is not independent from repair because it is included in $\widehat{PU(i)}$ where $PU(i)$ is the trigger of a RB. Despite this, $PU(i)$ is instead independent from repair because it does not belong to a larger subtree whose root is the trigger of a RB.

1.	**Modules detection**	8.	**GFT Aggregation**
2.	**Modules classification**	9.	**GFT conversion** to (p)BDD
3.	if $\widehat{TE}$ is a minimal SSM go to Step 12, else go to Step 4	10.	**(p)BDD analysis**
4.	**GFT Decomposition** into minimal SSMs	11.	end
5.	**Module parameters simplification**	12.	**GFT conversion** into SWN
6.	**Modules conversion** into SWNs	13.	**SWN analysis**
7.	**SWNs analysis**	14.	end

Table 3: The GFT analysis procedure.

5.2 Module classification

GFT modules can be classified according to the required analysis technique: a *Combinatorial Solution Module* (CSM) contains no RBs and no dynamic gates; a *State space Solution Module* (SSM) contains at least one RB or at least one dynamic gate. A SSM is *minimal* if it does not contain any inner SSM.

According to the independence conditions to be satisfied by a module, each dynamic gate or RB in the model is contained inside a minimal SSM. Therefore, minimal SSMs are the minimal parts of the model that specifically requires the state space analysis in isolation.

5.3 GFT analysis steps

In this paper, we are interested to compute the system unreliability at time t on the GFT model; this measure corresponds to the TE probability to be $true$ at time t. To this aim, the analysis of a GFT model exploiting modules, SWN and pBDD, follows the steps in Tab. 3:

Step 1. Modules are detected according to the independence conditions described in Sec. 5.1.
Case study. In Fig. 5, the structural independent subtrees are: $\widehat{TE}$, $\widehat{DA}$, $\widehat{CM}$, $\widehat{PU(i)}$, $\widehat{MEM(i)}$. The parametrically independent subtrees are: $\widehat{TE}$, $\widehat{DA}$, $\widehat{UPD}$, $\widehat{MS}$, $\widehat{CM}$. The subtrees independent from dynamic gates are: $\widehat{TE}$, $\widehat{DA}$, $\widehat{UPD}$, $\widehat{MS}$, $\widehat{CM}$, $\widehat{PU(i)}$, $\widehat{MEM(i)}$. The subtrees independent from repair are: $\widehat{TE}$, $\widehat{DA}$, $\widehat{CM}$, $\widehat{PU(i)}$. So, the set of modules is given by the intersection of the above sets: $\widehat{TE}$, $\widehat{DA}$, $\widehat{CM}$.

Step 2. Modules are classified according to the condition specified in Sec. 5.2.
Case study. In in Fig. 5, the modules $\widehat{TE}$, $\widehat{DA}$, $\widehat{CM}$ are all SSMs: each of them contains dynamic gates and RBs. $\widehat{TE}$ is not minimal because it includes $\widehat{DA}$ and $\widehat{CM}$ which are instead minimal.

Step 4. Minimal SSMs are detached from the GFT model in order to be analyzed in isolation. At this point, a problem arises: the root event M of a minimal SSM $\widehat{M}$ may have a non empty parameter set; in this case, M folds several events according to its parameter set. However M is the "TE" of $\widehat{M}$ and we have to compute its probability in a following step. Therefore M should be an atomic event.
Case study. $\widehat{DA}$ and $\widehat{CM}$, the minimal SSMs in Fig. 5, are detached from the GFT.

Step 5. We solve the problem in Step 4, by removing the parameter set of M from the parameter set of every event in $\widehat{M}$ (including M). This can be done because a module is a parametrically independent subtree, so the parameter set of every event inside $\widehat{M}$ includes the parameters of M (Sec. 5.1).

Step 6. Each of the minimal SSMs can be converted into a distinct SWN by means of the rules reported in [13]. In the resulting SWN, each event becomes a specific place which is initially empty to represent the $false$ value and becomes marked (containing one token) to represent the $true$ value. BEs and RBs generate stochastic transitions, while gates are mapped to immediate transitions. The types of parameters become the color classes [12] for tokens.
Case study. $\widehat{DA}$ and $\widehat{CM}$ are converted into SWN. The SWN obtained from $\widehat{CM}$ is shown in Fig. 6 where the portions corresponding to the gates and the RB are put in evidence.

Step 7. Each of the minimal SSMs in SWN form is analyzed in isolation requiring the probability of the place corresponding to the root event of the module, to be marked at time t. Such value corresponds to the probability of the SSM root event to be $true$ (failed).
Case study. Assuming that $t = 10000$ h, we compute on the SWN of $\widehat{CM}$ (Fig. 6), the probability that the place CM_dn contains one token at time t. This condition is equivalent to the $true$ value of the event CM (root of $\widehat{CM}$). In the same way, we compute the probability of the root event of $\widehat{DA}$. We obtain these probability values: $Pr\{DA = true, t\}$=$Pr\{\#DA\_dn = 1, t\}$ = 3.660650E-5. $Pr\{CM = true, t\}$=$Pr\{\#CM\_dn = 1, t\}$ = 2.966267E-5.

Step 8. Each of the minimal SSMs is replaced by an event. In particular, if the SSM root event was an IE, the module is replaced by a BE; if instead the module root event was a RE, it is replaced by a BRE. In both cases, the BE or BRE replacing the minimal SSM is characterized by: **1)** the same parameter set of the module root event before the simplification step (Step 5). **2)** a constant probability to occur (become $true$), which is the module root event probability at time t, computed in the Step 7. Now the GFT contains no dynamic gates and no RBs because they all belong to minimal SSMs that have been replaced by BEs or BRE; so, the model is actually a FT or a PFT.
Case study. In the GFT, the module $\widehat{DA}$ is replaced by the BE DA having 3.660650E-5 as probability to be $true$; the module $\widehat{CM}$ is replaced by the BE CM whose probability to be $true$ is 2.966267E-5. In this way, we obtain the GFT in Fig. 7.a, which is actually a FT.

Step 9. The corresponding BDD or pBDD can be generated according to the conversion rules reported in [5].
Case study. From the FT in Fig. 7.a, the BDD in Fig. 7.b is generated.

Step 10. A (p)BDD can deal with B(R)Es characterized by a failure rate or an occurrence probability at time t. In the first case, the probability of the B(R)E to occur at time t is computed in this step according to the failure rate. The resulting BDD or pBDD is analyzed returning the probability of the TE to be $true$ at time t, according to the rules described in [5].
Case study. The BDD in Fig. 7 is analyzed returning the probability of the TE to be $true$ at time t, that is 6.626809E-5. Tab. 4 reports the probabilities of the modules and of the whole model, for a time varying from $2000h$ to $10000h$, obtained by means of a prototypical software tool [13] implementing the GFT analysis steps.

Steps 12 and 13. It may happen that the whole GFT ($\widehat{TE}$) is a minimal SSM; in this case, the whole model is mapped to SWN and analyzed in this form.

5.4 Parametric form efficiency

Now we investigate the effect on the model, of the increase of processing units in the case study. In order to represent such increase in the GFT in Fig. 5, thanks to the parametric form, we only have to change the cardinality of the type C_i associated with the parameter i. If instead, we want to model such increase in the DFT model (Fig. 4.b), we have to add some new subtrees, as shown in Tab. 5 in terms of number of events. When we convert the minimal SSM $\widehat{CM}$ into SWN, the only variation still concerns C_i. Tab. 5 shows the number of symbolic states obtained from the SWN, and compares them with the corresponding ordinary ones, for a number of processing units varying from 3 to 10: the state space size reduction is evident, and is reflected in the time required for the state space analysis.

5.5 Alternative version of the case study

Fig. 8.a shows the GFT model of an alternative version of the original case study (Fig. 5), and presents several differences: **1)** the subtree $\widehat{DA}$ now contains no dynamic gates and no RBs, and is in parametric form. In particular, it represents two redundant disks (event $D(k)$), each supported by a specific bus (event $DBUS(k)$) instead of a common one. **2)** The spare memories are not shared among the processing units, but each unit is equipped with a distinct couple of reserved spare memories (event $R(i, j)$ instead of $R(j)$). **3)** The failure of

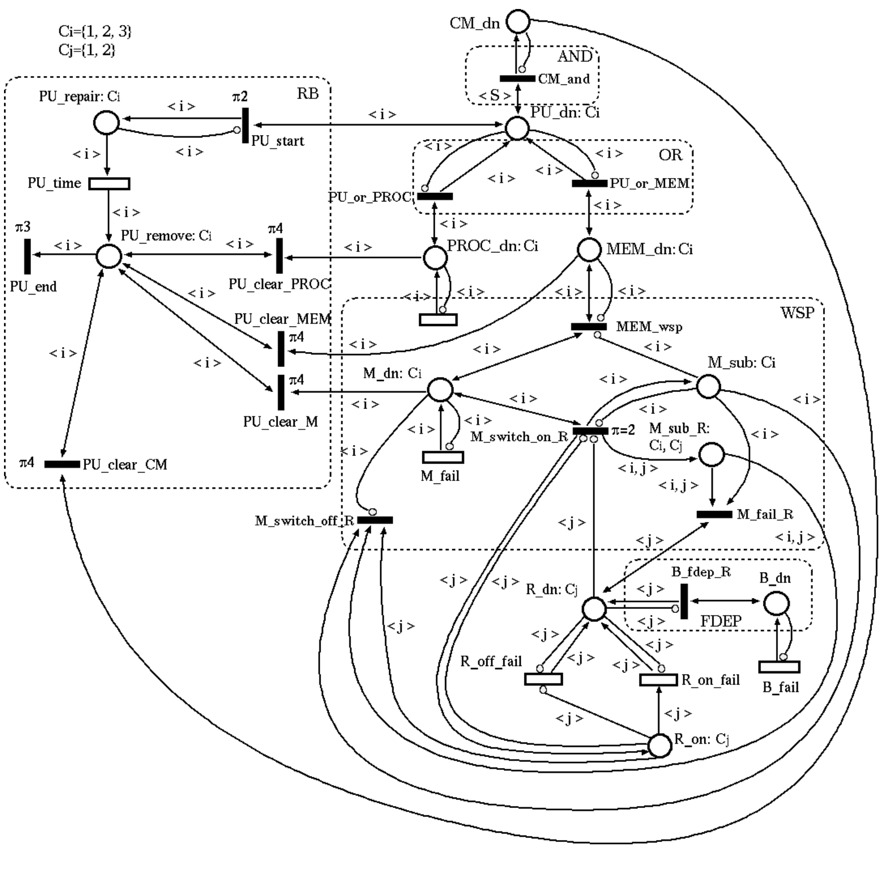
Figure 6: The SWN corresponding to the minimal SSM $\widehat{CM}$ of the GFT in Fig. 5.

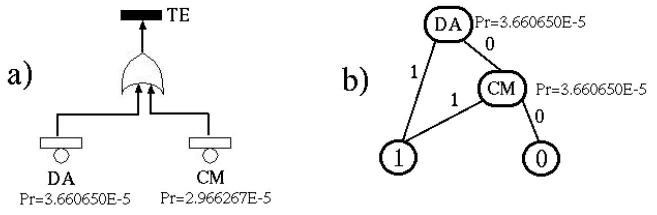

Figure 7: a) The GFT model in Fig. 5 after the analysis of the minimal SSMs. b) The BDD corresponding to the GFT model in Fig. 7.a.

the memory bus is not taken into account, so the spare memories do not functionally depend on it. **4)** A new BE named PS appears to represent the state of the power supply.

Such differences determine particular effects in the GFT analysis process (Sec. 5.3) that did not emerge in the original case study; we will evidence them during the model analysis: we first detect and classify the modules: the CSMs are $\widehat{MS(k)}$, $\widehat{DA}$; the SSMs are $\widehat{CM}$, $\widehat{PU(i)}$, $\widehat{TE}$. In particular, $\widehat{PU(i)}$ is now a module because each of the events in $\widehat{PU(i)}$ contain i in its parameter set; $\widehat{PU(i)}$ is the only minimal SSM. Since the parameter set of the the root event $PU(i)$ is not empty, we need to perform the parameters simplification step (Step 5 in Sec. 5.3) obtaining the module $\widehat{PU}$ depicted in Fig. 8.b, where the root event PU is atomic. This step was not necessary in the analysis of the previous version of the model (Sec. 5.3).

Then, $\widehat{PU}$ is converted into SWN and its analysis returns that the probability of PU to be $true$ at time $t = 10000$ h is 3.095534E-2. So, in the GFT model, we replace $\widehat{PU(i)}$ with the BRE $PU(i)$ characterized by such probability. In this way, we obtain the GFT in Fig. 9.a, which is actually a PFT. In the previous version of the model (Sec. 5.3), we instead obtained a FT (Fig. 7.a). From the GFT in Fig. 9.a, we can generate the equivalent pBDD in Fig. 9.b; in Sec. 5.3, we generated a BDD instead (Fig. 7.b).

Two types of nodes are present in a pBDD: variables corresponding to BEs, and parametric boxes (pBox) [5]

time t	$Pr\{DA,t\}$	$Pr\{CM,t\}$	$Pr\{TE,t\}$	time t	$Pr\{DA,t\}$	$Pr\{CM,t\}$	$Pr\{TE,t\}$
4000 h	7.518959E-6	4.355595E-6	1.187452E-5	8000 h	2.417223E-5	1.981770E-5	4.398946E-5
6000 h	1.448120E-5	1.102637E-5	2.550741E-5	10000 h	3.660650E-5	2.966267E-5	6.626809E-5

Table 4: Probabilities of the minimal SSMs root events, and of the TE of the GFT in Fig. 5.

#proc. units	GFT #events	#sym. states	an. t. (sec.)	DFT #events	#ord. states	an. t. (sec.)	#proc. units	GFT #events	#sym. states	an. t. (sec.)	DFT #events	#ord. states
3	7	140	<1	16	1016	2	7	7	1180	12	32	954240
4	7	276	2	20	5936	14	8	7	1700	17	36	4858624
5	7	484	3	24	33504	113	9	7	2356	25	40	24137216
6	7	780	7	28	181952	790	10	7	3164	36	44	117484544

Table 5: The size of the module $\widehat{CM}$ of the GFT in Fig. 5, according to the number of processing units. The time to analyze the state space is obtained by an Intel CORE 2.5GHz cpu with 4GB of RAM.

corresponding to parametric subtrees or BREs, and containing an internal (p)BDD. In Fig. 9.b, the pBox PB_k is equivalent to the subtree $\widehat{DA}$ of the GFT in Fig. 9.a, while PB_i is equivalent to $\widehat{CM}$. The solution of the pBDD [5] in Fig. 9.b returns that the TE probability at time $t = 10000$ h is 1.034665E-4.

If we have to increase the number of processing units, we still have only to increase the cardinality of C_i, without any variation of the structure of the GFT and of the deriving SWN and pBDD.

6 Conclusions and future work

The aim of the paper is significantly improve the modeling power of FTs by presenting a single generalized FT formalism including and integrating Boolean gates, dynamic gates, the parametric form and the RB, previously proposed in separate formalisms (Sec. 2). For each primitive in GFT, we have provided its notation, semantics, parametric form, and application in the GFT model of the case study (Sec. 4). In particular, the semantics of each dynamic gate has been described also in case of integration with RBs; we took into account the possibility of contemporary input events for the PAND gate; we considered the probabilistic version of the FDEP gate (pFDEP). The GFT model (Fig. 5) of the case study (Sec. 3) is a case of combined use of all the primitives available in the GFT formalism, and could not be created using the FT, PFT, DFT, or RFT formalism (Sec. 2).

The solution process proposed for GFT models (Sec. 5) exploits the modular approach. Each step of the process has been described and then applied to the GFT model of the case study. In particular, in the modules detection step, the technique takes into account the presence of any primitive available in GFT, so it checks the structural, parametric, dynamic, and repair independence of subtrees. The parametric form is exploited **1)** at the modeling level, to reduce the size of the GFT models in terms of number of events; **2)** at the solution level, to reduce the size of the state space of modules, by resorting to SWNs. Both advantages have been successfully experimented by increasing the number of processing units in the case study. The alternative version of the case study (Fig. 8) shows: **1)** the application of the parameter simplification step for modules consisting of parametric subtrees; **2)** the use of pBDD.

The goals of this paper were already addressed in [15], but in a very preliminary way: only few combinations of primitives were considered; for instance, it was possible using a RB only if connected to the input events of a SP gate. So, the repair could not act on any component or subsystem, and the RB could not be combined with any gate. The parametric form was defined only for the gates SP and FDEP, and not for any gate and for the RB. The description of the notation and the semantics of the primitives was not generalized, but it was based and limited to very simple examples. The detection of modules did not take into account the presence of RBs, while FDEP and SEQ gate required the use of "dummy" output events, in order to correctly detect the structural independent subtrees. Finally, the use of (p)BDDs in the solution process was not considered.

Our intention in the future is solving GFT models by means of *Dynamic Bayesian Networks* (DBN) [11], with the advantage of computing also predictive and diagnostic measures conditioned on observations about the system or components state at particular times. The use of DBNs avoid the generation of the state space that is instead factorized in a set of dependent variables describing the system state. The way to map a DFT with

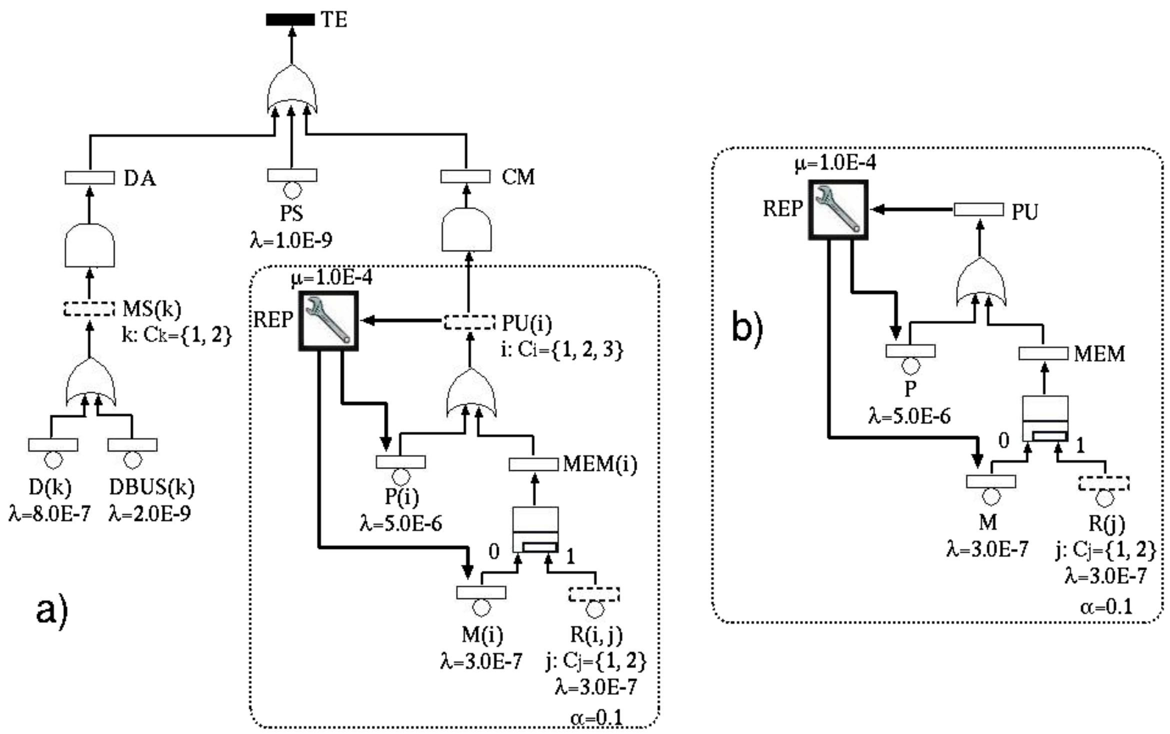

Figure 8: a) The GFT model of the alternative case study: the minimal SSM $\widehat{PU(i)}$ is put in evidence. b) The minimal SSM $\widehat{PU(i)}$ after the parameters simplification.

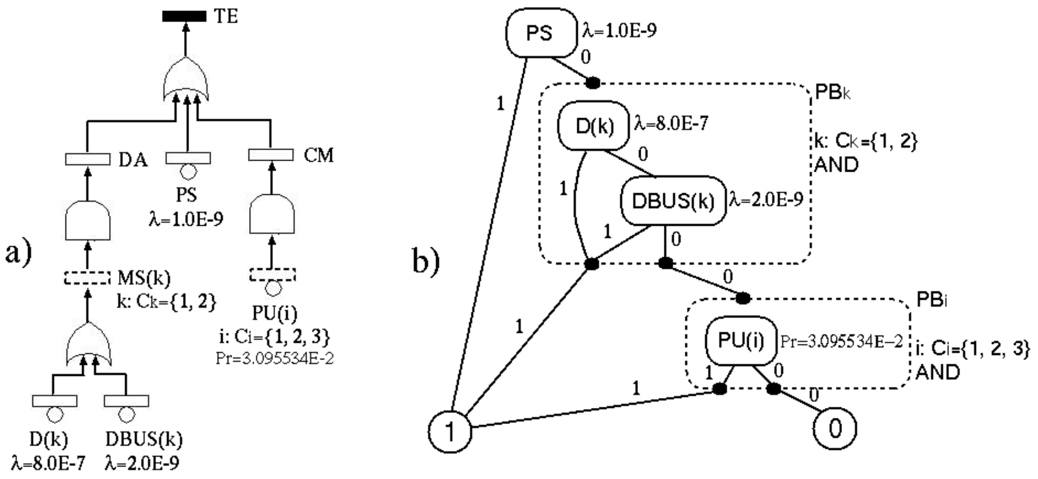

Figure 9: a) The GFT model in Fig. 8.a after the analysis of minimal SSMs. b) The pBDD corresponding to the GFT model in Fig. 9.a.

RBs, in the equivalent DBN, has been recently defined in [11]: we still need to find the way to deal with the parametric form. However, in DBNs the time is discretized, so the analysis results are approximated. Therefore, if we are interested in the exact value of the system unreliability, we still have to resort to the solution process described in this paper, where the time is continuous.

References

[1] R. A. Sahner, K. S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems; An Example-based Approach Using the SHARPE Software Package*. Kluwer Academic Publisher, 1996.

[2] A. Rauzy. New Algorithms for Fault Trees Analysis. *Reliability Engineering & System Safety*, 05(59):203–211, 1993.

[3] R. Gulati and J. B. Dugan. A modular approach for analyzing static and dynamic fault-trees. In *Proceedings of the Annual Reliability and Maintainability Symposium*, pages 57–63, 2003.

[4] A. Bobbio, G. Franceschinis, R. Gaeta, and G. Portinale. Parametric fault tree for the dependability analysis of redundant systems and its high-level Petri net semantics. *IEEE Transactions on Software Engineering*, 29(3):270–287, March 2003.

[5] D. Codetta-Raiteri. BDD based analysis of Parametric Fault Tress. In *Proceedings of the Annual Reliability and Maintainability Symposium*, pages 442–449, Newport Beach, CA USA, January 2006.

[6] A. Bobbio, G. Franceschinis, R. Gaeta, and G. Portinale. Dependability Assessment of an Industrial Programmable Logic Controller via Parametric Fault-Tree and High Level Petri Net. In *Proc. of the Int. Workshop on Petri Nets and Performance Models*, pages 29–38, Aachen, Germany, Sep. 2001.

[7] J. B. Dugan, S. J. Bavuso, and M. A. Boyd. Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems. *IEEE Transactions on Reliability*, 41:363–377, 1992.

[8] K. J. Sullivan, J. B. Dugan, and D. Coppit. The Galileo fault tree analysis tool. In *Proc. of the Annual Int. Symposium on Fault-Tolerant Computing*, pages 232–235, Madison, WI USA, June 1999.

[9] D. Codetta-Raiteri, G. Franceschinis, M. Iacono, and V. Vittorini. Repairable Fault Tree for the automatic evaluation of repair policies. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks*, pages 659–668, Florence, Italy, June 2004.

[10] F. Flammini, N. Mazzocca, M. Iacono, and S. Marrone. Using repairable fault trees for the evaluation of design choices for critical repairable systems. In *Proceedings of the IEEE International Symposium on High-Assurance Systems Engineering*, pages 163–172, Heidelberg, Germany, October 2005.

[11] L. Portinale, D. Codetta R., and S. Montani. Supporting Reliability Engineers in Exploiting the Power of Dynamic Bayesian Networks. *Int. Journal of Approximate Reasoning*, 51(2):179–195, Jan. 2010.

[12] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications. *IEEE Transactions on Computers*, 42:1343–1360, 1993.

[13] D. Codetta-Raiteri. *Extended Fault Trees Analysis supported by Stochastic Petri Nets*. PhD thesis, Dip. di Informatica, Univ. di Torino, Nov. 2005. http://www.di.unito.it/~phd/phd_theses.html.

[14] Y. Dutuit and A. Rauzy. A Linear-Time Algorithm to Find Modules of Fault Trees. *IEEE Transactions on Reliability*, 45:422–425, 1996.

[15] A. Bobbio and D. Codetta-Raiteri. Parametric Fault-trees with dynamic gates and repair boxes. In *Proc. of the Annual Reliability and Maintainability Symposium*, pages 459–465, Los Angeles, USA, Jan. 2004.