EPiC Series in Computing
Volume 73, 2020, Pages 354–372
LPAR23. LPAR-23: 23rd International
Conference on Logic for Programming,
Artificial Intelligence and Reasoning

A compositional semantics for Repairable Fault Trees with general distributions [*]

Raúl Monti[1], Carlos E. Budde[1], Pedro R. D'Argenio[2,3,4]

[1] University of Twente, Formal Methods and Tools, Enschede, the Netherlands
[2] Universidad Nacional de Córdoba, FAMAF, Córdoba, Argentina
[3] CONICET, Córdoba, Argentina
[4] Saarland University, Department of Computer Science, Saarbrücken, Germany

Abstract

Fault Tree Analysis (FTA) is a prominent technique in industrial and scientific risk assessment. Repairable Fault Trees (RFT) enhance the classical Fault Tree (FT) model by introducing the possibility to describe complex dependent repairs of system components. Usual frameworks for analyzing FTs such as BDD, SBDD, and Markov chains fail to assess the desired properties over RFT complex models, either because these become too large, or due to cyclic behaviour introduced by dependent repairs. Simulation is another way to carry out this kind of analysis. In this paper we review the RFT model with Repair Boxes as introduced by Daniele Codetta-Raiteri. We present compositional semantics for this model in terms of Input/Output Stochastic Automata, which allows for the modelling of events occurring according to general continuous distribution. Moreover, we prove that the semantics generates (weakly) deterministic models, hence suitable for discrete event simulation, and prominently for rare event simulation using the `FIG` tool.

1 Introduction

Fault Tree Analysis is a prominent technique for dependability assessment of complex industrial systems. Standard or *Static Fault Trees* (SFTs [41]) are directed acyclic graphs whose leaves are called Basic Events, and usually represent the failure of a physical system component. Each leaf is equipped with a failure rate or discrete probability, indicating the frequency at which the component breaks. The other FT nodes are called gates, and they model how basic components failures combine to induce more complex system failures, until the failure of interest (the *top event* of the tree) occurs. SFTs thus encode a logical formula. One of the most efficient analysis techniques uses Binary Decision Diagrams (BDD) to represent the formula, and then perform dependability studies using specialised algorithms. This assumes the absence of stochastic dependency among BEs.

[*] Partially supported by NWO project 15474 (*SEQUOIA*), EU project 102112 (*SUCCESS*), ERC grant 695614 (*POWVER*), ANPCyT PICT-2017-3894 (*RAFTSys*), and SeCyT project 33620180100354CB (*ARES*).

Many extensions to SFTs allow for further modelling capabilities. One of the most studied are *Dynamic Fault Trees* (DFTs [21, 25]). DFTs add gates to describe time- and order-dependence among the tree nodes, in contrast to the plain combinatorial behavior of SFT gates. New analysis methods were introduced in order to capture temporal requirements, such as cut sequences, translation to Markov models [21, 7], Sequence BDDs [23, 33, 44], algebraic approaches [29, 1], simulation, and combination and optimisations thereof [4, 24].

Repairable Fault Trees (RFT [3, 17, 2, 7]) increase FTs expressiveness by introducing the possibility to model complex interdependent repair mechanisms for basic components, i.e. system components that produce the basic events. In former models such as DFTs, certain notions of repair had been addressed by allowing components to be repaired independently. Nevertheless, this is not usual in real world systems, where repair scheduling, resources management, and maintenance play an important role. To address this, we will focus on the *Repair Box* model (RBOX [22, 17]). In the Repair Box model, the leaves of the tree are called Basic Elements (BEs) and represent basic system components that not only fail but can also be repaired. Each RBOX models a repair unit in charge of repairing certain BEs following a specific policy. Different repair policies such as *first come first serve*, *priority service*, or *uniform choice*, allow for analysing the impact of taking these decisions in the real system. The introduction of these boxes greatly changes the dynamic of the tree. Traditional quantitative analyses are no longer a combinatorial calculation, since the evolution of the system over time has to be considered [38]. The cyclic behavior introduced by this model disallows the use of combinatorial solutions proposed for non repairable FTs, and require a state based solution instead [4]. Furthermore, traditional qualitative analyses such as *cut sets* are also ruled out by not taking repairs into account, or requiring the memoryless property.

Considerations on RFT analysis and contribution. In this work we present a formal definition of *Repairable Fault Trees*, along with its semantics given in terms of Input/Output Stochastic Automata with Urgency (IOSA) [19, 20]. This allows us to model RFTs with general continuous failure and repair distributions. In Section 5 we formally define the syntax for RFTs in a similar manner as [6] has done for DFTs. Furthermore, in order to define compositional and weakly deterministic semantics using IOSA, we discuss different concerns about determinism on RFTs.

As discussed before, RFT analysis requires a state-space solution, which usually means one of the following two approaches. A first approach would be *to translate the model to a Markov model*, applying as much optimisations as possible during the modelling and analysis in order to relieve the state explosion problem as much as possible. This is the approach followed by many works such as [2, 4, 3]. Two main drawbacks can be pointed out on this approach. The first one is that no matter which existing optimisation methods are used, there is no guarantee that there will be a significant state space reduction in general models. This is a specially difficult situation in big and complex industrial size systems analysis involving repair. A second drawback is the restriction to exponentially distributed events, not allowing to correctly model real life systems where timing is governed by other continuous distributions. This is the case for example of phenomena such as timeouts in communication protocols, hard deadlines in real-time systems, human response times or the variability of the delay of sound and video frames (so-called jitter) in modern multi-media communication systems, which are typically described by non-memoryless distributions such as uniform, log-normal, or Weibull distributions [20].

A second approach to RFT analysis is to *recur to simulation*, which does not require to construct the full state space of the model, and does not impose *per se* restrictions to any kind of probabilistic distribution. The main problem when opting for simulation is the large amount of computation needed to reach a sufficiently accurate result. This issue is most relevant in the analysis of highly-dependable or fault-tolerant systems, where failure probabilities are very small and classical Monte Carlo simulation becomes infeasible. Rare event simulation techniques such as Importance Sampling or Importance Splitting exist to overcome this problem [42, 15, 34]. In this work we show that the underlying IOSA semantics of the RFT specification is *weakly deterministic*, that is, all non-determinism present in the IOSA model is spurious in the sense that any possible resolution of non-determinism leads to the same stochastic properties. In other words, metrics such as MTTF (mean time to failure) or availability are unaffected by such type of non-determinism. Hence the RFT model is equivalent to a fully stochastic model, amenable to discrete event simulation.

Our main contribution in this work consists in a method for precisely modelling RFTs with generally distributed events. In particular, achieving this is not straightforward since it requires to keep track of the event times and their evolution by means of clocks in the underlying semantics [20]. Furthermore, our approach is compositional and guarantees to deliver (weakly) deterministic IOSA models, hence amenable to discrete event simulation. This enables its analysis via the statistical model checker FIG [13] which specialises in rare event simulation. Our results provide the theoretical foundations on which the tool chain from [14] is based, where they were used for efficient dependability analyses of highly reliable systems.

Outline. After discussing related work, in Sec. 2 we describe the Repairable Fault Trees considered in this paper. In Sec. 3 we present the mathematical definition of the IOSA modelling language, as well as a symbolic language in Sec. 4 that permits defining such models in plain text files. The formal syntax for RFT and its semantics are first introduced in Sec. 5, and then extended in Sec. 7 to further consider spare components. The semantics introduced in Sec. 5 are proven weakly deterministic in Sec. 6. This works concludes in Sec. 9 after mentioning some practical applications in Sec. 8.

Related work. A variety of works address the problem of defining a rigorous syntax and semantics to FTs, DFTs, and RFTs [18, 7, 6, 3, 2, etc.]. They usually differ e.g. in the types and meaning of gates (their expressive power), how spare elements are claimed, and how repair races are resolved. The presence of non-deterministic situations is also a main discording issue. The work [26] presents a unifying GSPN semantics for the current flavours of DFT, although it does not include repair models nor general distributed failures which are central to our work. Comprehensive surveys on FTs can be found in [25] and [38].

Works based on I/O-IMC [7, 6] are closely related to ours. Even though they allow for more freedom in the definition of some gates, they are restricted to exponentially distributed events, which is the main limitation being overcome in our work. More specifically and just like us, the modelling framework and analysis mechanism in [7, 6] is compositional. However, a major difference is that I/O-IMC semantics allow non-determinism, while we exclude it by formally proving that our fault tree models are weakly deterministic. Unlike [6, 7], this allows us to use simulation, and more precisely rare event simulation, as an effective analysis technique for dependable systems reliability.

Modeling languages such as Figaro [12] and AltaRica [32] are general purpose languages that target the verification analysis of industrial projects. With our work we offer a domain specific language (an extended version of Galileo [14]) for RFT in contrast to the generality of these languages. Although [11] uses Figaro to analyse BMDPs, an extension of FTs parallel to DFT, our approach follows more closely to languages such as Modest [5], which stands on a robust mathematical theory. This is particularly relevant when dealing with the analysis of non-deterministic systems. In effect, we follow in our work a three-steps approach to the RFT analysis problem: (1) We employ Galileo as domain specific language for RFT description, which abstracts the user from a general purpose modelling language. (2) We use the simulation-amenable IOSA symbolic formalism as a semantic for the Galileo defined models. Furthermore we use this semantic to elaborate on the deterministic properties of our RFT models. (3) Finally we stand on the (mathematical) well defined semantics of the IOSA language in terms of NLMP [20].

Beyond the already mentioned works, recent advances in [37] take on the matter of using rare event simulation to analyse DFTs with complex repairs. Nevertheless, they are restricted to exponential and Erlang distributions. Furthermore their analysis is not compositional and performs breadth-first search over Markov models, thus suffering from a potential state space explosion.

2 Repairable Fault Trees

In Fig. 1 we depict the set of RFT elements considered in this work. Each of them has a set of inputs where children (sub-trees) are connected and, if applicable, also an output to propagate its own failure, repair, and other signals. Failure and repair propagation starts at the leaves of the fault tree, which are (spare) basic elements. When a (spare) basic element fails or gets repaired, it instantaneously propagates the event to its parent gates. The state of a gate changes based on the signals it receives from its children, and propagates its new state to its own parent gates. Thus, a proper combination and timing of 'fail' signals from its children may change the state of a gate to 'fail'. Analogously, a proper combination and timing of 'repair' signals may change the state back to operational. In a gate, a state change is instantaneously propagated as an output signal. This includes all signals, viz. not only fail and repair, but e.g. also in the case of an RBOX that outputs a "start repairing" signal to an input basic element.

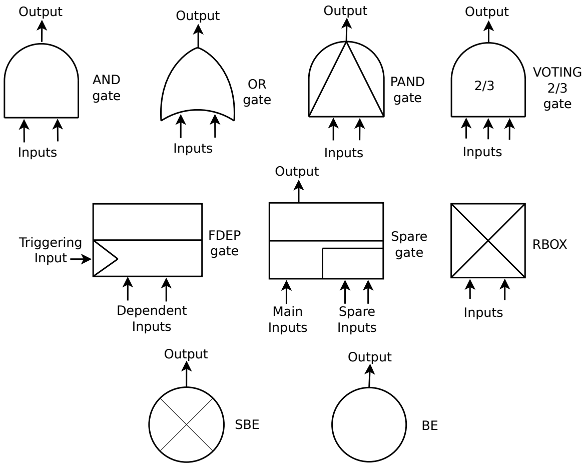


Figure 1: RFT elements

The exact combination of input signals (and their order) that makes a gate change its state depends entirely on its type. The intuition behind the behavior of each gate is as follows. An AND gate fails whenever all its inputs fail, and stops failing when at least one of its inputs is repaired. An OR gate fails whenever at least one of its inputs fails and stops failing when all of its inputs are repaired. A k/n VOTING gate fails whenever at least k of its n inputs fail and stops failing if at most $k-1$ of its inputs remain failed. A PAND gate fails whenever its inputs fail from left to right, inducing an order on the failure occurrence, and it stops failing when its last input is repaired. A *functional dependency gate* (FDEP) has $n+1$ inputs. The fail signal of one of its inputs (the triggering one) makes all the other inputs inaccessible to the rest of the system. Note that the dependent inputs do not necessarily fail, and they will be accessible again as soon as the triggering component is repaired (note the difference with [7, 36] where dependent BEs do fail). In fact this gate can be easily replaced by a system of OR gates [43]. A *spare basic element* (SBE) is a special case of BE which can be enabled and disabled, and can be used as spare parts for other BEs through spare gates. An SBE can be shared by several spare gates, and different sharing policies are introduced for this purpose. A *spare gate* (SG) can replace a basic element by one of several spare basic elements in case it fails. Each spare gate has a main input and n spare inputs. The main input must be a BE, and the spare inputs must be SBEs. As soon as the main input fails, the SG uses its own policy to ask for the replacement by one of its SBEs. The SG will fail whenever it does not obtain a replacement, and will signal repair whenever the main input gets repaired, or a spare input is obtained. If an in-use replacement fails, the SG will look for a new one. If the main input is repaired, the SG will release the acquired spare input, in case there is one. A *repair box* (RBOX) is the unit in charge of managing the repair of failed BEs and SBEs. An RBOX has n inputs, which are the elements administered for repairing, and a dummy output. An RBOX policy determines in which order the failing elements will be repaired. Also, an RBOX can only repair one of its inputs at a time, while the rest of its failed inputs wait for repair.

3 Input/Output Stochastic Automata

Input/Output Stochastic Automata (IOSA [19, 20]) provide a formal framework tailored to model stochastic systems for the purpose of simulation. IOSA combine continuous probability jumps from Stochastic Automata, with discrete event synchronisation for a compositional style of modelling. IOSAs use continuous random variables to control and observe the passage of time. These variables, called *clocks*, are set to a random value according to their associated probability distribution. As time evolves, all clocks count down at the same rate until they reach the value zero. Clocks control the moment when actions are taken, and thus allow to model systems where events occur at random continuous time points. Output and input transitions can be used to synchronize and communicate between different IOSAs. Output transitions are autonomous, while inputs occurrence depends on synchronisation with outputs. A transversal classification for actions allows to mark them as urgent or non-urgent. While a non-urgent output is controlled by the expiration of clocks (i.e. clocks reaching the value zero), an urgent output action is taken as soon as the state in which it is enabled is reached. Though an IOSA may be non-deterministic, [20] provides a set of sufficient conditions that guarantee weak determinism. That is, resolving non-deterministic choices leads to the same stochastically-enabled transitions—this will be made more precise with the concept of *confluence* in Def. 2. The existence of such set of sufficient conditions can be checked via an algorithm of polynomial complexity on the number of components in the model.

Definition 1. *An* input/output stochastic automaton with urgency *(IOSA) is a structure* $(\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow, C_0, s_0)$*, where* $\mathcal{S}$ *is a (denumerable) set of states,* $\mathcal{A}$ *is a (denumerable) set of labels partitioned into disjoint sets of* input *labels* $\mathcal{A}^{\mathsf{i}}$ *and* output *labels* $\mathcal{A}^{\mathsf{o}}$*, from which a subset* $\mathcal{A}^{\mathsf{u}} \subseteq \mathcal{A}$ *is marked as* urgent*,* $\mathcal{C}$ *is a (finite) set of clocks such that each* $x \in \mathcal{C}$ *has an associated continuous probability measure* μ_x *on* $\mathbb{R}$ *s.t.* $\mu_x(\mathbb{R}_{>0}) = 1$*,* $\rightarrow \subseteq \mathcal{S} \times 2^{\mathcal{C}} \times \mathcal{A} \times 2^{\mathcal{C}} \times S$ *is a transition function,* C_0 *is the set of clocks that are initialized in the initial state, and* $s_0 \in \mathcal{S}$ *is the initial state. In addition, an IOSA must satisfy the following constraints:*

(a) *If* $s \xrightarrow{C,a,C'} s'$ *and* $a \in \mathcal{A}^{\mathsf{i}} \cup \mathcal{A}^{\mathsf{u}}$*, then* $C = \varnothing$*.*

(b) *If* $s \xrightarrow{C,a,C'} s'$ *and* $a \in \mathcal{A}^{\mathsf{o}} \setminus \mathcal{A}^{\mathsf{u}}$*, then* C *is a singleton set.*

(c) *If* $s \xrightarrow{\{x\},a_1,C_1} s_1$ *and* $s \xrightarrow{\{x\},a_2,C_2} s_2$ *then* $a_1 = a_2$, $C_1 = C_2$ *and* $s_1 = s_2$.

(d) *For every* $a \in \mathcal{A}^{\mathsf{i}}$ *and state* s, *there exists a transition* $s \xrightarrow{\varnothing,a,C} s'$.

(e) *For every* $a \in \mathcal{A}^{\mathsf{i}}$, *if* $s \xrightarrow{\varnothing,a,C'_1} s_1$ *and* $s \xrightarrow{\varnothing,a,C'_2} s_2$, $C'_1 = C'_2$ *and* $s_1 = s_2$.

(f) *There exists a function* $active : \mathcal{S} \to 2^{\mathcal{C}}$ *such that:* (i) $active(s_0) \subseteq C_0$, (ii) $enabling(s) \subseteq active(s)$, (iii) *if* s *is stable,* $active(s) = enabling(s)$, *and* (iv) *if* $t \xrightarrow{C,a,C'} s$ *then* $active(s) \subseteq (active(t) \setminus C) \cup C'$.

where $enabling(s) = \{y \mid s \xrightarrow{\{y\},\_,\_} \_\}$, *and* s *is* stable *if there is no* $a \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\mathsf{o}}$ *such that* $s \xrightarrow{\varnothing,a,\_} \_$. *(_ indicates the existential quantification of a parameter.)*

The occurrence of an output transition is controlled by the expiration of clocks: $s \xrightarrow{C,a,C'} s$ indicates that there is a transition from state s to state s' that can be taken only when all clocks in C have expired and, when taken, it triggers the output action $a \in \mathcal{A}^{\mathsf{o}}$, and sets every clock in C' to a value sampled from its associated probability distribution. By (a), urgent output transitions are taken instantly. Besides, input transitions are only intended to take place if external outputs synchronize with them.

Restrictions (a) to (f) ensure that at most one non-urgent output action is enabled at a time. If in addition the IOSA is closed, i.e. all communication has been resolved and hence the set of inputs is empty, and all its urgent actions are confluent—in the sense of [30], see also Def. 2—then all the non-determinism is spurious [19, 20]. This means that it does not alter the stochastic behavior of the model, so regardless of how a non-deterministic choice is resolved, the stochastic properties remain the same. We call this property *weak determinism*. In particular for RFTs, metrics such as system reliability are unaffected by such non-determinism.

IOSAs are closed under parallel composition, which is defined according to rules (1)–(3) in Fig. 2. In order to avoid unintended behavior, the component IOSAs must be *compatible*, that is, they should not share output actions nor clocks, and they should be consistent with respect to urgent actions, i.e. $(\mathcal{A}_i \backslash \mathcal{A}^{\mathsf{u}}_i) \cap \mathcal{A}^{\mathsf{u}}_j = \varnothing$ [20].

$$\frac{s_1 \xrightarrow{C,a,C'}_1 s'_1}{s_1 \,||\, s_2 \xrightarrow{C,a,C'} s'_1 \,||\, s_2} \; a \in \mathcal{A}_1 \backslash \mathcal{A}_2 \qquad (1) \qquad\qquad \frac{s_2 \xrightarrow{C,a,C'}_2 s'_2}{s_1 \,||\, s_2 \xrightarrow{C,a,C'} s_1 \,||\, s'_2} \; a \in \mathcal{A}_2 \backslash \mathcal{A}_1 \qquad (2)$$

$$\frac{s_1 \xrightarrow{C_1,a,C'_1}_1 s'_1 \quad s_2 \xrightarrow{C_2,a,C'_2}_2 s'_2}{s_1 \,||\, s_2 \xrightarrow{C_1 \cup C_2,a,C'_1 \cup C'_2} s'_1 \,||\, s'_2} \; a \in \mathcal{A}_1 \cap \mathcal{A}_2 \qquad (3)$$

Figure 2: Parallel composition of IOSA

Together with the notion of parallel composition appears the notion of confluence. This is often exploited as a reduction technique due to the fact that, under some conditions, it does not alter the properties of the system. Such is also the case in our setting:

Definition 2. *An IOSA is* confluent *if for all pair of urgent actions* a *and* b, *and for every (reachable) state* s, *it satisfies that, if* $s \xrightarrow{\varnothing,a,C_1} s_1$ *and* $s \xrightarrow{\varnothing,b,C_2} s_2$, *then there is a state* s_3 *such that* $s_2 \xrightarrow{\varnothing,a,C_1} s_3$ *and* $s_1 \xrightarrow{\varnothing,b,C_2} s_3$.

Note that, according to this definition, the same state is reached regardless of the order of the confluent transitions. As mentioned earlier, this non-determinism is spurious in the sense that it does not alter the stochastic properties of the given IOSA, regardless of how it is resolved. Since non-determinism can only arise on urgent actions, we say that a *closed* IOSA is *weakly deterministic* if all its urgent actions are confluent.

4 IOSA symbolic language

We present a symbolic language to describe an IOSA model. This language is the input language of the simulation tool FIG and has some strong resemblance with the PRISM modelling language [27]. IOSAs compositional style of modelling is also reflected in the language, where each component is modeled separately by what we call a *module*. A module is composed of a set of variables, whose valuation represents the actual state of the component, a set of clocks corresponding to the enabling clocks for non-urgent transitions, and a set of transitions which symbolically describe the possible jumps between states (changes of valuations and resetting of clocks). Fig. 3 models an RFT basic element as an example. Variables can be of integer (with finite range) or boolean type. As we will see, also fixed-size arrays can be defined as variables. An initial value for each variable is determined after the keyword `init`. The probability distribution of a clock is defined at the transitions where it is reset. A transition is described by the name of the action which takes place, a guard that defines the origin states, an enabling clock (only for the case of non-urgent output transitions), a condition describing the target states, and the set of clocks to be reset. A quick overview of Fig. 3 will help to further understand our symbolic language: Two clocks, `fc` and `rc`, are defined in line 2. These clocks will be used as enabling clocks for transitions at lines 6 and 8, and reset on transitions at lines 7 and 9 where γ and μ are the distribution associated with `rc` and `fc`, respectively. Lines 3 and 4 define variables `inform` and `broken`, both of integer type ranging between 0 and 2, and initialized with value 0. Line 6 defines a set of non-urgent output transitions, which produce the output action *fl*. More precisely, this line defines the set of non-urgent transitions $s \xrightarrow{\{\texttt{fc}\},\mathit{fl}!,\varnothing} s'$, where s meets the condition `broken=0`, and s' is the result of changing the values of variables `inform` and `broken` to `1` while other variables remain with the same values as those in state s. The symbol `@` precedes the enabling clock for the transition. The symbol `->` distinguishes between conditions for the origin state on its left, and for the target state on its right. The conditions on the target state are expressed as assignments to the next values of the variables, indicated with an apostrophe.

```
 1  module BE
 2   fc, rc : clock;
 3   inform : [0..2] init 0;
 4   broken : [0..2] init 0;
 5
 6   [ fl ! ] broken=0 @ fc -> (inform'=1) & (broken'=1);
 7   [ r ??] broken=1 -> (broken'=2) & (rc'=γ);
 8   [up!] broken=2 @ rc -> (inform'=2) &
 9                          (broken'=0) & (fc'=μ);
10
11   [f !!] inform=1 -> (inform'=0);
12   [u!!] inform=2 -> (inform'=0);
13  endmodule
```

Figure 3: IOSA symbolic model of a Basic Element

Line 7 defines an urgent input transition with label *r*. Double question marks after the label name indicate that it describes an urgent input transition. Urgent output transition are indicated with double exclamation marks (`!!`), non-urgent input transitions with a single question mark, and non-urgent output transitions with a single exclamation mark. At the end of line 7 the clock `rc` is reset to a value sampled from a probability distribution γ. This line thus defines transitions $s \xrightarrow{\varnothing,r??,\{\texttt{rc}\}} s'$, where s satisfies the condition `broken=1` and s' is identical to s except for variable `broken` which has value `2`. At line 11, an urgent output transition is defined, indicating the failure of this component through action `f!!`. We will typically use these urgent transitions to synchronize among modules.

IOSA models are input-enabled: in our symbolic language, any input in any constraint that is not explicit, is implicitly completed with self-loops. For example in Fig. 3, the line "`[r??]  broken != 1 -> ;`" is assumed to exist, thus completing the case of line 7 to make the model input-enabled on the urgent input `r`.

5 A formal syntax for RFTs and its semantics

In this section we present a formal definition of the RFT (similar to those of [6, 8]), along with its semantics given in terms of IOSA. Each element of an RFT is characterized by a tuple consisting of its type, its arity (i.e. the number of inputs), and possibly other parameters like probability distributions for fail and repair events in a BE.

Definition 3. *Let $n, m, k \in \mathbb{N}^+$, and let μ, ν and γ be continuous probability distributions. We define the set $\mathcal{E}$ of elements of a RFT to be composed of the following tuples:*

- $(\mathsf{be}, 0, \mu, \gamma)$ *and* $(\mathsf{sbe}, 0, \mu, \nu, \gamma)$*, which represents basic and spare basic elements, with no inputs, with an active failure distribution μ, a dormant failure distribution ν, and a repair distribution γ.*
- (and, n)*,* (or, n) *and* (pand, n)*, which represent AND, OR and PAND gates with n inputs, respectively,*
- (vot, n, k)*, which represent a k from n voting gate,*
- (fdep, n)*, which represents a functional dependency gate, with 1 trigger input and $n-1$ dependent ones. By convention the first input is the triggering one.*
- (sg, n)*, which represents a SPARE gate with one main input and $n-1$ spare inputs. By convention the first input is the main one.*
- (rbox, n)*, which represents an RBOX element for n BEs (or SBEs).*

An RFT is a directed acyclic graph (DAG), for which every vertex v is given a label $l(v)$. An edge from v to w means that the output of v is connected to an input of w. Since the order of the inputs is relevant, we give them in terms of a list $i(w)$ instead of a set. Similarly, $si(v)$ will list all the spare gates to which a spare basic element v is connected as an input. Let $\mathrm{t}(v)$ indicate the *type* of v, and $\#(v)$ the number of inputs of v. Here, $\mathrm{t}(v)$ is the first projection of the label $l(v)$, and $\#(v)$ is the second projection of $l(v)$.

Definition 4. *A repairable fault tree is a four-tuple $T = (V, i, si, l)$, where V is a set of vertices, $l \colon V \to \mathcal{E}$ is a function labeling each vertex with an RFT element, $i \colon V \to 2^V$ is a function assigning $\#(v)$ inputs to each element v in V, and $si \colon V \to 2^V$ indicates which spare gates manage each SBEs. The set of edges $E = \{(v, w) \in V^2 \mid \exists j \cdot v = (i(w))[j]\}$ is the set of pairs (v, w) such that v is an input of w. If such an edge exists, we will say that v is connected to w and w to v. In addition, an RFT T should satisfy the following conditions:*

- *The tuple (V, E) is a DAG.*

- *T has a unique top element, i.e. a unique element whose non dummy output is not connected to another gate. That is, there is a unique vertex $v \in V$ such that for all $w \in V$, $(v, w) \notin E$ and $\mathrm{t}(v) \neq \mathsf{fdep}, \mathsf{rbox}$.*
- *FDEP and RBOX outputs are dummy: if $(v, w) \in E$ then $\mathrm{t}(v) \notin \{\mathsf{fdep}, \mathsf{rbox}\}$.*
- *An output can not be more than once the input of a same gate. That is, for all $0 \leq j, k \leq |i(w)| - 1$ with $i(w)[j] = i(w)[k]$, we have $j = k$.*
- *The inputs of a repair box can only be basic elements: if $(v, w) \in E$ and $\mathrm{t}(w) = \mathsf{rbox}$ then either $\mathrm{t}(v) = \mathsf{be}$ or $\mathrm{t}(v) = \mathsf{sbe}$.*
- *Each (spare) basic element can be connected to a single RBOX: if $(v, w) \in E$ and $(v, w') \in E$ and $\mathrm{t}(w) = \mathrm{t}(w') = \mathsf{rbox}$, then $w = w'$.*
- *The spare inputs of a spare gate can only be spare basic elements, while its main input can only be a basic element: if $\mathrm{t}(v) = \mathsf{sg}$ then $\mathrm{t}(i(v)[0]) = \mathsf{be}$ and for $j > 0$, $\mathrm{t}(i(v)[j]) = \mathsf{sbe}$.*
- *A spare basic element can only be connected to a spare gate or an RBOX: if $(v, w) \in E$ and $\mathrm{t}(v) = \mathsf{sbe}$ then $\mathrm{t}(w) \in \{\mathsf{sg}, \mathsf{rbox}\}$.*
- *A spare basic element is an input of a spare gate, if and only if that spare gate is spare input of the spare basic element: for v and v' such that $l(v') = (\mathsf{sbe}, 0, \mu, \nu, \gamma)$ and $l(v) = (\mathsf{sg}, n)$, $(v', v) \in E$ if and only if there exists j such that $v = si(v')[j]$.*
- *A basic element can be connected to at most one spare gate: if $(v, w) \in E$ and $(v, w') \in E$ with $\mathrm{t}(w) = \mathrm{t}(w') = \mathsf{sg}$ and $\mathrm{t}(v) = \mathsf{be}$ then $w = w'$.*
- *If a (spare)basic element is connected to a spare gate then it can not be connected to a FDEP gate: if $(v, w) \in E$ and $\mathrm{t}(v) \in \{\mathsf{be}, \mathsf{sbe}\}$ and $\mathrm{t}(w) = \mathsf{sg}$, then there is no $(v, w') \in E$ such that $\mathrm{t}(w') = \mathsf{fdep}$.*

In the following, we present a parametric semantics for RFT elements. This will be used later to obtain the semantics for each vertex in a given RFT, and the consequent semantics of the full model as a parallel composition of its components. In this section, we only give the semantics for BEs, AND gates, OR gates, PAND gates, and RBOX. Note that FDEP can be replaced by OR gates. Similarly, voting gates can be modeled by a series of AND and OR gates, although a simpler model can be found in an extended version of this work [31]. In the design of the IOSA modules we should take into account the communication of each element of an RFT with its children and parents. For instance a basic element has to communicate its failure and repair to all parent gates. Similarly, an RBOX must communicate to its inputs a *start repair* signal. In order to do so, the semantics of each element will be given by a function, which takes actions as parameters.

For *BE* element $e \in \mathcal{E}$, its semantics is a function $[\![e]\!] : \mathcal{A}^5 \to \mathit{IOSA}$, where $[\![(\mathsf{be}, 0, \mu, \gamma)]\!](\mathit{fl}, \mathit{up}, f, u, r)$ results in the IOSA shown in Fig. 4. The state of a basic element is defined by the fail clock `fc`, the repair clock `rc`, a variable `inform` that indicates when to signal the failure or repair, and variable `broken` to distinguish between broken and normal states. Note that at

```
module BE
 fc, rc : clock;
 inform : [0..2] init 0;
 broken : [0..2] init 0;

 [fl ! ] broken=0 @ fc -> (inform'=1) & (broken'=1);
 [r ??] broken=1 -> (broken'=2) & (rc'=γ);
 [up!] broken=2 @ rc -> (inform'=2) &
                          (broken'=0) & (fc'=μ);

 [f !!] inform=1 -> (inform'=0);
 [u!!] inform=2 -> (inform'=0);
endmodule
```

Figure 4: IOSA symbolic model of a Basic Element (same as Fig. 3)

the starting state of an IOSA module, all its clocks are set randomly according to their associated distributions. A basic element fails when clock `fc` expires (line 6) and immediately informs it with the urgent signal *f*!! at line 11. As soon as the repair begins by the corresponding connected repair box (communicated through *r*?? in line 7), clock `rc` is set. When it expires, the component becomes repaired. Hence, the end of the repair is communicated to the RBOX with *up*! and `fc` is set again at line 8. Immediately after, the repair is signaled with urgent action *u*!! at line 11. The three values of variable `broken` keep track of the current state: value `0` indicates that the BE is operational, `1` indicates that the BE is in a failed state, and `2` indicates that the BE is in the process of being repaired.

For an *AND gate* element with two inputs, its semantics is a function $[\![e]\!] : \mathcal{A}^6 \rightarrow \mathit{IOSA}$, where $[\![(\mathsf{and}, 2)]\!](f, u, f_1, u_1, f_2, u_2)$ results in the IOSA of Fig. 5. At lines 6 to 11, the AND gate gets informed of the failure of either of its inputs. Upon failure of some input, we distinguish between the case where the other input has already failed (`count=1`) and the case where it has not (`count=0`). In the first case the AND gate has to move to a failure state, for which we set the `informf` variable in order to enable the signaling of failure at line 20. Furthermore in both cases we increase the value of `count` so that we take note of the failure of an input. A similar reasoning is done for the case of the repairing of an input at lines 13 to 18. In this case we have to set the module to signal a repair when an input gets repaired at a state where both inputs were failing (lines 13 and 16), by enabling transition at line 21. From now on, we omit writing down self-loops originated by IOSA's input enabledness, such as lines 8, 11, 15 and 18 as they are assumed to be there. Nevertheless, we remark that it is necessary to take them into account when analyzing confluence in the next section. The semantics for an OR gate is similar to the AND gate and can be found in the extended version of this work [31].

```
 1 module AND
 2  informf: bool init false;
 3  informu: bool init false;
 4  count: [0..2] init 0;
 5
 6  [f1??] count=1 -> (count'=2) & (informf'=true);
 7  [f1??] count=0 -> (count'=1);
 8  [f1??] count=2 -> ;
 9  [f2??] count=1 -> (count'=2) & (informf'=true);
10  [f2??] count=0 -> (count'=1);
11  [f2??] count=2 -> ;
12
13  [u1??] count=2 -> (count'=1) & (informu'=true);
14  [u1??] count=1 -> (count'=0);
15  [u1??] count=0 -> ;
16  [u2??] count=2 -> (count'=1) & (informu'=true);
17  [u2??] count=1 -> (count'=0);
18  [u2??] count=0 -> ;
19
20  [f!!] informf & count=2 -> (informf'=false);
21  [u!!] informu & count!=2 -> (informu'=false);
22 endmodule
```

Figure 5: AND gate IOSA symbolic model.

The semantics of an *n inputs repair box with priority policy*, is a function $[\![(\mathsf{rbox}, n)]\!] : \mathcal{A}^{3*n} \rightarrow \mathit{IOSA}$, where $[\![(\mathsf{rbox}, n)]\!](\mathit{fl}_0, \mathit{up}_0, r_0, ..., \mathit{fl}_{n-1}, \mathit{up}_{n-1}, r_{n-1})$ results in the IOSA of Fig. 6. The RBOX with priority uses the array `broken[n]` to keep track of failed inputs, updating it when it receives their fail signals (lines 5 to 7) and up signals (lines 14 to 16). At the same time, when not busy, it sends repair signals to broken inputs (lines 9 to 12). Guards ensure the priority order for repairing. Note that

```
 1 module RBOX
 2  broken[n]: bool init false;
 3  busy: bool init false;
 4
 5  [fl0?] -> (broken[0]'=true);
 6  ...
 7  [fl_{n-1}?] -> (broken[n-1]'=true);
 8
 9  [r0!!] !busy & broken[0] -> (busy'=true);
10  ...
11  [r_{n-1}!!] !busy & broken[n-1] & !broken[n-2]
12                & ... & !broken[0] -> (busy'=true);
13
14  [up0?] -> (broken[0]'=false) & (busy'=false);
15  ...
16  [up_{n-1}?] -> (broken[n-1]'=false) & (busy'=false);
17 endmodule
```

Figure 6: RBOX with priority policy

instead of listening to the urgent output signals of the input BEs, it listens for the non-urgent actions of the transitions that trigger the failure or repair. This is done with the only purpose of facilitating the confluence analysis over this module. Other types of repair boxes can be modeled, by using different repairing policies—see [31].

The semantics of a *Priority AND gate* with 2 inputs is defined by $[\![(\mathsf{pand},2)]\!] : \mathcal{A}^6 \to \mathit{IOSA}$, where $[\![(\mathsf{pand},2)]\!](\mathsf{f},\mathsf{u},\mathsf{f}_0,\mathsf{u}_0,\mathsf{f}_1,\mathsf{u}_1)$ results in the IOSA of Fig. 7. PAND gates fail only when their inputs fail from left to right. This allows to condition the failure of a system not only to the failure of the subsystems but also to the ordering in which they fail. Literature is not always clear or even disagrees on what should be the behavior of the PAND gate in case both inputs fail at the same time [28, 18]. This situation arises in some constructions with AND and OR gates, or when the inputs of a PAND gate are connected to the same FDEP (see Fig. 8). Some proposals disallow these situations and discard them on early syntactic checks [36]. Some others assume a non-deterministic situation and find it important to analyze scenarios where the behavior is in fact unknown [6]. Other works decided that the PAND gate does not fail unless its inputs break strictly from left to right [7, 3]. Some others state that PAND gates also fail when both their inputs fail at the same time [18, 10, 9]. We opted for this last case, so PAND gates must identify whether time has passed between the occurrence of the failures, and act consequently. In the case where no time passes between the failure of the inputs, we consider that the order of failures does not matter: the gate invariantly fails. To identify if time has passed between the occurrence of the input failures, the model listens to any output actions, which indicate that a clock has expired. This is done by a special input action at line 8, which synchronizes with all non-urgent outputs, regardless the name of the action. Notice that there is only one scenario that we want to rule out, which is when the second input fails and then time passes without the first input failing too. This is in fact the case described by the guard of line 8. Furthermore, this transition moves to the "unbreakable" state, from which it can only go back when the second input (i.e., input 1) is fixed. In consequence, the failure of the gate

```
 1  module PAND
 2
 3   f0: bool init false;
 4   f1: bool init false;
 5   st: [0..4] init 0;  \\ up, inform fail, failed,
 6                       \\ inform up, unbreakable
 7
 8   [_?] st=0 & f1 & !f0 -> (st'=4);
 9
10   [f0??] st=0 & !f0 & !f1-> (f0'=true);
11   [f0??] st=0 & !f0 & f1 -> (st'=1) & (f0'=true);
12   [f0??] st!=0 & !f0 -> (f0'=true);
13   [f0??] f0 -> ;
14
15   [f1??] st=0 & !f0 & !f1 -> (f1'=true);
16   [f1??] st=0 & f0 & !f1 -> (st'=1) & (f1'=true);
17   [f1??] st=3 & !f1 -> (st'=2) & (f1'=true);
18   [f1??] (st==1|st==2|st=4) & !f1 -> (f1'=true);
19   [f1??] f1 -> ;
20
21   [u0??] st!=1 & f0 -> (f0'=false);
22   [u0??] st=1 & f0 -> (st'=0) & (f0'=false);
23   [u0??] !f0 -> ;
24
25   [u1??] (st=0|st=3) & f1 -> (f1'=false);
26   [u1??] (st=1|st=4) & f1 -> (st'=0) & (f1'=false);
27   [u1??] st=2 & f1 -> (st'=3) & (f1'=false);
28
29   [f!!] st=1 -> (st'=2);
30   [u!!] st=3 -> (st'=0);
31
32  endmodule
```

Figure 7: PAND gate.

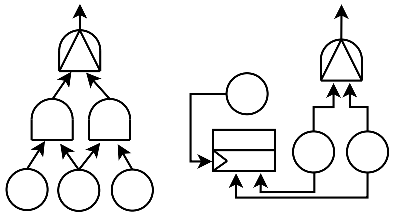


Figure 8: Spurious non-determinism in Dynamic Fault Trees with PANDs

occurs either if both inputs fail at the same time or if the first input fails, then time passes, and then the second input fails. Notice that an n-input PAND gate is simply a syntactic sugar for a system of $n-1$ two-input PAND gates connected in cascade.

The semantics of an RFT is that of the parallel composition of the semantics of its components, being conveniently synchronized.

Definition 5. *Given an RFT $T = (V, i, si, l)$ we define the semantics of T as $[\![T]\!] = ||_{v \in V} [\![v]\!]$ where $[\![v]\!]$ is defined by:*

$$[\![v]\!] = \begin{cases} [\![l(v)]\!](\mathsf{fl}_v, \mathsf{up}_v, \mathsf{f}_v, \mathsf{u}_v, \mathsf{r}_v) & \textit{if } l(v) = (\mathsf{be}, 0, \mu, \gamma) \\ [\![l(v)]\!](\mathsf{f}_v, \mathsf{u}_v, \mathsf{f}_{i(v)[0]}, \mathsf{u}_{i(v)[0]}, ..., \mathsf{f}_{i(v)[n-1]}, \mathsf{u}_{i(v)[n-1]}) & \\ & \textit{if } l(v) \in \{(\mathsf{and}, n), (\mathsf{or}, n)\} \\ [\![l(v)]\!](\mathsf{f}_v, \mathsf{u}_v, \mathsf{f}_{i(v)[0]}, \mathsf{u}_{i(v)[0]}, \mathsf{f}_{i(v)[1]}, \mathsf{u}_{i(v)[1]}) & \textit{if } l(v) = (\mathsf{pand}, 2) \\ [\![l(v)]\!](\mathsf{fl}_{i(v)[0]}, \mathsf{up}_{i(v)[0]}, \mathsf{r}_{i(v)[0]}, ..., \mathsf{fl}_{i(v)[n-1]}, \mathsf{up}_{i(v)[n-1]}, \mathsf{r}_{i(v)[n-1]}) & \\ & \textit{if } l(v) = (\mathsf{rbox}, n) \end{cases}$$

In Section 7, we extend the semantics to spare gates and spare basic elements.

6 RFTs are weakly deterministic

In this section we show that RFTs composed only by BEs, AND gates, OR gates, PAND gates, and RBOX, are weakly deterministic. Since voting and FDEP gates can be constructed using OR and AND gates, the result extends to these gates. Results in this section rely heavily on results about weak determinism on IOSA proven in [20]. Therefore, we first summarize the essentials of [20] relevant for this paper.

In [20], we provide sufficient conditions to ensure that a closed IOSA is weakly deterministic. This is stated in Theorem 1 below, which requires the following definition.

Definition 6. *Given an IOSA $\mathcal{I}$ with state space $\mathcal{S}$ and actions $\mathcal{A}$, we distinguish the following sets of actions:*

- *a set of urgent output actions $B \subseteq \mathcal{A}^{\mathsf{o}} \cap \mathcal{A}^{\mathsf{u}}$ is* initial *if each $b \in B$ is enabled in s_0, i.e. if for each $b \in B$ there is a state $s \in \mathcal{S}$ and $C \subseteq \mathcal{C}$, such that $s_0 \xrightarrow{\varnothing,b,C} s$;*
- *we say that a set $B \subseteq \mathcal{A}^{\mathsf{o}} \cap \mathcal{A}^{\mathsf{u}}$ of urgent output actions is* spontaneously enabled *by $b \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$ if there are stochastically reachable states $s, s' \in \mathcal{S}$ such that s is stable, $s \xrightarrow{\_,b,\_} s'$, and all actions in B are enabled in s';*
- *let $a \in \mathcal{A}^{\mathsf{u}}$ and $b \in \mathcal{A}^{\mathsf{o}} \cap \mathcal{A}^{\mathsf{u}}$. We say that a* triggers *b if there are stochastically reachable states $s_1, s_2, s_3 \in \mathcal{S}$ such that: $s_1 \xrightarrow{\_,a,\_} s_2$, $s_2 \xrightarrow{\_,b,\_} s_3$, and, if $a \neq b$, then there is no outgoing transition from s_1 labeled with b. The set $\{(a, b) \mid a \text{ triggers } b\}$ is called the* triggering relation*;*

where a state is stochastically reachable *if there is a path in the IOSA from the initial state that reaches such state with probability greater than zero.*

The *approximate indirect triggering relation* of a composite IOSA is defined as the reflexive transitive closure of the union of the triggering relations of its components. The following theorem from [20] gives necessary conditions for a closed IOSA not to be confluent. As a consequence, it provides sufficient conditions for a closed IOSA to be *weakly deterministic*.

Theorem 1 ([20])**.** *Given a closed composite IOSA $\mathcal{I} = (\mathcal{I}_1 \,||\, \dots \,||\, \mathcal{I}_n)$ with actions $\mathcal{A}$, if $\mathcal{I}$ is not confluent then there exists a pair of urgent actions $a, b \in \mathcal{A}^{\mathsf{u}}$ such that*

1. *one of the components is not confluent with respect to a and b,*
2. *there are actions c and d that approximately indirectly trigger both a and b, respectively, and*
3. *either (i) c and d are initial actions, or (ii) there exists an action e and possible empty sets B_1 to B_n spontaneously enabled by e in $\mathcal{I}_1$ to $\mathcal{I}_n$ respectively, such that c and d are in $\bigcup_{i=1}^{n} B_i$.*

In the following, we prove accessory propositions to eventually prove, using Theorem 1, that the IOSA defined by an RFT is weakly deterministic.

Proposition 1. *Let $[\![T]\!] = ||_{v \in V} [\![v]\!]$ as in Def. 5. Then:*

(a) if $\mathrm{t}(l(v)) = \mathsf{be}$ then the only spontaneously enabled sets in $[\![v]\!]$ are $\{\mathsf{f}_v\}$, spontaneously enabled by fl_v, and $\{\mathsf{u}_v\}$, spontaneously enabled by up_v;

(b) if $\mathrm{t}(l(v)) = \mathsf{rbox}$ then the only spontaneously enabled sets in $[\![v]\!]$ are of the form $\{\mathsf{r}_w\}$, spontaneously enabled by fl_w, and $\{\mathsf{r}_{w'}\}$, spontaneously enabled by up_w for $w \neq w' \in i(v)$;

(c) if $\mathrm{t}(l(v)) \notin \{\mathsf{be}, \mathsf{rbox}\}$ then $[\![v]\!]$ has no spontaneously enabled sets;

(d) for all $v \in V$, $[\![v]\!]$ has no initially enabled actions.

Proof sketch. Points *(a)*, *(b)* and *(c)* can be proved by just inspecting each model. For the case of the RBOX it is important to notice that the repair transitions are mutually exclusive: this implies that the sets which are spontaneously enabled by up actions are singleton sets. For the AND, OR, and VOT gates which fall under point *(c)*—all models can be found in an extended version of this work [31]—it is direct to check that there are no non-urgent actions in their corresponding models, and thus no spontaneously enabled actions either. In fact this corresponds to their combinatorial behavior which does not introduce extra delays. For point *(d)* it again suffices to walk through the models and realize that all urgent output transitions are disabled in the initial state. □

Proposition 2. *Let T be an RFT. The only possible pairs of non-confluent actions in $[\![T]\!]$ are $\{(\mathsf{f}_v, \mathsf{u}_{v'}) \mid v, v' \in i(w), t(w) \in \{\mathsf{and}, \mathsf{or}, \mathsf{pand}\}\} \cup \{(\mathsf{f}_w, \mathsf{u}_v), (\mathsf{u}_w, \mathsf{f}_v) \mid v \in i(w), t(w) \in \{\mathsf{and}, \mathsf{or}\}\}$.*

Proof. The proof of this Proposition follows an exhaustive check over each urgent transition of each model, in order to single out any non-confluent situation, and can be found in an extended version of this work [31]. □

Proposition 3. *Let T be an RFT. For each $v \in V$, the triggering relation of $[\![v]\!]$ is given by:*

- *$\{\}$, if $l(v) \in \{(\mathsf{be}, 0, \mu, \gamma), (\mathsf{rbox}, n)\}$,*
- *$\{(\mathsf{f}_v, \mathsf{f}_w) \mid v \in i(w)\} \cup \{(\mathsf{u}_v, \mathsf{u}_w) \mid v \in i(w)\}$, if $l(w) \in \{(\mathsf{and}, n), (\mathsf{or}, n)\}$, and*
- *$\{(\mathsf{u}_v, \mathsf{u}_w) \mid v = i(w)[1]\} \cup \{(\mathsf{f}_v, \mathsf{f}_w) \mid v \in i(w)\}$, if $l(w) = (\mathsf{pand}, 2)$.*

Proof sketch. It suffices to make a satisfiability analysis over guards and postconditions of each pair (t_a, t_b) with t_b an output urgent symbolic transition and t_a any urgent symbolic transition, taking into account only reachable states. □

Theorem 2. *Let T be an RFT. Then $[\![T]\!]$ is weakly deterministic.*

Proof. We look for a, b, c, d and e as well as sets B_i with $i = 1 \dots n$ as Theorem 1 suggests. From [20], Prop. 6, we can build the initially enabled set of a IOSA model as the union of the initially enabled sets of its components. Since Prop. 1 point *(d)* ensures that there are no initially enabled actions in the components of $[\![T]\!]$, c and d should be spontaneously enabled actions. Also by Prop. 1 points *(a)* and *(b)*, either e is of the form $\mathtt{fl}_v$ for some v and then $\bigcup_{i=1}^{n} B_i$ contains only fail actions (f_v for some v) and repair actions (r_v for some v), or e is of the form $\mathtt{up}_v$ for some v and then $\bigcup_{i=1}^{n} B_i$ contains only up actions (u_v for some v) and repair actions. Since repair actions do not trigger any action (see Prop. 3), in the first case we get that c and d are fail actions, while they are up actions in the second case. Furthermore, by Prop. 2, either a is of the form f_w for some w and b is of the form $\mathsf{u}_{w'}$ for some w' or the other way around. As shown by Prop. 3, fail actions only trigger fail actions and up actions only trigger up actions, thus it is impossible that c and d indirectly trigger a and b respectively. Therefore, it is not possible to find actions a, b, c, d, and e satisfying conditions 1 to 3 in Theorem 1, and hence $[\![T]\!]$ is confluent. Since $[\![T]\!]$ is also closed, then it is weakly deterministic. □

7 An extended semantics

In this section we add the spare gate and spare basic element to the semantics of RFTs. As before, we aim to guarantee that the IOSA model derived from the RFT is weakly deterministic. In order to do so, we need to bring special attention to two particular scenarios that could introduce non-determinism if they are incorrectly addressed.

The first scenario is given when a main basic element fails in a spare gate which is connected to several spare basic elements. At this point, the question arises of which among the available spare basic elements should be taken by the spare gate. Traditionally, spare elements are selected in order from an ordered set. To generalize this mechanism for the selection of the spares we intend to allow for more complex state-involved policies. It should be always the case that this policy is deterministic in its elections.

The second scenario arises when several spare gates have requested a broken or already taken SBE, which eventually gets fixed by a repair box or released by the owning spare gate. At this point, it is unclear which of the requesting spare gates will take the newly available SBE. For this, we define sharing policies on the SBE. Thus, to provide semantics to an SBE, we actually introduce two IOSA modules: one providing the extended behavior of a BE that can be taken from dormant to enabled state and vice versa, and another one, the *multiplexer* module, which manages the sharing of the SBE. Notice that this scenario is not a problem in the absence of repair boxes, since in such cases SBEs do not become available after they are taken or fail. Neither is this a problem when spare elements are not shared by different spare gates [3, 4]. The work [25] also studies race conditions in spare gates when two spare gates fail at the same time. Such situation is infeasible in our semantics, i.e. it only occurs with probability 0, because the next failure time of (different) components is sampled from continuous probability distributions.

The models for the spare gate, the spare basic element and the multiplexer can be found in an extended version of this work [31]. Cold, warm and hot spare basic elements are defined with a very simple variations between their models, only involving the assignment of a suitable probability distribution to the failure of the SBE in a dormant state (or discarding the failing clock and corresponding transition in the case of cold SBEs.). We extend the semantics of the RFT with the SBE and SG elements as follows.

Definition 7. *Given an RFT $T = (V, E)$, we extend Definition 5 with the following cases:*

$$[\![v]\!] = \begin{cases} \cdots \\ [\![l(v)]\!](\mathit{fl}_v, \mathit{up}_v, f_v, u_v, r_v, e_v, d_v, \mathit{rq}_{(si(v)[0],v)}, \mathit{asg}_{(v,si(v)[0])}, \\ \qquad \mathit{rel}_{(si(v)[0],v)}, \mathit{acc}_{(si(v)[0],v)}, \mathit{rj}_{(v,si(v)[0])}, .., \mathit{rj}_{(v,si(v)[n-1])}) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{if}\ l(v) = (\mathsf{sbe}, n, \mu, \nu, \gamma) \\ [\![l(v)]\!](f_v, u_v, \mathit{fl}_{i(v)[0]}, \mathit{up}_{i(v)[0]}, \mathit{fl}_{i(v)[1]}, \mathit{up}_{i(v)[1]}, \mathit{rq}_{(v,i(v)[1])}, \mathit{asg}_{(i(v)[1],v)}, \\ \qquad \mathit{acc}_{(v,i(v)[1])}, \mathit{rj}_{(i(v)[1],v)}, \mathit{rel}_{(v,i(v)[1])}, \ldots, \mathit{rel}_{(v,i(v)[n-1])}) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{if}\ l(v) = (\mathsf{sg}, n) \end{cases}$$

Notice that in the case of the SBE and SG, several signals are indexed by a pair of elements. This pair indicates which gate performs the action and which one listens for synchronisation. As an example, $\mathit{asg}_{(v,si(v)[0])}$ indicates that the multiplexer that manages v, assigns its spare basic element to its first connected spare gate ($si(v)[0]$).

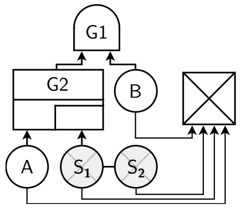
Unfortunately, we have not yet been able to find a general, easy or direct way to prove that this extension is indeed weakly deterministic, as we could do with the RFT without spares. This is due in part to the complexity of the IOSA modules, intended to avoid the aforementioned non-deterministic situations. While the spare basic element module can be easily proved to be confluent, this is not the case for the modules of the multiplexer and the spare gate. When analyzing these modules *in isolation* we find that some transitions are not confluent and Theorem 1 could not be used directly. However, by *partially composing* spare gates with multiplexers, we were able to check that conditions of Theorem 1 are not met: we automatically performed this check in several configurations, showing that they are indeed confluent.[1] As parallel composition preserves confluence, they can be inserted in other RFT contexts yielding weakly deterministic IOSAs.

8 Implementation and applications

The semantics of RFTs provided above is the backbone of the Fault Tree Analysis tool chain implemented in [14], where it was used to analyse unreliability and unavailability of highly resilient systems.

```
toplevel "G1";
"G1" and "G2" "B";
"G2" wsp "A" "S1" "S2";
"B" EXT_failPDF=rayleigh(6.0e-2) EXT_repairPDF=uniform(8,24);
"A" lambda=1.11e-3 EXT_repairPDF=normal(6,1);
"S1" lambda=0.2 EXT_dormPDF=erlang(9,0.3) EXT_repairPDF=normal(6,1);
"S2" lambda=0.2 EXT_dormPDF=erlang(9,0.3) EXT_repairPDF=normal(6,1);
"RB" repairbox_priority "B" "S2" "S1" "A";
```

(a) Extended Galileo textual format



(b) RFT described in Fig. 9a

Figure 9: Galileo description of RFTs for the tool chain from [14]

More in detail, the statistical model checker FIG[2] [13] runs simulations on IOSA models, to estimate the probability with which the model satisfies PCTL- and CSL-like property queries.

[1] Interested readers are referred to https://git.cs.famaf.unc.edu.ar/raulmonti/DeterminismScriptsRFT, where we offer a series of Python scripts that verify these configurations.

[2] FIG is open source software and is freely available in https://git.snt.utwente.nl/buddece/fig.

To provide FIG with the IOSA model corresponding to an RFT, the semantics introduced in this paper were implemented in a Java textual converter. The input to this converter is a file describing the RFT in an extended version of the Galileo textual format: a well-known syntax to describe Dynamic Fault Trees [39, 40]. Fig. 9a demonstrates this language to describe the RFT from Fig. 9b. The output of this converter is an IOSA model file according to the extended semantics presented in Sec. 7. FIG can (and has been used to) verify that the resulting IOSA file satisfies Def. 1 and is weakly deterministic. The application of our semantics to concrete case studies from the literature is demonstrated in [14], where numerous benchmarks are reported.

9 Conclusion

In this work we have defined a semantics for Dynamic Fault Trees with repair box in terms of Input/Output Stochastic Automata, introducing the novel feature of general probability distributions for failure and repairs of basic elements. Furthermore, we have shown that our semantics produces weakly deterministic models, which are hence amenable to discrete event simulation. In particular, our models described in the IOSA symbolic language serve as direct input to the statistical model checker FIG as well as other tools through the JANI model exchange format [16].

An interesting direction for future work is to introduce maintenance mechanisms and levels of degradation as in [35]. A further upgrade to our framework would be allowing *pairwise disjoint* sub-trees—i.e. that share no node—to act as primary and spare components, as well as supporting repairable gates [7]. Relaxing the disjoint condition on the sub-trees allowed as spares would make for a more ambitious extension: in this case, deeper analyses are needed to discard all non-deterministic behaviour. There is also the possibility of allowing multiple repair boxes to be assigned to one (shared) basic element. For this, it may well be necessary to introduce artifacts similar to the multiplexer that we defined for spare gates in Sec. 7. The reason for this is the need to resolve deterministically any situation where several repair boxes compete to repair the shared basic element. Regarding our extended semantics we see two lines of investigation: either a new model of spare gates must be developed, which can be proved deterministic in the general case; or a new proof scheme is required, to show that the present spare gate model is weakly deterministic in general.

We finally highlight the great potential of counting with a graphical interface to describe, modify, and analyse the RFTs. This would ease the modelling steps significantly, helping to visualise the trees and to interpret the result of experimentation with industrial-scale systems.

References

[1] Suprasad Amari, Glenn Dill, and Eileen Howald. A new approach to solve dynamic fault trees. In *RAMS 2003*, pages 374–379. IEEE, 2003.

[2] Marco Beccuti, Daniele Codetta-Raiteri, Giuliana Franceschinis, and Serge Haddad. Non deterministic repairable fault trees for computing optimal repair strategy. In *VALUETOOLS 2008*, page 56. ICST/ACM, 2008.

[3] Andrea Bobbio and Daniele Codetta-Raiteri. Parametric fault trees with dynamic gates and repair boxes. In *RAMS 2004*, pages 459–465. IEEE, 2004.

[4] Andrea Bobbio, Giuliana Franceschinis, Rossano Gaeta, and Luigi Portinale. Parametric fault tree for the dependability analysis of redundant systems and its high-level petri net semantics. *IEEE Trans. Software Eng.*, 29(3):270–287, 2003.

[5] Henrik C. Bohnenkamp, Pedro R. D'Argenio, Holger Hermanns, and Joost-Pieter Katoen. MODEST: A compositional modeling formalism for hard and softly timed systems. *IEEE Trans. Software Eng.*, 32(10):812–830, 2006.

[6] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. A compositional semantics for dynamic fault trees in terms of interactive markov chains. In *ATVA 2007*, volume 4762 of *LNCS*, pages 441–456. Springer, 2007.

[7] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. Dynamic fault tree analysis using input/output interactive markov chains. In *DSN 2007*, pages 708–717. IEEE Computer Society, 2007.

[8] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *IEEE Trans. Dependable Sec. Comput.*, 7(2):128–143, 2010.

[9] Hichem Boudali and Joanne B. Dugan. A discrete-time Bayesian network reliability modeling and analysis framework. *Rel. Eng. & Sys. Safety*, 87(3):337–349, 2005.

[10] Hichem Boudali and Joanne B. Dugan. Corrections on "A Continuous-Time Bayesian Network Reliability Modeling and Analysis Framework". *IEEE Trans. Reliability*, 57(3):532–533, 2008.

[11] Marc Bouissou and Jean-Louis Bon. A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes. *Reliability Engineering & System Safety*, 82(2):149–163, 2003.

[12] Marc Bouissou, N. Villatte, H. Bouhadana, and M. Bannelier. Knowledge modelling and reliability processing: presentation of the FIGARO language and associated tools. Technical report, Electricite de France (EDF), 1991.

[13] Carlos E. Budde. FIG: the Finite Improbability Generator. In *TACAS 2020*, to appear.

[14] Carlos E. Budde, Marco Biagi, Raúl E. Monti, Pedro R. D'Argenio, and Mariëlle Stoelinga. Rare Event Simulation for non-Markovian repairable Fault Trees. In *TACAS 2020*, to appear.

[15] Carlos E. Budde, Pedro R. D'Argenio, and Holger Hermanns. Rare event simulation with fully automated importance splitting. In Marta Beltrán, William J. Knottenbelt, and Jeremy T. Bradley, editors, *EPEW 2015*, volume 9272 of *LNCS*, pages 275–290. Springer, 2015.

[16] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. JANI: quantitative model and tool interaction. In Axel Legay and Tiziana Margaria, editors, *TACAS 2017*, volume 10206 of *LNCS*, pages 151–168, 2017.

[17] Daniele Codetta-Raiteri, Mauro Iacono, Giuliana Franceschinis, and Valeria Vittorini. Repairable fault tree for the automatic evaluation of repair policies. In *DSN 2004*, pages 659–668. IEEE Computer Society, 2004.

[18] David Coppit, Kevin J. Sullivan, and Joanne B. Dugan. Formal semantics for computational engineering: A case study on dynamic fault trees. In *ISSRE 2000*, pages 270–282. IEEE Computer Society, 2000.

[19] Pedro R. D'Argenio, Matias David Lee, and Raúl E. Monti. Input/Output Stochastic Automata - Compositionality and Determinism. In *FORMATS 2016*, volume 9884 of *LNCS*, pages 53–68. Springer, 2016.

[20] Pedro R. D'Argenio and Raúl E. Monti. Input/Output Stochastic Automata with Urgency: Confluence and Weak Determinism. In *ICTAC'18*, volume 11187 of *LNCS*, pages 132–152. Springer, 2018.

[21] Joanne B. Dugan, Salvatore J. Bavuso, and Mark A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3):363–377, 1992.

[22] Giuliana Franceschinis, Marco Gribaudo, Mauro Iacono, Nicola Mazzocca, and Valeria Vittorini. Towards an object based multi-formalism multi-solution modeling approach. In *MOCA 2002*, volume 26, pages 47–65, 2002.

[23] Daochuan Ge, Meng Lin, Yanhua Yang, Ruoxing Zhang, and Qiang Chou. Quantitative analysis

of dynamic fault trees using improved Sequential Binary Decision Diagrams. *Rel. Eng. & Sys. Safety*, 142:289–299, 2015.

[24] Rohit Gulati and Joanne B. Dugan. A modular approach for analyzing static and dynamic fault trees. In *RAMS 1997*, pages 57–63. IEEE, 1997.

[25] Sebastian Junges, Dennis Guck, Joost-Pieter Katoen, and Mariëlle Stoelinga. Uncovering Dynamic Fault Trees. In *DSN 2016*, pages 299–310. IEEE Computer Society, 2016.

[26] Sebastian Junges, Joost-Pieter Katoen, Mariëlle Stoelinga, and Matthias Volk. One Net Fits All - A Unifying Semantics of Dynamic Fault Trees Using GSPNs. In *PETRI NETS 2018*, volume 10877 of *LNCS*, pages 272–293. Springer, 2018.

[27] Marta Kwiatkowska, Gethin Norman, and Dave Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV 2011*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[28] Ragavan Manian, David Coppit, Kevin J. Sullivan, and Joanne B. Dugan. Bridging the gap between systems and dynamic fault tree models. In *RAMS 1999*, pages 105–111. IEEE, 1999.

[29] Guillaume Merle, Jean-Marc Roussel, Jean-Jacques Lesage, and Andrea Bobbio. Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events. *IEEE Trans. Reliability*, 59(1):250–261, 2010.

[30] Robin Milner. *Communication and concurrency.* PHI Series in computer science. Prentice Hall, 1989.

[31] Raul E. Monti, Pedro R. D'Argenio, and Carlos E. Budde. A compositional semantics for Repairable Fault Trees with general distributions. *arXiv e-prints*, October 2019. arXiv:1910.10507.

[32] Tatiana Prosvirnova, Michel Batteux Batteux, Pierre-Antoine Brameret, Abraham Cherfi, Thomas Friedlhuber, Jean-Marc Roussel, and Antoine Rauzy. The AltaRica 3.0 Project for Model-Based Safety Assessment. In *DCDS 2013*, 2013.

[33] Antoine Rauzy. Sequence Algebra, Sequence Decision Diagrams and Dynamic Fault Trees. *Rel. Eng. & Sys. Safety*, 96(7):785–792, 2011.

[34] Gerardo Rubino and Bruno Tuffin. *Rare Event Simulation Using Monte Carlo Methods.* Wiley Publishing, 2009.

[35] Enno Ruijters, Dennis Guck, Peter Drolenga, and Mariëlle Stoelinga. Fault maintenance trees: reliability centered maintenance via statistical model checking. In *RAMS 2016*, pages 1–6. IEEE, 2016.

[36] Enno Ruijters, Daniël Reijsbergen, Pieter-Tjerk de Boer, and Mariëlle Stoelinga. Rare event simulation for dynamic fault trees. In *SAFECOMP 2017*, volume 10488 of *LNCS*, pages 20–35. Springer, 2017.

[37] Enno Ruijters, Daniël Reijsbergen, Pieter-Tjerk de Boer, and Mariëlle Stoelinga. Rare event simulation for dynamic fault trees. *Reliability Engineering & System Safety*, 186:220–231, 2019.

[38] Enno Ruijters and Mariëlle Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15:29–62, 2015.

[39] Kevin J. Sullivan and Joanne B. Dugan. Galileo user's manual & design overview. `https://www.cse.msu.edu/~cse870/Materials/FaultTolerant/manual-galileo.htm`, 1998. v2.1-alpha.

[40] Kevin J. Sullivan, Joanne B. Dugan, and David Coppit. The Galileo fault tree analysis tool. In *29th Annual International Symposium on Fault-Tolerant Computing (Cat. No.99CB36352)*, pages 232–235, 1999.

[41] William Vesely, Francine Goldberg, Norman Roberts, and David Haasl. Fault Tree Handbook. Technical Report NUREG-0492, US Nuclear Regulatory Commission, Washington DC, 1981.

[42] Manuel Villén-Altamirano and José Villén-Altamirano. The rare event simulation method RESTART: efficiency analysis and guidelines for its application. In *Network Performance Engineering - A Handbook on Convergent Multi-Service Networks and Next Generation Internet*, volume 5233 of *LNCS*, pages 509–547. Springer, 2011.

[43] Liudong Xing, Joanne B. Dugan, and Brock A. Morrissette. Efficient reliability analysis of sys-

tems with functional dependence loops. *Eksploatacja I Niezawodnosc-Maintenance and Reliability*, (3):65–69, 2009.

[44] Liudong Xing, Akhilesh Shrestha, and Yuanshun Dai. Exact combinatorial reliability analysis of dynamic systems with sequence-dependent failures. *Rel. Eng. & Sys. Safety*, 96(10):1375–1385, 2011.