Integrating Satisfiability Solving in the Assessment of System Reliability Modeled by Dynamic Fault Trees

Margaux Duroeulx[1,2], Nicolae Brînzei[1], Marie Duflot[2], Stephan Merz[2]

[1] *University of Lorraine, CNRS, CRAN, F-54000 Nancy, France.*

[2] *University of Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France. email: margaux.duroeulx@loria.fr*

Fault trees (FTs) and their extensions are diagrammatic formalisms that are commonly used for reliability assessment and that represent the structure function of systems. The structure function determines tie sets and cut sets, and minimal tie sets are instrumental for assessing systems reliability. In a previous paper, we used satisfiability (SAT) techniques to compute tie sets from the structure function. In this paper we define minimal tie sets with sequences (MTSSs) as an extension of minimal tie sets for analyzing dynamic fault trees (DFTs), and we extend our previous techniques for computing MTSSs. We illustrate our approach using a standard case study and assess its performance over several industrial-size benchmarks.

Keywords: reliability, satisfiability, structure function, dynamic fault trees, tie sets, tie sets with sequences.

1. Introduction

While SFTs (standard, or static, FTs) provide intuitive diagrammatic representations for reliability assessment, they lack in expressivity for modeling spare management, dependent events, and different operational modes (Ruijters and Stoelinga (2015)). In order to take into account this type of phenomena, several extensions have been developed, such as dynamic fault trees (DFTs). DFTs contain standard gates like AND, OR and KooN (K-out-of-N); but also dynamic gates, such as FDEP, PAND and SPARE. Each standard gate (AND, OR, KooN) is associated with a Boolean expression that defines how failures located at the inputs of these gates propagate. The Boolean expression associated with the gate at the root of the fault tree is called the structure function; it indicates under which conditions the system fails.

Systems represented by DFTs can be simulated and analyzed using Monte Carlo techniques but the computational effort is high if tight confidence intervals are required. Boudali et al. (2007) consider representing DFTs as interactive Markov chains in order to make existing techniques available for simulating and analyzing DFTs. Zhu et al. (2014) study a stochasic approach for analyzing DFTs, but they only consider PAND gates. Xing et al. (2011) introduce a combinatorial technique based on sequential BDDs to compute the dynamic analogue of cut sets and from there derive quantitative analyses, and Ge et al. (2015) refine this work.

In a previous paper (Duroeulx et al. (2017)) we presented algorithms based on Boolean satisfiability solving to compute the minimal tie sets (MTSs) of SFTs. A tie set, also called a path set, is a set of system components whose simultaneous functioning ensures that the system functions properly.

The main contributions of the present paper are the definition of the structure function represented by a DFT and the extension of the SAT techniques for analyzing this function by introducing the analogue of MTSs for dynamic failures, namely minimal tie sets with sequences (MTSSs).

Section 2 describes in details the fundamental concept of a structure function corresponding to a system modeled by a DFT, and Section 3 inductively defines the structure function for a DFT. Section 4 introduces MTSSs and develops our approach for computing MTSSs, based on SAT solving. Section 5 illustrates our approach by applying it to a cardiac assist system and evaluates its performance using benchmark applications from the electrical and transport industries, while Section 6 concludes the paper with some perspectives for future work.

2. Structure functions of systems

An FT consists of a hierarchy of gates indicating how failures of components, represented as the leaves of the FT, propagate to system failures. In the remainder of this paper, the term "gate" will include the leaves of the FT corresponding to the system components. SFTs are appropriate for representing systems where the order of failures is irrelevant. However, certain systems may function or fail depending on the order in which component failures occur. For example, consider a switch for activating a backup component in case the main component fails. If the switch fails before the failure of the main component, the failure of the latter causes a system failure; if the switch

Proceedings of the 29th European Safety and Reliability Conference.
Edited by Michael Beer and Enrico Zio

ISBN: 981-973-0000-00-0 :: doi: 10.3850/981-973-0000-00-0_final

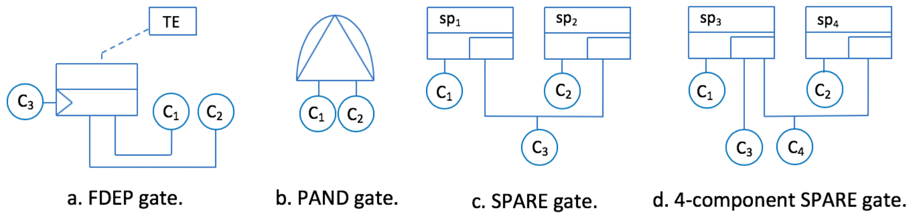

Fig. 1. Dynamic gates.

fails later, it does not. DFTs were introduced for modeling such situations, and this article focuses on them.

We consider (non-repairable) coherent systems with a binary behavior where components and systems can be functioning or failed. We associate a Boolean variable with every gate of the DFT to indicate if a failure of the subsystem corresponding to the gate has occurred.

Given the fault tree representing a system, we associate two variables with each gate γ of the fault tree: the Boolean variable c_γ of a gate is true if and only if the subsystem corresponding to the gate failed. The time-valued variable t_γ indicates when the failure of γ occurred; its value is irrelevant as long as γ functions.

The *structure function* $f : \zeta \to \{0,1\}$, where ζ is the set of configurations, associates to a configuration the state of the system. The structure function of a coherent system is monotonous: a component failure cannot make a failed system function again.

For SFTs, a configuration is simply given by the failure state of each component. In contrast, for DFTs, the configuration records the failure state of each component as well as the time of their failure.

3. Computing the structure function of dynamic fault trees

Whereas the variables c_γ and t_γ for a leaf of a FT are unconstrained, the variables for inner gates relate to the variables for their child gates according to formulas c_γ and t_γ, which we now define for each gate. In these expressions, the Boolean and time variables corresponding to a component C_i are identified as c_i and t_i, respectively.

3.1. *Static gates*

We consider three kinds of ordinary ("static") gates of fault trees, corresponding to conjunction, disjunction, and K-out-of-N. Figure 2 contains the graphical representation of these gates.

AND gate. The conjunction gate *AND* fails when both subsystems fail. The structure function therefore corresponds to conjunction and the timing function to the maximum:

$$c_{AND} = c_1 \wedge c_2 \qquad t_{AND} = \max(t_1, t_2) \qquad (1)$$

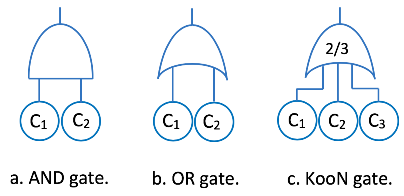

Fig. 2. Static gates.

OR gate. Dually, the disjunction gate *OR* fails when at least one of its subsystems fails. Its structure function corresponds to disjunction and the timing function to the minimum:

$$c_{OR} = c_1 \vee c_2 \qquad t_{OR} = \min(t_1, t_2) \qquad (2)$$

KooN gate. The gate *2-out-of-3* (2oo3) fails when at least 2 of its 3 components fail. Although it could be defined as a combination of AND and OR gates, it is convenient to consider it as a primitive gate. The structure and timing functions of the gate are:

$$c_{2oo3} = (c_1 \wedge c_2) \vee (c_1 \wedge c_3) \vee (c_2 \wedge c_3) \qquad (3)$$

$$t_{2oo3} = \min\big(\max(t_1, t_2), \max(t_1, t_3), \max(t_2, t_3)\big) \qquad (4)$$

We have only shown the formulas for binary or ternary instances of these gates, but their generalizations to n-ary gates are straightforward.

3.2. *Dynamic gates*

Since the order of failures is important for understanding the behavior of dynamic gates, we use chronograms to illustrate them. A low value (0) indicates proper functioning of a gate and a high value (1) means that it has failed. Again, we describe the formulas associated to gates using examples.

FDEP gate. The Functional-Dependency gate (*FDEP*) of Fig. 1.a indicates that the failure of the component C_3 on the left of the gate induces failures of the components C_1 and C_2 that depend on C_3.

On the left-hand chronogram of Figure 3, components C_1 and C_2 fail before the component C_3.

In this case, the failure of C_3 is irrelevant. On the right-hand chronogram, C_1 fails first and is followed by the failure of C_3, which induces the failure of C_2.

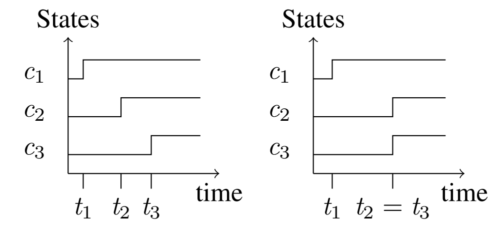

Fig. 3. Chronograms of the FDEP gate.

We formally represent an FDEP gate by adding constraints to the representation of the system in the form of implications for the Boolean-valued structure function and inequalities for the timing variables corresponding to the components. These constraints formally express the dependencies of failures. Note that we do not need to introduce expressions c_γ or t_γ for the gate itself.

$$c_3 \Rightarrow c_1 \qquad t_1 \leq t_3 \tag{5}$$
$$c_3 \Rightarrow c_2 \qquad t_2 \leq t_3 \tag{6}$$

PAND gate. The Priority-AND (*PAND*) gate, shown in Figure 1.b, was defined by Vesely et al. (1981). The gate fails when all the input events of the gate occur in the order from left to right. As illustrated in Fig. 4, a PAND gate with two child gates C_1 and C_2 fails when C_1 and C_2 fail in that order. If C_2 fails before C_1, the gate does not fail.

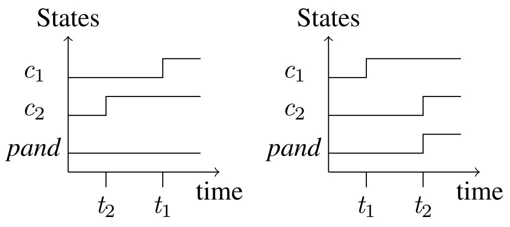

Fig. 4. Chronograms of the PAND gate.

The structure and timing functions corresponding to a binary PAND gate are therefore defined as follows:

$$c_{pand} = c_1 \wedge c_2 \wedge t_1 < t_2 \tag{7}$$
$$t_{pand} = t_2 \tag{8}$$

SPARE gate. System designers frequently introduce redundant (or spare) components that are initially in standby mode and that are activated when the main component fails. We assume that spare gates representing such redundancies can only have components as input events and no (proper) gates. However, several spare gates may share components, and a spare component that is used by one gate becomes unavailable for the other gates.

The literature on DFTs distinguishes cold, warm, and hot spare gates. The spare components of a hot spare gate may fail in standby mode, whereas the spare components of a cold spare gate only fail when they are actually used. The remainder of this section introduces the formulas for hot spare gates. We represent cold spare gates by adding a timing constraint to represent the fact that the spare component cannot fail before at least one of the main components (see Section 5). Warm spare gates are similar to hot spare gates, but the probabilities of failure of spare components are different in standby and active mode. For the purposes of this paper, they are modeled like hot spares: the difference appears only in the quantitative analysis.

When a spare gate does not share components with any other spare gate, its behavior is identical to an AND gate: it fails when all its child gates have failed. Now consider the gates sp_1 and sp_2 of Fig. 1.c that share the spare component C_3. Their behavior is illustrated by the chronograms of Fig. 5. On the chronogram shown on the left, the component C_2 fails first, and component C_3 will be used to replace it. Subsequently, component C_1 fails, causing sp_1 to fail since there is no more spare component to replace C_1. Similarly, the failure of C_2 leads to that of sp_2. The opposite situation would be symmetrical. The chronogram on the right illustrates the situation where the spare component fails first and then each component failure would automatically lead to the failure of its gate, since no more spare component is available.

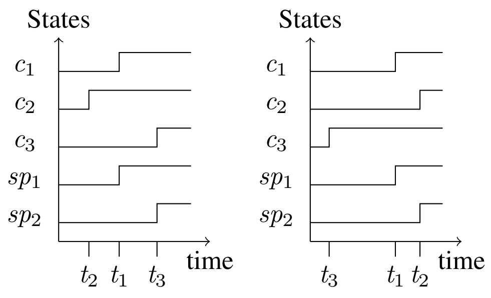

Fig. 5. Chronograms of the SPARE gates of Fig. 1.c.

The behavior of the gate sp_1 of Fig. 1.c is described by the following equations:

$$\begin{aligned} c_{sp_1} &= (c_1 \wedge c_3) \vee (c_1 \wedge c_2 \wedge t_2 < t_1) \\ &= c_1 \wedge (c_3 \vee (c_2 \wedge t_2 < t_1)) \end{aligned} \tag{9}$$

$$t_{sp_1} = \begin{cases} \max(t_1, t_3) & \text{if } t_1 < t_2 \\ t_1 & \text{otherwise} \end{cases} \tag{10}$$

Figure 1.d. shows another example of a SPARE gate. Here, C_3 is a spare component that exclusively replaces C_1. The second spare component C_4 is shared by both gates. Possible behaviors of sp_3 and sp_4 are illustrated in the chronograms of Fig. 6. On the left-hand side, component C_2 fails first, causing it to be replaced by C_4. Next, C_1 fails, and C_3 is used to replace it. The failures of C_3 and C_4 cause those of sp_3 and sp_4, respectively. The right-hand figure illustrates a different sequence of failure events.

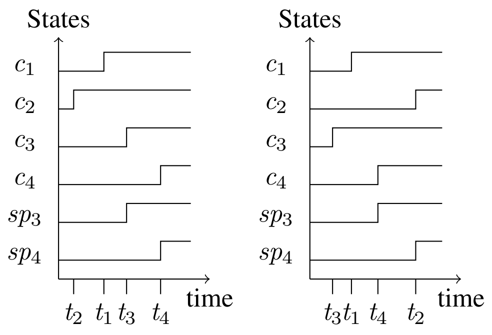

Fig. 6. Chronograms of the SPARE gate of Fig. 1.d.

The structure and timing functions corresponding to gate sp_3 are

$$\begin{aligned} c_{sp_3} = \quad & c_1 \wedge c_3 \\ & \wedge\ c_4 \vee (c_2 \wedge t_2 < \max(t_1, t_3)) \end{aligned} \tag{11}$$

$$t_{sp_3} = \begin{cases} \max(t_1, t_3, t_4) & \text{if } \max(t_1, t_3) < t_2 \\ \max(t_1, t_3) & \text{otherwise} \end{cases} \tag{12}$$

The corresponding functions for gate sp_4 are:

$$\begin{aligned} c_{sp_4} = c_2 \wedge (c_4 \vee \\ c_1 \wedge c_3 \wedge \max(t_1, t_3) < t_2) \end{aligned} \tag{13}$$

$$t_{sp_4} = \begin{cases} t_2 & \text{if } \max(t_1, t_3) < t_2 \\ \max(t_2, t_4) & \text{otherwise} \end{cases} \tag{14}$$

In order to derive these expressions for spare gates, it is useful to identify the conditions and times at which the components connected to a gate become unusable for that gate. For example, the gate sp_3 fails when it cannot use any of the components C_1, C_3, and C_4. Since C_1 and C_3 are not connected to another spare gate, they are unusable iff they fail. Component C_4 becomes unusable for sp_3 if it fails, but also if it is used by gate sp_4. The latter holds iff component C_2 fails before both of C_1 and C_3 have failed, and this motivates equation (11).

Eliminating time variables. Combining the expressions corresponding to the individual gates of a DFT, we obtain the structure function for the entire system. This function is expressed in terms of the Boolean variables c_γ that model failure of gates and the time variables t_γ that represent when those failures occur. In order to use a SAT solver for analyzing the structure function, we now show how the variables t_γ can be eliminated.

The key idea is to observe that time variables only occur in comparisons $e_\alpha < e_\beta$ where the expressions e_α, e_β are built from time variables by applying minima and maxima. The latter operators can be removed by applying the following equivalences:

$$\begin{aligned} \min(e_\alpha, e_\beta) < e_\gamma &\Leftrightarrow e_\alpha < e_\gamma \vee e_\beta < e_\gamma \\ \max(e_\alpha, e_\beta) < e_\gamma &\Leftrightarrow e_\alpha < e_\gamma \wedge e_\beta < e_\gamma \\ e_\gamma < \min(e_\alpha, e_\beta) &\Leftrightarrow e_\gamma < e_\alpha \wedge e_\gamma < e_\beta \\ e_\gamma < \max(e_\alpha, e_\beta) &\Leftrightarrow e_\gamma < e_\alpha \vee e_\gamma < e_\beta \end{aligned}$$

We can now introduce Boolean variables $t_{\alpha,\beta}$ to replace the inequalities $t_\alpha < t_\beta$ that occur in the definition of the structure function and thus obtain a purely propositional representation of the structure function. In order to minimize the number of variables and thus help speed up the analysis (see Section 4), we fix an order on gates. We only keep variables $t_{\alpha,\beta}$ where $\alpha < \beta$ and represent the inequalities $t_\alpha < t_\beta$ and $t_\beta < t_\alpha$ by $t_{\alpha,\beta}$ and $\neg t_{\alpha,\beta}$, respectively.

Prior to this transformation, the conditional equations governing time variables of spare gates can be eliminated by considering them as disjunctions of cases within the formulas in which these variables appear.

4. Analyzing the structure function.

Satisfiability (SAT) focuses on finding assignments of Boolean variables such that a formula of propositional logic is true (Biere et al. (2009)). SAT solvers employ efficient algorithms to compute a model, i.e., an assignment that satisfies the formula, if such an assignment exists, and return “unsat” otherwise. We now describe how we use SAT techniques for computing minimal tie sets of a structure function. A tie set is a set of components whose proper functioning ensures that the system works. Dually, a cut set is a set of components whose failure leads to the failure of the system. A cut set or tie set is minimal when the removal of any component from the set results in a set that is no longer a cut set or a tie set. Our objective is to compute all assignments representing minimal tie sets because they represent the critical configurations of the system, and they are sufficient for the assessment of the system reliability (Brînzei and Aubry (2018)).

The definition of the structure function corresponding to a fault tree given in Section 3 associates the Boolean expression c_α with the failure of gate α. A model of such an expression therefore corresponds to a cut set. Since tie sets are dual to cut sets, we will consider the dual of the structure function, corresponding to the complement

$$\bar{f} = \neg f$$

of the structure function in which all literals (i.e., Boolean variables or their negations) built from c_γ or $t_{i,j}$ have been replaced by their complements.

The models of the dualized structure function correspond to tie sets.

Since the structure function of a dynamic fault tree contains Boolean variables describing failure states as well as the temporal order between failures, we adapt the notion of tie set for dynamic fault trees, to take these temporal constraints into account.

We thus define *tie sets with sequences* (TSS) as tie sets to which we add, when appropriate, Boolean variables describing the temporal order between components failure. For a given TSS $\{x_1, \ldots, x_m, l_{i_1,i_2}, \ldots, l_{i_{n-1},i_n}\}$ (where $l_{\alpha,\beta}$ is either $t_{\alpha,\beta}$ or $\neg t_{\alpha,\beta}$), we understand that if components $C_1, \ldots, C_m$ function and failures of components $C_{i_1}, \ldots, C_{i_n}$ occurred in the indicated order, then the system works.

The timing constraints in a TSS represent an order and as such are subject to transitivity. We say that a TSS T is *complete* if it contains the constraint $t_{\alpha,\gamma}$ whenever it contains both $t_{\alpha,\beta}$ and $t_{\beta,\gamma}$ for some β, and *mutatis mutandis* for the constraints involving negative literals. We then define the *completion* T^* of a TSS T as the smallest complete TSS that contains T. The set T^* can be computed by adding implied constraints until a fixed point is reached.

We can now define a partial order on TSSs by defining

$$T_1 \preceq T_2 \quad \text{iff} \quad T_1^* \subseteq T_2^*. \tag{15}$$

In other words, a TSS T_1 is "smaller" than T_2 iff the set of functioning components represented by the variables x_i in T_1 is a subset of those in T_2 and if the timing constraints of T_1 are less restrictive than those of T_2.

The relation $\preceq$ is partial, and two TSSs can be incomparable such as $\{x_1, x_3, t_{2,4}\}$ and $\{x_1, x_2, t_{3,4}\}$.

For the purpose of a qualitative analysis, we aim at computing the *minimal tie sets with sequences* (MTSSs), based on the partial order above.

Example 4.1. To illustrate the notion of tie sets with sequences, we consider the system from Fig.1.d, whose (dualized) structure function is

$$\begin{aligned} x_{sp_3} = \quad & x_1 \\ & \vee\ x_3 \\ & \vee\ x_4 \wedge x_2 \\ & \vee\ x_4 \wedge (\neg x_1 \wedge \neg x_2 \wedge t_{1,2}) \\ & \qquad \wedge (\neg x_2 \wedge \neg x_3 \wedge \neg t_{2,3}) \end{aligned} \tag{16}$$

Its MTSSs are given by the sets:

$$\{x_1\}, \{x_3\}, \{x_2, x_4\}, \{x_4, t_{1,2}, \neg t_{2,3}\} \tag{17}$$

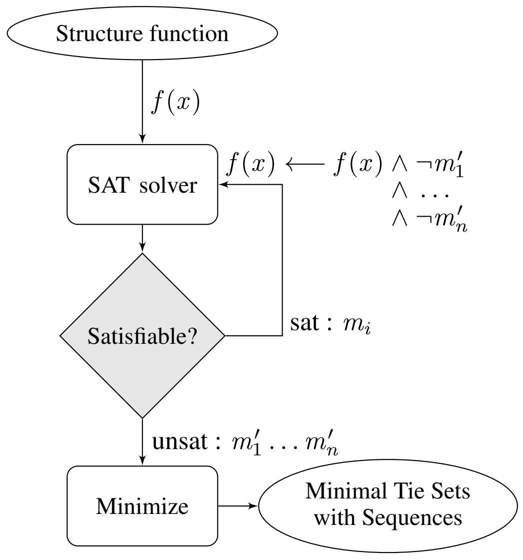
Structure function
$f(x)$
SAT solver
$f(x) \longleftarrow f(x) \wedge \neg m'_1 \wedge \ldots \wedge \neg m'_n$
Satisfiable?
sat : m_i
unsat : $m'_1 \ldots m'_n$
Minimize
Minimal Tie Sets with Sequences

Fig. 7. Outline of the algorithm.

We now present an algorithm for computing the minimal tie sets with sequences from the structure function by using satisfiability techniques. The method is outlined in Fig. 7. The first step is to call a SAT solver to check if the structure function is satisfiable and, if so, obtain a model m_1. The model computed by the SAT solver may assign false to some Boolean variables x_i, represented as $\neg x_i$ in the output of the SAT solver. However, since we consider coherent systems, we know that the failure of a component cannot contribute to the functioning of the overall system, implying that the status of component C_i is irrelevant. We can therefore delete negated variables x_i from the model obtained by the solver. Furthermore, we can remove literals built from variables $t_{i,j}$ or $t_{j,i}$ if the model also contains x_i: since component C_i functions, the value assigned to its time of failure t_i is irrelevant, and so is its order w.r.t. the failure of component C_j. Pruning the model m_1 in this way, we obtain a smaller assignment m'_1 that represents a tie set (cf. Example 4.2 below).

In order to compute further TSSs, we add the clause $\neg m'_1$ to the input given to the SAT solver.

Table 1. Number of MTSs computed and time of computation.

Number of variables	Number of components	Number of (inner) gates	Number of TSs found	Number of MTSs	Time (s)
60	25	35	15	12	0.6
71	32	39	1993	960	218
80	51	29	5	5	0.2
82	53	29	131	53	5.1
84	49	35	31	19	1.1
129	89	40	$>$ 37 000		$>$ 3 days
186	80	106	$>$ 25 000		$>$ 1 day

We repeat the procedure until the SAT solver declares the input formula to be unsatisfiable. At this point, we have computed a set $\mathscr{T}$ of TSSs that cover the set of all TSSs in the sense that for every TSS T that is a model of the structure function, $\mathscr{T}$ contains a TSS $T' \preceq T$. In practice, we have found it useful to apply an option that causes the SAT solver to prefer assigning true to Boolean variables in order to speed up the procedure.

Example 4.2. In order to illustrate the pruning step, assume that the SAT solver produced the model $\{x_1, \neg x_2, \neg x_3, \neg t_{1,2}, t_{1,3}, \neg t_{2,3}\}$ at some step during the computation. We discard $\neg x_2$ and $\neg x_3$ because failed components are irrelevant for tie sets of a coherent system, as well as $\neg t_{1,2}$ and $t_{1,3}$ because timing constraints concerning the functioning component 1 are irrelevant. The result of the pruning step for this example is therefore $\{x_1, \neg t_{2,3}\}$, and we add the clause $\neg x_1 \vee t_{2,3}$ for the next round of computation.

As explained above, the set $\mathscr{T}$ computed as a result of the repeated calls to the SAT solver represents a cover for the TSSs of the system represented by the structure function. In particular, it contains all MTSSs. However, it may still contain some TSSs that are not minimal. It therefore remains to remove all TSSs T from $\mathscr{T}$ for which $\mathscr{T}$ contains some $T' \prec T$. This step is represented by the box labeled "Minimize" in Figure 7.

5. Application

We implemented the approach presented in Section 4 for computing the minimal tie sets with sequences (MTSSs) from the structure function obtained from dynamic fault trees and validated it by applying it to industrial-range systems. This section gathers our results.

5.1. *Validation of the approach*

In a first step, we validate our approach by comparing the results that we obtain to those computed by the software GRIF (Satodev (2018)), based on the ZBDD-calculus. Zero-suppressed BDDs (ZBDDs) introduced by Minato (1993) are BDDs based on a new reduction rule in order to create a unique and compact representation of sets that appear in many combinatorial problems.

GRIF only applies to SFTs and computes their minimal cut sets. Since cut sets are the dual of tie sets, it is straightforward to adapt our approach for computing minimal cut sets (MCSs) instead of minimal tie sets. For each system considered, we obtained the same MCSs using GRIF and using our approach.

In a second step, we compute the tie sets (TSs) for the same benchmarks. The results appear in Table 1. The time of computation of the minimal tie sets (MTSs) depends on the number of components of the system and the number of (inner) gates of the structure function, which determine the number of variables that appear in the structure function. However, it also depends on the shape of the structure function.

Whereas our approach yields satisfactory response times for systems with less than a (few) hundred MTSs, we had to interrupt the computation for two systems with more than a hundred variables for which the SAT solver found several tens of thousands of tie sets. Indeed the more TSs we compute, the longer the SAT solver takes to produce a new model. We believe that the response times could be significantly improved by filtering out non-minimal tie sets (or TSSs) during the interaction loop with the SAT solver. In particular, doing so reduces the size of the input given to the solver.

5.2. *Application to a dynamic fault tree*

We now apply our method to the analysis of a system whose structure function is obtained from a DFT. The considered case study of a *Hypothetical Cardiac Assist System* (HCAS) was proposed by Boudali and Dugan (2005), inspired by a Cardiac Assist System described in Vemuri et al. (1999). Merle et al. (2014, 2016) computed the probability of appearance of the top event and then estimated the failure probabilities using Monte Carlo simulation.

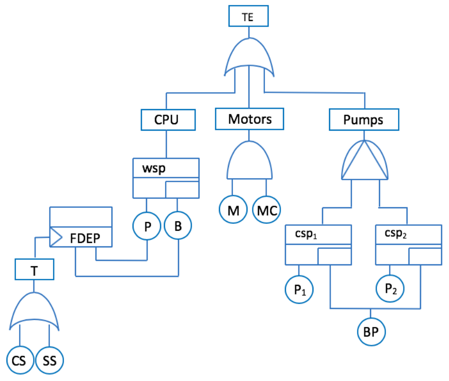

Fig. 8. Dynamic fault tree of the HCAS.

The HCAS is designed to treat mechanical and electrical failures of the heart. The system is divided into four subsystems: trigger, CPU unit, motor section, and pumps. The crossbar switch (CS) and the system supervisor (SS) represent the trigger, because the failure of either CS or SS triggers the failure of both CPUs. The CPU unit is modeled as a warm spare gate with a primary P and a redundant unit B, which corresponds to the backup CPU. For the motor section, either the motor (M) or the motor cable (MC) needs to be working. The pump unit consists of two cold spare gates csp_1 and csp_2. Each one has a primary pump (P$_1$ and P$_2$), and they share a common redundant pump (BP). In order for the pump unit to fail, all three pumps have to fail, and the gate csp_1 needs to fail before the gate csp_2.

The dynamic fault tree that models the potential failure of the HCAS is shown in Fig. 8. We compute the corresponding structure function c_{HCAS} as explained in Section 3.

$$\begin{aligned}
c_{HCAS} &= c_{CPU} \vee c_{Motors} \vee c_{Pumps}\\
c_{CPU} &= c_P \wedge c_B.\\
c_{Motors} &= c_M \wedge c_{MC}\\
c_{Pumps} &= c_{csp_1} \wedge c_{csp_2} \wedge t_{csp_1,csp_2}\\
c_{csp_1} &= c_{P_1} \wedge (c_{BP} \vee (c_{P_2} \wedge t_{P_2,P_1}))\\
c_{csp_2} &= c_{P_2} \wedge (c_{BP} \vee (c_{P_1} \wedge t_{P_1,P_2}))\\
c_T &= c_{CS} \wedge c_{SS}\\
c_T &\Rightarrow c_P \wedge c_B\\
t_{hcas} &= \min(t_{CPU}, t_{Motors}, t_{Pumps})\\
t_{CPU} &= \min(t_T, \max(t_P, t_B))\\
t_{Motors} &= \max(t_M, t_{MC})\\
t_{Pumps} &= t_{csp_2}\\
t_T &= \min(t_{CS}, t_{SS})\\
t_{csp_1} &= \begin{cases} \max(t_{P_1}, t_{BP}) & \text{if } t_{P_1} < t_{P_2}\\ t_{P_1} & \text{otherwise}\end{cases}
\end{aligned}$$

$$t_{csp_2} = \begin{cases} \max(t_{P_2}, t_{BP}) & \text{if } t_{P_2} < t_{P_1}\\ t_{P_2} & \text{otherwise}\end{cases}$$

As explained at the end of Section 3, we eliminate conditional equations, maxima and minima and obtain:

$$\begin{aligned}
t_{csp_1,csp_2} = &\quad t_{P_1,P_2} \wedge \max(t_{P_1}, t_{BP}) < t_{P_2}\\
&\vee t_{P_2,P_1} \wedge t_{P_1} < \max(t_{P_2}, t_{BP})\\
= &\quad t_{P_1,P_2} \wedge t_{BP,P_2}\\
&\vee t_{P_2,P_1} \wedge t_{P_1,BP}
\end{aligned}$$

For cold spare gates, a timing constraint is added to show that the spare component can only fail after being activated, and therefore after a main component: $t_{BP} > \min(t_{P_1}, t_{P_2}) \vee \neg c_{BP}$.

We therefore add the constraint

$$(t_{P_1,BP} \wedge c_{P_1} \wedge c_{BP}) \vee (t_{P_2,BP} \wedge c_{P_2} \wedge c_{BP}) \vee \neg c_{BP}$$

to the Boolean representation of the structure function.

The transitivity between the temporal variables results in the following implications:

$$\begin{aligned}
t_{P_2,P_1} \wedge t_{P_1,BP} &\Rightarrow t_{P_2,BP}\\
\neg t_{P_2,P_1} \wedge t_{P_2,BP} &\Rightarrow t_{P_1,BP}\\
t_{P_2,BP} \wedge \neg t_{P_1,BP} &\Rightarrow t_{P_2,P_1}
\end{aligned} \qquad (18)$$

Applying our algorithm from Section 4, we compute the 16 MTSSs of the system:

$$\begin{aligned}
&\{x_{CS}, x_{SS}, x_P, x_M, t_{P_2,BP}, t_{BP,P_1}\}\\
&\{x_{CS}, x_{SS}, x_P, x_{MC}, t_{P_2,BP}, t_{BP,P_1}\}\\
&\{x_{CS}, x_{SS}, x_P, x_M, t_{P_1,P_2}, t_{P_2,BP}\}\\
&\{x_{CS}, x_{SS}, x_P, x_{MC}, t_{P_1,P_2}, t_{P_2,BP}\}\\
&\{x_{CS}, x_{SS}, x_P, x_M, t_{P_1,BP}, x_{P_2}\}\\
&\{x_{CS}, x_{SS}, x_P, x_{MC}, t_{P_1,BP}, x_{P_2}\}\\
&\{x_{CS}, x_{SS}, x_P, x_M, t_{P_2,P_1}, x_{BP}\}\\
&\{x_{CS}, x_{SS}, x_P, x_{MC}, t_{P_2,P_1}, x_{BP}\}\\
&\{x_{CS}, x_{SS}, x_B, x_M, t_{P_2,BP}, t_{BP,P_1}\}\\
&\{x_{CS}, x_{SS}, x_B, x_{MC}, t_{P_2,BP}, t_{BP,P_1}\}\\
&\{x_{CS}, x_{SS}, x_B, x_M, t_{P_1,P_2}, t_{P_2,BP}\}\\
&\{x_{CS}, x_{SS}, x_B, x_{MC} t_{P_1,P_2}, t_{P_2,BP}\}\\
&\{x_{CS}, x_{SS}, x_B, x_M, t_{P_1,BP}, x_{P_2}\}\\
&\{x_{CS}, x_{SS}, x_B, x_{MC}, t_{P_1,BP}, x_{P_2}\}\\
&\{x_{CS}, x_{SS}, x_B, x_M, t_{P_2,P_1}, x_{BP}\}\\
&\{x_{CS}, x_{SS}, x_B, x_{MC}, t_{P_2,P_1}, x_{BP}\}
\end{aligned}$$

The components CS and SS appear in each MTSS and are the most critical components of the system, since they are essential and have no redundancy.

6. Conclusion

This paper is a first step towards the reliability analysis of dynamic systems, in which we adapted the format of the structure function to dynamic fault trees. We also extended the notion of tie sets and defined the minimal tie sets with sequences to enable the qualitative analysis of dynamic fault trees. We validated our approach by applying it to several case studies, and we believe that some more algorithmic optimizations can make it even more competitive.

Kaufmann et al. (1977) proposed an approach for the direct analytical calculus of the reliability of a system based on the knowledge of the minimal tie sets. Brînzei and Aubry (2018) proposed an approach that enables quantitative reliability assessment based on graph models and more particularly the organization of tie and cut sets as a Hasse diagram. Based on these approaches, we will extend this work to the quantitative analysis of systems modeled by DFTs by defining the Hasse diagrams of tie and cut sets with orders and by extending the methods for computing the reliability function to such systems. Then we will use the work of Merle et al. (2010) to determine the reliability of the system from the reliability function.

A more detailed representation of spare gates would distinguish three possible states (stand-by, in operation, and failed) rather than just represent them using a Boolean variable. Besides, finer analyses could be obtained by considering more than two states of components that would correspond to degraded modes of operations. In future work, we are interested in extending our SAT-based approach to consider multi-state systems where component and system states can take more than two values.

Acknowledgement

This work was partly supported by the French PIA project “Lorraine Université d’Excellence”, reference ANR-15-IDEX-04-LUE.

References

Biere, A., M. Heule, H. van Maaren, and T. Walsh (Eds.) (2009). *Handbook of satisfiability*, Volume 185. IOS Press.

Boudali, H., P. Crouzen, and M. Stoelinga (2007). Dynamic fault tree analysis using input/output interactive markov chains. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN’07)*, pp. 708–717. IEEE.

Boudali, H. and J. B. Dugan (2005). A discrete-time bayesian network reliability modeling and analysis framework. *Reliability Engineering & System Safety 87*(3), 337–349.

Brînzei, N. and J.-F. Aubry (2018). Graphs models and algorithms for reliability assessment of coherent and non-coherent systems. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability 232*(2), 201–215.

Duroeulx, M., N. Brinzei, M. Duflot, and S. Merz (2017). Satisfiability techniques for computing minimal tie sets in reliability assessment. In *Mathematical Models on Reliability (MMR)*.

Ge, D., M. Lin, Y. Yang, R. Zhang, and Q. Chou (2015). Quantitative analysis of dynamic fault trees using improved sequential binary decision diagrams. *Reliability Engineering & System Safety 142*, 289–299.

Kaufmann, A., D. Grouchko, and R. Cruon (1977). *Mathematical Models for the Study of the Reliability of Systems* (Mathematics in Science and Engineering ed.), Volume 124. Elsevier.

Merle, G., J.-M. Roussel, and J.-J. Lesage (2014). Quantitative analysis of dynamic fault trees based on the structure function. *Quality and Reliability Engineering International 30*(1), 143–156.

Merle, G., J.-M. Roussel, J.-J. Lesage, V. Perchet, and N. Vayatis (2016). Quantitative analysis of dynamic fault trees based on the coupling of structure functions and monte carlo simulation. *Quality and Reliability Engineering International 32*(1), 7–18.

Merle, G., J.-M. Roussel, J.-J. Lesage, and N. Vayatis (2010). Analytical calculation of failure probabilities in dynamic fault trees including spare gates. In *European Safety and Reliability Conference (ESREL 2010)*, pp. pp–794. Taylor & Francis.

Minato, S.-i. (1993). Zero-suppressed bdds for set manipulation in combinatorial problems. In *Proceedings of the 30th international Design Automation Conference*, pp. 272–277. ACM.

Ruijters, E. and M. Stoelinga (2015). Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools. *Computer science review 15*, 29–62.

Satodev (2018). *GRIF - GRaphical Interface for reliability Forecasting software*. http://grif-workshop.com/.

Vemuri, K. K., J. B. Dugan, and K. J. Sullivan (1999). Automatic synthesis of fault trees for computer-based systems. *IEEE Transactions on Reliability 48*(4), 394–402.

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

Xing, L., O. Tannous, and J. B. Dugan (2011). Reliability analysis of nonrepairable cold-standby systems using sequential binary decision diagrams. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 42*(3), 715–726.

Zhu, P., J. Han, L. Liu, and M. J. Zuo (2014). A stochastic approach for the analysis of fault trees with priority and gates. *IEEE Transactions on Reliability 63*(2), 480–494.