Received August 29, 2019, accepted September 11, 2019, date of publication September 20, 2019, date of current version October 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942829

A Methodology for the Formal Verification of Dynamic Fault Trees Using HOL Theorem Proving

YASSMEEN ELDERHALLI, (Student Member, IEEE), OSMAN HASAN, (Senior Member, IEEE), AND SOFIÈNE TAHAR, (Senior Member, IEEE)
Department of Electrical and Computer Engineering, Concordia University, Montréal, QC H3G 1M8, Canada
Corresponding author: Yassmeen Elderhalli (y_elderh@ece.concordia.ca)

ABSTRACT Dynamic Fault Trees (DFTs) are increasingly being used for modeling the failure behaviors of systems, particularly dynamic behaviors that cannot be captured using conventional combinatorial models. Traditionally, paper and pencil or simulation are used for the analysis of DFTs. While the former can provide generic expressions for the probability of failure, its results are prone to human errors. The latter method is based on sampling and the results are not guaranteed to be complete. Leveraging upon the expressive and sound nature of higher-order logic (HOL) theorem proving, it has been recently proposed for the analysis of DFTs algebraically. In this paper, we propose a novel methodology for the formal analysis of DFTs, based on the algebraic approach, while capturing both the qualitative and probabilistic aspects using theorem proving. In this paper, we further enrich the DFT library in HOL by providing the formalization of spare gates with a shared spare and the verification details of their probabilistic behavior. To demonstrate the utilization of our methodology, we apply it for the formal analysis of two safety-critical systems, namely, a drive-by-wire system and a cardiac assist system.

INDEX TERMS Dynamic fault trees, qualitative analysis, quantitative analysis, higher-order logic, theorem proving, HOL4.

I. INTRODUCTION

Fault trees (FTs) have been widely used in modeling the causes of failure of systems [1]. The undesired system failure is modeled as the top event of the FT, while the basic events of faults that lead to the occurrence of this top event are modeled as FT inputs. The relationship among the basic events in leading to the occurrence of the top event are represented as FT gates. FTs can be categorized into Static Fault Trees (SFTs) and Dynamic Fault Trees (DFTs) depending on the modeled failure behavior [2]. In SFTs, the sources of failure are considered only without any specific order and the top event is modeled as a Boolean function. In DFTs, the failure dependencies are captured by introducing DFT gates that consider the time of failure of system components in affecting the failure of others [3].

Fault Tree Analysis (FTA) [1], including qualitative and quantitative aspects [1], is primarily used to provide information about the conditions required for the system failure as well as some numeric failure measures. In the *qualitative* analysis, the sources and factors contributing to system failure are identified. This is performed by first expressing the structure function of the top event, then the cut sets and cut sequences are identified. The cut sets represent the combinations of basic events that lead to the occurrence of the top event [1]. The cut sequences, on the other hand, identify the sequences of basic events that cause the occurrence of the top event [1]. The *quantitative* analysis is conducted by providing information about reliability metrics, such as the Mean Time To Failure (MTTF) as well as the probability of failure of the top event, which can be used to evaluate system reliability and thus devise appropriate recovery plans.

Several methods exist for conducting the DFT analysis, including Markov chain based approaches [4] and the algebraic approach [5]. In the Markov chain based analysis, the DFT is converted first into its equivalent Markov chain, and then analyzed using simulation methods, such as Markov chain Monte Carlo simulation [6]. However, the number of Markov chains states grows exponentially with the number

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Liu.

of system components, and thus would need a lot of computational resources for meaningful analysis. To overcome this issue, modularization can be applied to divide the DFT into static and dynamic parts [7]. The static part can be analyzed using any of the traditional methods, such as Binary Decision Diagrams [8]. While the Markov chain of the dynamic part can be analyzed using simulation. This way, the resulting Markov chain will be smaller and thus easier to handle. However, due to the sampling based nature of simulation, the analysis results cannot be guaranteed to be complete.

In the algebraic approach, the top event of the DFT is expressed as a function of basic input events. Temporal operators are then defined in the algebraic approach to capture the failure dependencies. In addition, several simplification rules are applied to obtain a simplified structure function that is used for the qualitative and probabilistic analyses of a DFT [9]. Although the algebraic approach generates a reduced form of the DFT's structure function as well as generic expressions of probability of failures that are independent of the distribution of the basic events [9], its results are prone to human errors, particularly if the underlying math of this approach is not formally verified. Simulation can be combined with the algebraic approaches for DFT analysis [10], but this hybrid approach also suffers from the same shortcomings as simulation.

Formal methods [11], which are computer based tools that analyze systems based on their mathematical models, can overcome the previously mentioned accuracy problems of simulation based analysis. Two main formal methods, i.e., model checking and theorem proving have been used in the context of DFT analysis. Model checking [12] is used to automatically analyze systems modeled as state machines, while higher-order-logic (HOL) theorem proving [13] verifies the properties of systems using interactive proofs based on deductive reasoning. Probabilistic Model Checkers (PMCs), such as STORM [14], have been successfully used for the analysis of DFTs [15]. Using PMCs in the analysis is appealing, as it is an automatic verification method. However, complex systems with large Markov chains cannot be handled unless some reduction algorithms are invoked. Usually the implementations of such reduction algorithms are not formally verified, which may lead to discrepancies between the reduced Markov chain and the original DFT. Moreover, PMCs have not been used in determining the cut sets and cut sequences of a DFT. More importantly, PMCs cannot be used to verify generic mathematical expressions for the failure probabilities, and thus whenever the failure rates of system components change, the whole analysis has to be repeated to reflect the impact of such a change. Leveraging upon the expressive and sound nature of HOL theorem proving, it has been recently proposed for the analysis of DFTs based on the algebraic approach, qualitatively [16] and quantitatively [17]. We believe that HOL theorem proving has a great potential to overcome the soundness related shortcomings of the non-formal algebraic analysis and thus can provide a complementary approach to existing techniques.

In this paper, we propose a methodology to perform formal qualitative and probabilistic analyses based on the algebraic approach [9] using HOL theorem proving. In particular, we verify a reduced form of the DFT structure function that can be used to obtain a formally verified simplified form of the cut sets and cut sequences as well as a generic expression for the probability of failure. Our proposed methodology can be used to provide the reliability engineer with the confidence that the obtained results using the non-formal techniques, such as paper-and-pencil or simulation, are correct. This is of utmost importance since wrong analysis results may lead to disastrous consequences, including the loss of human life. An accurate and complete dependability analysis could avoid such consequences.

It is important to note that the proposed methodology is primarily based on the formalization of the algebraic approach presented in [9]. However, a distinguishing feature of our formalization is that it allows us to conduct computer based proofs of the probability of failure expressions for DFT gates within the sound environment of a theorem prover software. These proofs are either unavailable in [9], or we are able to conduct them in a simpler manner. In addition, we explicitly provided a definition for DFT events that is used to provide the set of time to perform the probabilistic analysis. Moreover, as we are providing the formalization in a theorem prover, data-types should be carefully handled to capture both the behavior of DFT gates and the probability of their failure. These details are not provided in [9], which signifies the importance of the proposed methodology.

To highlight the importance of our methodology, we report our own experience in locating a flaw in one of the proposed algebraic approaches for DFT analysis, which affects the correctness of the reported results. Namely, in the work published in [18], a simple algebra for the analysis of FTs is proposed, which includes introducing new definitions for DFT gates and events as variables. In addition, temporal operators are defined to capture the order of inputs. Using these temporal operators, many simplification theorems are introduced to simplify the structure function of the top event. Using HOL theorem proving, we have been able to identify a flaw in one of the simplification theorems that is also used in the application section of the paper, which affects the integrity of the reported results. This further strengthens our claim that it is necessary to formally verify the correctness of the algebraic approach to preserve the integrity of the analysis.

A. NOVEL CONTRIBUTIONS OF THE PAPER

The main contributions of the paper can be summarized as follows:

- A comprehensive methodology for the formal analysis of DFTs based on the algebraic approach using theorem proving.
- The HOL formalization of the failure behavior of spare gates with a shared spare.

- The identification of a flaw in the fault tree algebra published in [18].
- The formal analysis of two safety-critical systems.

The proposed methodology provides complete formal qualitative and probabilistic analyses in the form of generic expressions of probability of failure using HOL theorem proving. Due to the involvement of a theorem prover in the analysis, it is mandatory to explicitly mention all of the required assumptions or conditions along with the corresponding theorems. We argue that this kind of assurance is not guaranteed by any other analysis technique for DFT. Given the safety-critical nature of the systems requiring DFT analysis, this assurance can be very useful to ensure the reliability of these systems. For example, the flaw that we were able to identify in [18] is a missing assumption in one of the simplification theorems. The authors of [18] assumed that this theorem can be used without any previous assumption or condition, which is not valid as we have been able to verify that this theorem is only true under certain conditions. Using such theorem without the required condition leads to wrong simplification results that can affect the results of the qualitative and the quantitative analyses.

In addition, we present a HOL formalization of the probabilistic behavior of a spare gate with a shared spare, which represents one of the common configurations in DFTs of real world-systems. This formalization allows us to formally analyze the corresponding DFTs that use this gate. Finally, to demonstrate the utilization of our framework, we apply our formal analysis to three DFT examples and two safety-critical systems. The first system is a drive-by-wire system [19], which represents an important component in modern cars that provides electrical control to the brakes and throttle systems of a car. The second system is the cardiac assist system [20], which is a device that provides cardiac assistance to patients with heart problems. Both systems require sound analysis as any flaw in the analysis may lead to losses in lives.

B. ORGANIZATION OF THE PAPER

The paper is structured as follows: The related work is presented in Section II. Some preliminaries that are required for the understanding of the proposed methodology are presented in Section III, including the DFT theory in HOL. In Section IV, we present our proposed methodology. Section V provides the HOL formalization of the spare gate with a shared spare. We use three DFT examples, in Section VI, to illustrate the usage of the proposed methodology. Section VII provides details about the flaw detected in one of the DFT algebras. The formal analyses of the drive-by-wire and cardiac assist systems are presented in Section VIII. Finally, we conclude the paper in Section IX.

II. RELATED WORK

There are several techniques to analyze DFTs based on the underlying model. For example, Markov chain based approaches are widely used for this purpose as the state based nature of the Markov chain can easily capture the dynamic failure behavior of DFTs. Since this analysis is state based, the size of the state-space grows exponentially with the number of components. This leads to an exponential growth of the state space, where it becomes impossible to exhaustively analyze a large state space given the limited amount of computational power and memory. These issues are usually catered for by using modularization [21]. The DFTCalc tool [22] analyzes the DFT using an Input/Output Interactive Markov Chain, an extension of Continuous Time Markov Chains (CTMCs), which is built based on a compositional aggregation technique [4]. Similarly, other stochastic approaches, like [23], have been used to evaluate the probability of failure of DFTs. However, they cannot guarantee the completeness of analysis as the computations are performed using stochastic computational models.

Formal methods have also been used for DFT analysis using Markov chains. Petri nets, for instance, are used for the analysis of DFTs by first converting the given DFT into a Generalized Stochastic Petri Net (GSPN) [24]. This Petri Net can then be converted into a CTMC using a tool that transforms a GSPN into its equivalent CTMC, like GreatSPN [25]. The generated CTMC can be analyzed using a formal method, like model checking (e.g., PRISM [26]), or using simulation. However, the problem in analyzing DFTs using Markov chain based approaches lies mainly in the fact that no generic expressions for the system reliability are obtained, which would require repeating the analysis when modifying the failure rates of the system components. In addition, state-space problems are usually encountered that require some reduction techniques [27]. However, as these reduction techniques are usually not formally verified, it is not very straightforward to ascertain that the obtained results correspond to the original system model.

Since the formalization of the probability theory in higher-order logic [28]–[31], theorem proving has also been used for reliability analysis. For example, some properties for continuous random variables were verified to formally reason about some system reliability properties, such as MTTF [32]. Moreover, some reliability theory elements were formally verified and used in the formal analysis of a reconfigurable memory array with stuck-at and coupling faults [33]. Although, in [34], a complete framework for the analysis of FTs using theorem proving has been proposed and used to formally analyze some real-world applications, like an air traffic management system [34] and a solar array for a satellite system [35], this HOL formalization cannot handle or be extended to verify the dynamic properties of DFTs.

So far, no framework for the analysis of DFTs using theorem proving exists, which is the scope of the current work. We propose a HOL formalization of DFT foundations, which, to the best of our knowledge, provides the first HOL theory for DFT analysis in the HOL theorem proving system. This formalization in turn can be used to formally analyze DFTs of real-world applications that would be very useful in the context of safety-critical applications. Moreover, higher-order logic allows universally quantification over functions and predicates, which makes developments in the HOL system generic for general use.

Generic expressions for the system failure behavior can be obtained using the algebraic approach [9]. In [5], the authors provided a framework for the analysis of DFTs based on algebraically expressing the structure function of the top event. However, the analysis results of this framework are based on analytically verified expressions using paper-and-pencil proof methods, which are prone to errors. In [16], we proposed an integrated methodology to perform DFT qualitative analysis using theorem proving, and probabilistic analysis using a PMC. We formalized using HOL theorem proving the DFT gates and the simplification theorems of the proposed algebraic approach in [9]. This integrated methodology provides sound DFT qualitative analysis. However, since a PMC is invoked, this integrated methodology cannot provide generic expressions of probability of failure. In [17], we provided enhanced HOL formal definitions for DFT gates and temporal operators that cater for the probabilistic analysis of DFTs and provided the details of the formal verification of probabilistic analysis of FT gates, static and dynamic ones using HOL theorem proving. Based on the formalization of DFTs in [17], in this work, we extend the library of DFT gates with spare gates with shared spare. Furthermore, we propose a thorough methodology to perform both DFT qualitative and probabilistic analysis in the form of generic expressions of probability distributions and density functions. Moreover, we extend the mathematical formalization to handle more complex mathematical models than in [17], which allows us to provide the formal analysis of two real-world case studies.

III. PRELIMINARIES

In this section, we present some preliminaries related to HOL4 theorem proving [36], probability and DFT theories to facilitate the understanding of our proposed methodology.

A. HOL4 THEOREM PROVING

Theorem proving is one of the formal methods techniques that uses a computerized program, i.e., a theorem prover, to carry out mathematical proofs of theorems based on deductive reasoning. The level of expressiveness of these theorems depends on the type of logic used, like first-order logic and higher-order logic (HOL). There are several HOL theorem provers that are available such as HOL4 [36], Isabelle [37] and Coq [38], which vary in the availability of the supported libraries.

HOL4 is an interactive theorem prover, which is capable of verifying a wide range of hardware and software systems as well as mathematical expressions constructed in HOL. Being an interactive tool, HOL4 requires the guidance of the verification engineer to complete the verification process. In order to verify certain properties of a system, a mathematical model for this system should be created first, then based on this model, HOL4 can be used to verify several system properties in the form of theorems. This makes HOL4 an expressive platform for the verification of any system that can be described mathematically. The main characteristic of HOL theorem proving is its soundness, i.e., no wrong proof goal can be proved. The core of HOL4 consists only of five axioms and eight inference rules. Soundness is assured as any new theorem should be verified based on these axioms and rules, or based on previously proven theorems. In addition, no approximation is involved in the models, as their behavior, such as the failure in the case of DFTs, is captured in mathematical terms. These features make HOL4 suitable for carrying out the DFT based analysis of safety-critical systems that require sound verification results. The term *formalization* means to mathematically model the behavior of a system in an appropriate logic. A *proof goal* consists of a list of assumptions of type Boolean and a conclusion. For example, "`∀(x:real). 0 < x ⇒ 0 < x`$^{-1}$" is a proof goal, which can be formally verified as a theorem in HOL4. A *theory* in HOL4 is a collection of definitions, constants and theorems that can be included in the working environment to be used in verifying other proof goals.

B. PROBABILITY AND LEBESGUE INTEGRAL THEORIES IN HOL4

A measure space is a triple (Ω, Σ, μ), where Ω is the sample space, Σ is a σ-algebra subsets of the sample space Ω, and μ is a measure with domain Σ. The probability theory in HOL4 is defined based on the measure theory, where a probability space is a measure space with a triple $(\Omega, \Sigma, \mathcal{P}r)$. For this triple, $\mathcal{P}r$ is the measure, where the $\mathcal{P}r$ of the sample space is 1 [30]. For a probability space p, the functions `p_space`, `events`, and `prob` correspond to Ω, Σ and $\mathcal{P}r$, respectively [30]. For example, `p_space` is formally defined as [30]:

Definition 1:

`p_space = m_space`

where `m_space` returns the space Ω of the measure space. One of the main concepts while dealing with the probability of events is the statistical independence (s-independence) of these events. In probability theory, the probability of the intersection of any two s-independent events equals the multiplication of the probability of the individual events, i.e.,

$$\mathcal{P}r(A \cap B) = \mathcal{P}r(A) \times \mathcal{P}r(B) \tag{1}$$

Random variables are used in probability theory to mathematically describe random or unpredictable events such as the outcome of dice rolling. These random variables are defined as measurable functions that map the outcomes of the probability space into another space that returns quantities; numbers for example. Random variables have been defined in HOL4 as measurable functions that map from the probability space to another σ-algebra space [30].

The probability distribution function is defined in HOL4 as the `distribution` function, which returns the probability of a random variable X for a given set in the probability space. This `distribution` function is used to define the cumulative distribution function (CDF) of a random variable, which is the probability that the value of the random variable

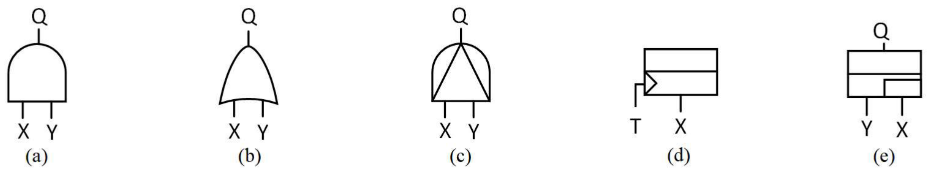


FIGURE 1. **Fault Tree Gates: (a) AND (b) OR (c) PAND (d) FDEP (e) SPARE.**

is less than or equal to a certain value t. This is formally defined as [35]:

Definition 2: Cumulative Distribution Function

```
⊢ ∀p X t. CDF p X t =
    distribution p X {y | y ≤ (t:real)}
```

where `p` is the probability space, `X` is the random variable and `t` is the value that we are finding the CDF for, i.e., $F_X(t)$.

It is quite common in DFT analysis to find the probability of an event that is composed of the union of several events corresponding to multiple random variables. In these cases, the probabilistic *Principle of Inclusion and Exclusion* (PIE) can be used, which has been formally verified in [35] as:

$$Pr(\bigcup_{i=1}^{n} A_i) = \sum_{t \neq \{\}, t \subseteq \{1,2,\ldots,n\}} (-1)^{|t|+1} Pr(\bigcap_{j \in t} A_j) \qquad (2)$$

The Lebesgue integral is defined in HOL4 using positive simple functions [39], which are measurable functions defined as a linear combinations of indicator functions of measurable sets representing a partition of the space [40].

In this paper, we are integrating the probability density functions (PDFs) over the real line. Therefore, we will use the Lebesgue-Borel measure with the Lebesgue integral. The Lebesgue-Borel (`lborel`) measure is a measure defined over the real line. As with any measure, `lborel` should have a space and measurable sets. For `lborel`, the real line represents its space and the borel sets represent the `lborel` measurable sets [41]. The reliability [33] and probability [40] related theories in HOL4 are quite rich, which makes HOL4 a more natural choice to build DFT theories.

C. DFT THEORY IN HOL4

We provide a brief description of DFTs and our formalization of DFT gates and simplification theorems [17] based on the algebraic approach presented in [9]. Since all the events are defined based on the time of failure, we will refer to the time of failure of any event A as $d(A)$ [9].

1) DFT GATES FORMALIZATION

Two identity elements are defined to model an event that always fails ($ALWAYS = 0$) and a fail safe event that can never fail ($NEVER = +\infty$). The mathematical expressions as well as the formal definitions of these two elements are listed in Table 1 [17], where `extreal` is a HOL data-type for extended-real numbers that include real numbers and $\pm\infty$. This data-type is chosen to be able to model the NEVER event using $+\infty$ [17].

TABLE 1. **Identity elements and temporal operators.**

Mathematical Expression	Formal Definition
ALWAYS	
$d(ALWAYS) = 0$	`⊢ ALWAYS = (λs. 0:extreal)`
NEVER	
$d(NEVER) = +\infty$	`⊢ NEVER = (λs. PosInf)`
Before	
$d(X \lhd Y) = \begin{cases} d(X), & d(X) < d(Y) \\ +\infty, & d(X) \geq d(Y) \end{cases}$	`⊢ ∀X Y. X ◁ Y = (λs. if X s < Y s then X s else PosInf)`
Simultaneous	
$d(X \Delta Y) = \begin{cases} d(X), & d(X) = d(Y) \\ +\infty, & d(X) \neq d(Y) \end{cases}$	`⊢ ∀X Y. X Δ Y = (λs. if X s = Y s then X s else PosInf)`
Inclusive Before	
$d(X \unlhd Y) = \begin{cases} d(X), & d(X) \leq d(Y) \\ +\infty, & d(X) > d(Y) \end{cases}$	`⊢ ∀X Y. X ⊴ Y = (λs. if X s ≤ Y s then X s else PosInf)`

Three temporal operators are defined in the algebraic approach [9] to represent the sequences of failure of input events, i.e., the first input event occurs before, at the same time or before or at the same time of the second input event for the *Before* ($\lhd$), *Simultaneous* (Δ) and *Inclusive Before* ($\unlhd$) operators, respectively. If the required condition of the operator is not satisfied, then the output event can never occur [9]. It is assumed that for any two basic events that exhibit continuous failure distributions, they cannot fail at the same time [9], i.e., $d(X \Delta Y) = NEVER$. Table 1 lists the mathematical expressions of the temporal operators [9] along with our HOL formal definitions [17].

Fig. 1 shows the common FT gates including static ones; AND and OR, as well as the dynamic ones; Priority-AND (PAND), Functional DEPendency (FDEP) and Spare gates [3]. Table 2 shows their mathematical expressions based on the algebraic approach [9] along with our formal definitions [17], where `max` and `min` are HOL functions that return the maximum and minimum values of their arguments, respectively.

The output event of the *PAND* gate occurs with the occurrence of both inputs as long as the occurrence of the input events is in sequence, conventionally from left to right. The *FDEP* gate is utilized when one system component triggers the failure of another component. For example, for the FDEP gate in Fig. 1(d), the occurrence of input X is triggered by the occurrence of input T. Therefore, variable X_T is used to model the global behavior of the input of the FDEP, where the behavior is modeled using the minimum of both inputs

TABLE 2. Fault tree gates definitions.

Mathematical Expression	Formal Definition
AND	
$d(X \cdot Y) = max(d(X), d(Y))$	`⊢ ∀X Y. D_AND X Y = (λs. max (X s) (Y s))`
OR	
$d(X+Y) = min(d(X), d(Y))$	`⊢ ∀X Y. D_OR X Y = (λs. min (X s) (Y s))`
PAND	
$d(Q_{PAND}) = \begin{cases} d(Y), & d(X) \leq d(Y) \\ +\infty, & d(X) > d(Y) \end{cases}$	`⊢ ∀X Y. PAND X Y = (λs. if X s ≤ Y s then Y s else PosInf)`
FDEP	
$d(X_T) = min(d(X), d(T))$	`⊢ ∀X T. FDEP X T = (λs. min (X s) (T s))`
HSP	
$d(Q_{HSP}) = max(d(Y), d(X))$	`⊢ ∀X Y. HSP Y X = (λs. max (Y s) (X s))`
CSP	
$d(Q_{CSP}) = \begin{cases} d(X), & d(Y) < d(X) \\ +\infty, & d(Y) \geq d(X) \end{cases}$	`⊢ ∀X Y. CSP Y X = (λs. if Y s < X s then X s else PosInf)`
WSP	
$d(Q_{WSP}) = d(Y \cdot (X_d \lhd Y) + X_a \cdot (Y \lhd X_a) + Y \Delta X_a + Y \Delta X_d)$	$\vdash \forall \mathtt{Y}\ \mathtt{X_a}\ \mathtt{X_d}.\ \mathtt{WSP}\ \mathtt{Y}\ \mathtt{X_a}\ \mathtt{X_d} = \mathtt{Y} \cdot (\mathtt{X_d} \lhd \mathtt{Y}) + \mathtt{X_a} \cdot (\mathtt{Y} \lhd \mathtt{X_a}) + \mathtt{Y}\ \Delta\ \mathtt{X_a} + \mathtt{Y}\ \Delta\ \mathtt{X_d}$

T and X. In this work, we treat the FDEP gate as an OR gate as proposed in [42], [43]. Finally, the *Spare* gate models spare parts in systems, where the spare part replaces the active part after failure. For the spare gate, shown in Fig. 1(e), Y is the main part and X is the spare. The spare gate has three forms depending on the failure behavior of the spare part in its dormant mode: (1) The *Hot SPare* (*HSP*) gate, where the spare has the same probability of failure in both active and dormant states. (2) *Cold SPare* (*CSP*) gate, where the spare part cannot fail in the dormancy state. (3) The *Warm SPare* (*WSP*) gate, where the spare part can fail in both its states but with different probabilities of failure. Usually, the probability of failure for the dormant state is attenuated by a dormancy factor from the probability of failure of the active state. Therefore, it is required to distinguish between both states while performing the analysis. Thus, the spare part X of Fig. 1(e) can be denoted by X_a and X_d for active and dormant states, respectively. The WSP is modeled in Table 2 using the temporal operators, AND and OR. Details about the formalization of the DFT gates can be found in [17].

In [9], many algebraic simplification theorems are introduced that enable the simplification of the structure function of the top event, such as the commutativity of the OR and AND. This will result in having a reduced form that facilitates the probabilistic analysis. We formally verified these simplification theorems [17] in HOL4, and identified the required conditions for these theorems to hold. More details about the corresponding formally verified simplification theorems can be found in [44] and [17].

2) FORMAL PROBABILISTIC BEHAVIOR OF DFT GATES

The foremost requirement for formally conducting the probabilistic analysis of DFTs is to have verified expressions for the probability of failure of DFT gates. A set of probabilistic expressions for the gates and operators were proposed in [9]. However, these expressions were not formally verified, and thus they cannot be used in a sound environment for the formal DFT analysis as such. We formalized the probabilistic expressions of all gates and operators that form the basis of any DFT [17]. In this section, we present a brief overview of this formalization for the probabilistic failure behavior of DFT gates with s-independent and s-dependent events.

We formally define a DFT event that represents the set of time until which we are interested to find the probability of failure at, and verify that it is equal to $F_X(t)$. We formally defined a DFT event as [17]:

Definition 3: DFT Event

```
⊢ ∀p X t. DFT_event p X t =
            {s | X s ≤ Normal t} ∩ p_space p
```

where p is the probability space, X is the random variable that we need to create the event for, which can be replaced with the function of a gate or a DFT, and finally t is the time until which we are interested in finding the probability at. The function `Normal` typecasts its argument from real to an extended-real value. The return type of DFT gates and operators is `extreal`. However, we chose time t to be real since the integration is over the real line. Therefore, in all our theorems we have to typecast time t from real to extended-real data-type using the `Normal` function, and have to typecast the random variables from extended-real to real using the `real` function to integrate the density and distribution functions over the real line.

When dealing with DFT gates, it is usually assumed that the input events are s-independent. However, sometimes this condition fails to hold when dealing with the WSP and CSP gates, as will be explained in the sequel. In the case of s-independent input events, we have four probabilistic expressions [45], in which we are interested in finding the probability of failure until time t. Our verified theorems for the four expressions are listed in Table 3 [17] (first four), where the theorems are presented in a mixed standard and formal math notations to facilitate their understanding. In Table 3, F_X and F_Y are the CDFs of random variables X and Y, respectively, and f_X and f_Y represent their corresponding PDFs. The first expression, in Table 3, represents the probability of failure of the top event of the AND and HSP gates, which is mainly based on the probability of the intersection of two s-independent events. The second expression in Table 3 represents the failure probability for the OR and FDEP gates, which is the probability of the union of two s-independent events. The following two expressions represent the probability of the *after* and *before* events, respectively. The former event expresses the probability of Y failing after X until time t. The latter expression represents the probability of X failing before Y until time t, so it is not necessary that Y fails in this case. The *after* event represents the PAND gate event, if the input events are basic events.

In Table 3, `prop_space p` is a predicate that ensures that p is a probability space and `rv_gt0_ninfinity`

TABLE 3. Formally verified probability of DFT gates.

Probability of AND & HSP gates
⊢ ∀p t X Y. rv_gt0_ninfinity [X; Y] ∧ indep_var p lborel (λs. real (X s)) lborel (λs. real (Y s)) ⇒ (prob p (DFT_event p (X · Y) t) = F_X(t) × F_Y(t))
Probability of OR & FDEP gates
⊢ ∀p t X Y. rv_gt0_ninfinity [X; Y] ∧ All_distinct_events p [X;Y] t ∧ indep_var p lborel (λs. real (X s)) lborel (λs. real (Y s)) ⇒ (prob p (DFT_event p (X + Y) t) = F_X(t) + F_Y(t) − F_X(t) × F_Y(t))
Probability of PAND gate & *after* event
⊢ ∀X Y p f_y t. rv_gt0_ninfinity [X; Y] ∧ 0 ≤ t ∧ prob_space p ∧ indep_var p lborel (λs. real (X s)) lborel (λs. real (Y s)) ∧ distributed p lborel (λs. real (Y s)) f_y ∧ (∀y. 0 ≤ f_y(y)) ∧ cont_CDF p (λs. real (X s)) ∧ measurable_CDF p (λs. real (X s)) ⇒ (prob p (DFT_event p (Y·(X⊲Y)) t) = $\int_0^t$ f_Y(y) F_X(y) dy)
Probability of *before* event
⊢ ∀X Y p f_x t. rv_gt0_ninfinity [X; Y] ∧ 0 ≤ t ∧ prob_space p ∧ indep_var p lborel (λs. real (X s)) lborel (λs. real (Y s)) ∧ distributed p lborel (λs. real (X s)) f_x ∧ (∀x. 0 ≤ f_x(x)) ∧ measurable_CDF p (λ s. real (Y s)) ⇒ (prob p (DFT_event p (X ⊲ Y) t) = $\int_0^t$ f_X(x)(1−F_Y(x)) dx)
Probability of CSP gate
⊢ ∀p X Y f_{xy} f_y $f_{X_a \mid Y}$ t. rv_gt0_ninfinity [X; Y] ∧ 0 ≤ t ∧ (∀y. cond_density lborel lborel p (λs. real (X s)) (λs. real (Y s)) y f_{xy} f_y $f_{X_a \mid Y}$) ∧ prob_space p ∧ den_gt0_ninfinity f_{xy} f_y $f_{X_a \mid Y}$ ⇒ (prob p (DFT_event p (CSP Y X) t) = $\int_0^t \left(\int_y^t f_{(X_a \mid Y=y)}(x)\ dx \right) f_Y(y)\ dy$
Probability of WSP gate
⊢ ∀p Y X_a X_d t f_y f_{xy} $f_{X_a \mid Y}$. prob_space p ∧ (∀s. ALL_DISTINCT [X_a s; X_d s; Y s]) ∧ DISJOINT_WSP Y X_a X_d t ∧ rv_gt0_ninfinity [X_a; X_d; Y] ∧ 0 ≤ t ∧ (∀y. cond_density lborel lborel p (λs. real (X_a s))(λs. real (Y s)) y f_{xy} f_y $f_{X_a \mid Y}$) ∧ den_gt0_ninfinity f_{xy} f_y $f_{X_a \mid Y}$∧ indep_var p lborel (λs. real (X_d s)) lborel (λs. real (Y s)) ∧ cont_CDF p (λs. real (X_d s)) ∧ measurable_CDF p (λs. real (X_d s)) ⇒ (prob p (DFT_event p (WSP Y X_a X_d) t) = $\int_0^t \left(\int_y^t f_{(X_a \mid Y=y)}(x)\ dx \right) f_Y(y)\ dy\ +$ $\int_0^t f_Y(y)\ F_{X_d}(y)\ dy$

ascertains that X and Y are greater than 0 but not equal to $+\infty$, `indep_var` ensures that the random variables X and Y are s-independent. The HOL function `real` is used in this context to type-cast the functions X and Y from extended-real data-type to real as explained previously. `distributed` is required to ensure that the random variable X has a density function f_x. The same applies for random variable Y. Finally, the functions `cont_CDF` and `measurable_CDF` ensure that the CDFs are continuous and measurable, respectively. More details about the verification steps of these theorems can be found in [17].

It is worth mentioning that the inputs of these gates can be s-dependent in case of having a common cause of failure. In this case the probability of intersection should be handled using conditional probabilities. For the WSP and CSP gates, it is required to deal with dependent events, as the failure of the main part affects the probability of failure of the spare part since it is activated after the failure of the main part. Therefore, their probability expressions involve conditional density functions. Table 3 shows the formally verified probability of failure theorems of CSP and WSP gates [17], where $f_{(X_a|Y=y)}$ is the conditional density function for the active state of the spare part (X_a) given that the main part (Y) failed at time y. The probability of failure for the WSP gate encompasses the behavior of both the CSP and the *after* event, since the behavior of the WSP includes the behavior of both [17]. For the expression of the WSP gate in Table 3, (`DISJOINT_WSP Y` X_a · X_d `t`) indicates that until time t, the spare part X can only fail in one of its states. f_{xy}, f_y and $f_{X_a \mid Y}$ correspond to the joint, marginal and conditional density functions of X and Y. `cond_density` defines the conditional density function $f_{X_a \mid Y=y}$ and ensures that X_a and Y are random variables that map from the probability space p to the Lebesgue-Borel measure (`lborel`) and with a joint density function f_{xy}. Finally, `den_gt0_ninfinity` is a predicate that ensures the allowable values of the density functions such as $0 \leq f_{xy}$. Since the WSP gate encompasses the behavior of the CSP gate and the after event, it inherits the required conditions for both expressions.

The given conditions in Table 3 are not explicitly mentioned as a part of the final theorems of each gate in [45]. Knowing the exact list of conditions for the theorems to hold is quite imperative to perform correct analysis, as missing any of these conditions may lead to some corner cases, which may falsify the results of the overall reliability analysis.

IV. DFT ANALYSIS METHODOLOGY

The proposed methodology, shown in Fig. 2, allows us to use HOL4 theorem prover to conduct both DFT qualitative and quantitative analysis.

As with any other DFT analysis tool, the analysis starts by a system description that can be used to build a DFT model and some reliability requirements that are related to the qualitative or quantitative analyses of the system. The main objective of the proposed methodology is to check if the given system model satisfies these requirements.

The first step in the proposed methodology is to develop a HOL formal DFT model of the given system. This step requires the formal definitions for DFT gates. Then, the structure function of the DFT's top event has to be reduced and we propose to formally verify this reduction based on a library of generic formally verified simplification theorems. This ensures that the reduced formal DFT model corresponds to the original DFT model. This verified reduced structure function is then used in the qualitative analysis to produce a reduced form of the cut sets and cut sequences that satisfy

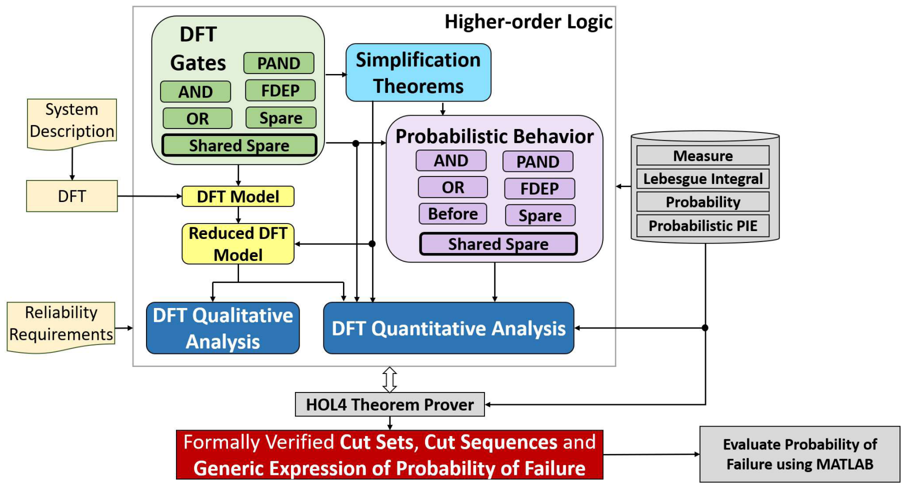


FIGURE 2. **DFT analysis methodology.**

the requirements. The cut sets can be defined as a group of sets, where each set has the inputs that their combined failure leads to the occurrence of the top event. The cut sequences, on the other hand, is a group of lists, where each list has a certain sequence of inputs that their failure in this particular sequence leads to the failure of the top event. The quantitative analysis is conducted by utilizing the reduced structure function to generate formally verified generic expressions of probability of failure. This last step requires invoking the available verified probabilistic behavior of DFT gates, which in turn requires DFT gates definitions and simplification theorems as well as some existing libraries in HOL4 such as the measure, Lebesgue integral, probability and probabilistic PIE. We are providing probabilistic expressions that are generic by using universally quantified probability density and distribution functions. The distribution and density functions and the variables in these generic expressions can be instantiated and evaluated in any other tool, such as MATLAB [46], to evaluate the probability of failure of a given system.

In order to make the methodology useful in practice for real-word systems, we need to consider spare gates with shared spare. Therefore, in the next section, we describe its formalization in HOL that enables the verification of a wider range of systems. Furthermore, in the following sections, we provide an illustration of the application of this methodology in the analysis of three small DFT examples, where the conditions required for correct analysis are clearly identified. To further highlight the need to identify the required conditions, we present a flaw in one of the published DFT algebras that is identified using HOL theorem proving. Finally, we describe the main blocks in the proposed methodology and how it is applied to perform the formal analysis of two safety-critical systems.

V. PROBABILISTIC BEHAVIOR OF A SPARE GATE WITH A SHARED SPARE

In real-world systems, sometimes it is required to have a spare that can replace one of two main parts in case of failure.

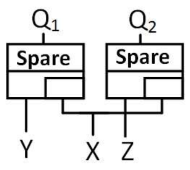


FIGURE 3. **Spare Gates with Shared Spare.**

This can be modeled as shown in Fig. 3 [45], where Y and Z are the main parts sharing the same spare X.

The output Q_1 of the first gate can occur in three different cases: (1) Y occurs, then the spare X occurs in its active state. (2) X occurs in its dormant state, then Y occurs. (3) Z occurs before Y, so it will be replaced with X, then Y occurs and finds no spare to replace it. Based on the definition of Q_1 in [9], we formally define it in HOL as [17]:

Definition 4: Shared Spare Q_1

```
⊢ ∀Y Z Xa Xd.
  shared_spare Y Z Xa Xd =
  Y · (Xd ◁ Y) + Xa · (Y ◁ Xa) + Y · (Z ◁ Y)
```

It is worth mentioning that the definition in [9] does not allow the simultaneous failures of the main parts and thus we use the same constraint.

Q_1 is represented as a sum of disjoint products in order to express its probability. This is accomplished by introducing the complement of an input event to be able to create the disjoint events. Thus Q_1 can be expressed as [45]:

$$\begin{aligned} Q_1 = \; & X_a \cdot (Y \lhd Z) \cdot (Z \lhd X_a) \\ & + Z \cdot (Y \lhd X_a) \cdot (X_a \lhd Z) \\ & + X_a \cdot (Y \lhd X_a) \cdot \cancel{Z} \\ & + Z \cdot (X_d \lhd Y) \cdot (Y \lhd Z) \\ & + Y \cdot (X_d \lhd Y) \cdot \cancel{Z} + Y \cdot (Z \lhd Y) \end{aligned} \tag{3}$$

where, $\cancel{Z}$ indicates the event when Z cannot happen. We formally verify this as:

Theorem 1:

```
⊢ ∀X_a X_d Y Z p t.
    rv_gt0_ninfinity [X_a; X_d; Y; Z] ∧
    (∀s. ALL_DISTINCT[Y s; X_a s; X_d s; Z s])∧
    DISJOINT_WSP Y X_a X_d t ∧
    DISJOINT_WSP Z X_a X_d t ∧
    (∀s. ((Z ◁ X_d)·(X_d ◁ Y)) s = NEVER s) ∧
    (∀s. ((Y ◁ X_d)·(X_d ◁ Z)) s = NEVER s)∧
    (∀s. ((X_a ◁ Y)·(X_a ◁ Z)) s = NEVER s) ⇒
    (DFT_event p Q_1 t =
     DFT_event p (Xa·(Y ◁ Z)·(Z ◁ X_a)) t ∪
     DFT_event p (Z·(Y ◁ X_a)·(X_a ◁ Z)) t ∪
     DFT_event p (X_a·(Y ◁ X_a)) t ∩
     (p_space p DIFF DFT_event p Z t) ∪
     DFT_event p (Z·(X_d ◁ Y)·(Y ◁ Z)) t ∪
     DFT_event p (Y·(X_d ◁ Y)) t ∩
     (p_space p DIFF DFT_event p Z t) ∪
     DFT_event p (Y·(Z ◁ Y)) t)
```

The first two conditions ensure that the time of occurrence of any event is always greater than or equal to 0 but not equal to $+\infty$ and are not equal. While the remaining conditions are required to ensure the proper behavior of the spare gates. For instance, the first two conditions mean that until time t, the spare part can fail in either the active or the dormant state. While the last two conditions indicate that the spare part cannot fail after any of the main parts while it is dormant. Since after the failure of one of the main parts, the spare part will be activated (working in the active state) and in case it fails it will be in the active state and not the dormant state. Similarly, the spare part cannot fail in its active state before the failure of both main parts, as it will be in its dormant state.

The difference between the expression in (3) and the verified expression in Theorem 1 is that we formally verified the DFT event of (3) based on the DFT event of the inputs. We decided to deal with the sets of the input events as there is no gate that can exhibit the behavior of $\cancel{Z}$ in the algebraic approach. This is due to the fact that in the algebraic approach, we are dealing with extended-real numbers and there is no possibility to implement a NOT gate using extended-reals. This means that instead of ANDing with $\cancel{Z}$, we intersect with the complement of `DFT_event p Z t`, i.e., `space - DFT_event p Z t` or more formally `p_space p DIFF DFT_event p Z t`. As a result, instead of verifying (3), we verified that the event of the left hand side is equal to the union of the events of the six products on the right hand side, and whenever we encounter $(\cdot \cancel{Z})$ we use (∩ `p_space p DIFF DFT_event p Z t`).

Since Q_1 is represented as the sum of disjoint products, the probability of Q_1 is expressed as the sum of the probabilities of the individual products as given in (4), as shown at the bottom of this page, [45]. We verified that these products are disjoint to be able to sum the individual probabilities. Some of these probabilistic expressions utilize our existing verified expressions for DFT gates, while the rest requires handling three iterated integrals while dealing with conditional density functions in addition to verifying the probability of a complement of a DFT event.

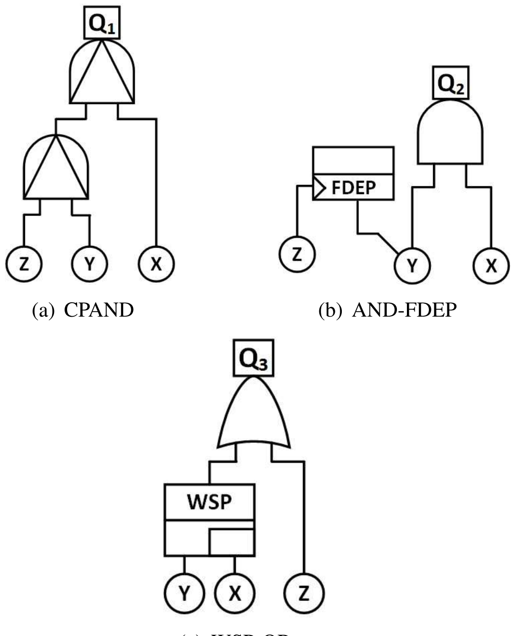
We have been able to verify (4), but as the final form of our verified theorem for (4) is quite long, we will explain some details about the proof and the theorem here and the complete theorem can be accessed from [47]. Since this theorem combines many previous formalized expressions, it requires the conditions for those expressions, such as having a conditional density of X_a given $Y = y$, having a density function for Y and Z. Also, the CDF of Z is measurable and continuous, beside the obvious conditions such as $0 \leq t$. The proof of the first term of the six terms in (4) is quite similar to the proof of the CSP gate. However, in this case, we are dealing with three iterated integrals which makes things a bit complex, since each time we need to prove that the single integral and the double iterated integrals are measurable. In addition, the s-independence of the random variables that correspond to the input events should be handled appropriately, i.e., Z should be s-independent of the joint random variables of (Y, X_a). The proof of the second term is conducted in a similar way to the first term since it consists of three iterated integrals with conditional density function. However, the density function lies this time in the inner integral. The proof of the third term is primarily based on proving that $Pr($`p_space p DIFF DFT_event p Z t`$) = 1 - F_Z(t)$. The proof of the fifth term also requires the same result. The fourth term corresponds to finding the probability of a cascaded PAND gate for three inputs (this will be explained in the following section). Finally, the last term corresponds to the probability of the after event. The size for this proof script is around 7700 lines.

$$
\begin{aligned}
Pr(Q_1)(t) = & \int_0^t \left(\int_y^t \left(\int_y^x f_Z(z)\, dz \right) f_{(X_a|Y=y)}(x)\, dx \right) f_Y(y)\, dy \\
& + \int_0^t \left(\int_0^z \left(\int_y^z f_{(X_a|Y=y)}(x)\, dx \right) f_Y(y)\, dy \right) f_Z(z)\, dz \\
& + (1 - F_Z(t)) \int_0^t \left(\int_y^t f_{(X_a|Y=y)}(x)\, dx \right) f_Y(y)\, dy \; + \; \int_0^t \left(\int_0^z f_Y(y) F_{X_d}(y)\, dy \right) f_Z(z)\, dz \\
& + (1 - F_Z(t)) \int_0^t f_Y(y) F_{X_d}(y)\, dy \; + \; \int_0^t f_Y(y) F_Z(y)\, dy
\end{aligned}
\tag{4}
$$

So far, we have presented our formalization for the probabilistic expressions of DFT gates. In the following section, we use some examples to show how the DFT definitions and theorems are utilized in the qualitative analysis of some DFTs.

VI. ANALYSIS OF DFT EXAMPLES

In this section, we apply our methodology to conduct both the qualitative and quantitative analyses of the DFT examples given in Fig. 4; (1) Cascaded PAND gates (CPAND) [5], (2) AND with FDEP gate (AND-FDEP) and (3) WSP with OR gate (WSP-OR).



(a) CPAND (b) AND-FDEP (c) WSP-OR

FIGURE 4. DFT examples.

A. QUALITATIVE ANALYSIS OF DFT EXAMPLES

Using our proposed methodology, we have been able to build a HOL formal DFT model and formally verify the reduction of the given DFT examples as in the following three theorems. We assume that all inputs are basic events, i.e., they cannot fail at the same time. This condition can be relaxed if we are modeling a system with a common cause of failure for the inputs.

Theorem 2: Reduced CPAND

```
⊢ ∀X Y Z.
   (∀s. ALL_DISTINCT [X s; Y s; Z s]) ⇒
   (PAND (PAND Z Y) X =
    X · (Z ◁ Y) · (Y ◁ X))
```

Theorem 3: Reduced AND-FDEP

```
⊢ ∀X Y Z. X·(FDEP Y Z) = X · Y + X · Z
```

Theorem 4: Reduced WSP-OR

```
⊢ ∀Y Xa Xd Z.
    (∀s. ALL_DISTINCT[Y s; Xa s; Xd s; Z s])⇒
   ((WSP Y Xa Xd) + Z =
    Xa · (Y ◁ Xa) + Y · (Xd ◁ Y) + Z)
```

As mentioned previously, each theorem can have a list of required conditions and a conclusion. For Theorem 2, the condition ensures that all random variables that represent system components are not equal, i.e., these random variables represent basic events. This is accomplished using the HOL4 function `ALL_DISTINCT`. The left hand side of the conclusion of this theorem represents the formal DFT expression for the given DFT, while the right hand side represents the verified reduced structure function. This applies also to Theorems 3 and 4. Using these verified reduced expressions, one can determine the cut sets and cut sequences to conduct the qualitative analysis. For example, the CPAND has only one sequence that can cause the occurrence of the top event which is $[Z; Y; X]$. Similarly, the AND-FDEP DFT has only two cut sets $\{X; Y\}$ and $\{X; Z\}$. Finally, the WSP-OR DFT has two cut sequences $[Y; X_a]$ and $[X_d; Y]$ and one cut set represented by the single element $\{Z\}$. This indicates that using our proposed methodology, we have been able to formally conduct the qualitative analysis and determine the cut sets and sequences of a given DFT.

B. PROBABILISTIC ANALYSIS OF DFT EXAMPLES

In this section, we perform the failure probabilistic analysis of the three DFT examples shown in Fig. 4.

We verify the probability of failure for the CPAND as in Theorem 5. The verification steps are similar to the *after* event. However, we are dealing now with three inputs instead of two. Hence, X is assumed to be s-independent of the joint random variable (Y, Z) using `indep_CPAND` in Theorem 5, where it is defined for X over the `lborel` measure and (Y, Z) over the two dimensional `lborel`.

Theorem 5: Probability of CPAND

```
⊢ ∀p X Y Z t fy fx.
  prob_space p ∧ 0 ≤ t ∧
  indep_CPAND X Y Z p ∧
  rv_gt0_ninfinity [X; Y; Z] ∧
  (∀s. ALL_DISTINCT [X s; Y s; Z s]) ∧
  distributed p lborel (λx. real (X x)) fx ∧
  distributed p lborel (λx. real (Y x)) fy ∧
  (∀y. 0 ≤ fy(y)) ∧ (∀x. 0 ≤ fx(x)) ∧
  cont_CDF p (λx. real (Z x)) x)) ⇒
  (prob p (DFT_event p (Q1) t) =
```

$$\int_0^t \left(\int_0^x f_Y(y)\ F_Z(y)\ dy\right) f_X(x)\ dx$$

We verify the probability of failure of AND-FDEP DFT as in Theorem 6. The main idea of this proof is to replace the FDEP gate by an OR gate as they are equivalent. Then, the probability of the union of two events, each of which is the intersection of two basic events, $\{X; Y\}$ and $\{X; Z\}$ is verified. These two events represent the cut sets of the AND-FDEP DFT. For this proof, it is required to ensure that the random variables are s-independent using `indep_vars3` and that the events of the two cut sets of the DFT are not equal using `All_distinct_events`.

Theorem 6: Probability of AND-FDEP

$\vdash \forall$X Y Z p t.
  rv_gt0_ninfinity [X; Y; Z] $\wedge$
  All_distinct_events p [X·Y; X·Z] t $\wedge$
  indep_vars3 X Y Z p $\Rightarrow$
  (prob p (DFT_event p (Q_2) t) =
   $F_X(t) \times F_Y(t) + F_X(t) \times F_Z(t) F_X(t) \times F_Y(t) \times F_Z(t)$

Finally, we verify the probability of the top event of the WSP-OR DFT as in Theorem 7. The top event is composed of the union of the WSP event and the basic event Z. Hence, the final form of the probability is the probability of the union of two events; the WSP and Z. Therefore, it is required to include the conditions needed for expressing the probability of the WSP event in the list of assumptions, and ensure that the WSP event is s-independent of event Z using `indep_var_set_WOR`.

Theorem 7: Probability of WSP-OR

$\vdash \forall$Y X_a X_d Z p t f_{xy} f_y $f_{X_a|Y}$.
  DISJOINT_WSP Y X_a X_d t $\wedge$
  ($\forall$s. ALL_DISTINCT [Y s; X_a s; X_d s; Z s]) $\wedge$
  All_distinct_events p [WSP Y X_a X_d; Z] t $\wedge$
  rv_gt0_ninfinity [Y; X_d; X_a; Z] $\wedge$ 0 $\leq$ t $\wedge$
  ($\forall$y. cond_density lborel lborel p
      (λs. real (X_a s)) (λs. real (Y s))
      y f_{xy} f_y $f_{X_a|Y}$) $\wedge$
  den_gt0_ninfinity f_{xy} f_y $f_{X_a|Y}$ $\wedge$
  cont_CDF p (λs. real (X_d s)) $\wedge$
  measurable_CDF p (λs. real (X_d s)) $\wedge$
  indep_var_set_WOR Y X_a X_d Z p t $\Rightarrow$
  (prob p (DFT_event p (Q_3) t) =

$$\int_0^t \left(\int_y^t f_{(X_a|Y=y)}(x)\,dx\right) f_Y(y)\,dy + \int_0^t f_Y(y)\,F_{X_d}(y)\,dy + F_Z(t) - \left(\int_0^t \left(\int_y^t f_{(X_a|Y=y)}(x)\,dx\right) f_Y(y)\,dy + \int_0^t f_Y(y)\,F_{X_d}(y)\,dy\right) \times F_Z(t)$$

After having the verified generic expressions for the probability of failure of the three examples, these expressions can be used to evaluate the probability of failure for any integrable distribution functions that represent the failure distribution of the system components. For example, assuming exponential distributions for the inputs with failure rates: 2×10^{-2}, 3×10^{-3}, and 1×10^{-2} for X, Y and Z, respectively, we evaluated the probability of failure using MATLAB until 400 working hours with a dormancy factor of 0.1. The results are shown in Fig. 5.

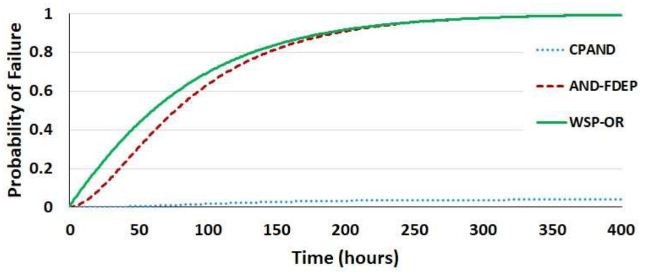


FIGURE 5. Probability of Failure of CPAND, AND-FDEP and WSP-OR.

To sum up, in this section, we have demonstrated the application of our methodology in the formal probabilistic analysis of three DFT examples, which included listing the required conditions to formally conduct the proofs. Next, we show the importance of identifying the required conditions by listing a flaw that we detected in a published DFT algebra.

VII. FLAW DETECTED IN THE ALGEBRAIC APPROACH OF [18]

In order to emphasize on the importance of formally verifying the underlying math of the algebraic approach and the significance of knowing the required conditions for the analysis results to be valid, we provide more details regarding the flaw in one of the algebraic approaches mentioned in Section I. In [18], a new simple algebra is introduced that provides definitions and simplification theorems for DFTs. We have been able to identify an error in one of the simplification theorems, which is the distributivity property of the sequence operator over the OR operator, i.e.,

$$A.(B+C) = A.B + A.C \tag{5}$$

where . and + represent the sequence and OR operators, respectively. The sequence operator indicates that its output occurs if the input events occur in sequence from left to right, i.e., the time of occurrence of the left input event is less than that of the right input event. While the OR operator is represented by the minimum time of occurrence for the input events. Now, assuming that the time of occurrence of the input events A, B, and C are d_A, d_B and d_C, respectively, the left hand side of (5) occurs only if d_A is less than $min(d_B, d_C)$. While, the right hand side of the same equation occurs if $d_A < d_B$ or $d_A < d_C$. This property fails to hold when $d_A > min(d_B, d_C)$ but at the same time $d_A < max(d_B, d_C)$, i.e., d_A falls in the middle between d_B and d_C. In this particular case, the left hand side of (5) will not occur because d_A is not less than the minimum of d_B and d_C, however, one of the terms of the right hand side occurs. For this property to hold, it is required to have the condition $d_A < min(d_B, d_C)$. We have been able to identify this flaw using theorem proving, as the property was not verifiable for the aforementioned case unless this particular condition is added. As a consequence, using this property without the required condition in any application, including the application part of the mentioned paper [18], would lead to erroneous results, which is serious specially for safety-critical systems that cannot tolerate any error in the analysis. These findings emphasize on the importance of having a formal rigorous framework for DFT analysis, that would allow building the analysis on a sound basis to be used with systems, specially the safety-critical ones. In the next section, we will apply our methodology on two real-world case studies.

VIII. CASE STUDIES

In this section, we apply our methodology on two safety-critical systems, i.e., a drive-by-wire system to control the brakes and throttle systems of modern vehicles [19] and a cardiac assist system that provides care to patients with heart failure [20]. The analysis of these systems should be carefully conducted as any error may lead to even the loss of life in extreme cases.

In order to facilitate the analysis of these systems and any similar systems, we verified several generic properties that can be used to reduce the manual interaction of the verification engineer in the theorem proving related tasks. For example, for any group of s-independent random variables, we verified that the probability of the preimage of any two random variables out of the original set equals to the multiplication of the individual probabilities as :

Theorem 8:

```
⊢ ∀p M X ii A s t. s ≠ t ∧ prob_space p ∧
   indep_vars p M X ii ∧ {s; t} ⊂ ii ∧
   (∀i. i ∈ {s; t} ⇒
         A i ∈ measurable_sets (M i))⇒
   (prob p
     (PREIMAGE (X s) (A s) ∩ p_space p ∩
     (PREIMAGE (X t) (A t) ∩ p_space p)) =
    prob p (PREIMAGE(X s)(A s) ∩ p_space p)*
    prob p (PREIMAGE(X t)(A t) ∩ p_space p))
```

The formal DFT analysis now requires proving the required conditions for this property to hold only. As an example, consider that we have a group of 10 random variables, and we need to prove that the probability of the preimages of the 6th and the 8th random variables equals the multiplication of their individual probabilities. Therefore, in Theorem 8, $s = 6$, $t = 8$ and ii equals the set of numbers from $0 - 9$. We just need to verify the following properties for this proof:

- $6 \neq 8$
- $\{6; 8\} \subset \{0; 1; 2; 3; 4; 5; 6; 7; 8; 9\}$
- The sets of the preimages are measurable

These requirements can be easily verified using various built-in arithmetic tactics in HOL4. Similarly, we verified the same property for up to ten random variables out of a group of independent random variables. These properties are very helpful in the verification process of the probabilistic analysis of DFTs, in particular when applying the probabilistic PIE. We also verified several additional properties that allow the direct usage of the PIE in its final form with a system that can be represented as the union of six elements as the behavior of both case studies can be represented as the union of six events. However, our formalization can be extended easily to verify larger systems, as the flow of the proofs will remain the same but will extend to a larger number of inputs.

A. FORMAL VERIFICATION OF DRIVE-BY-WIRE SYSTEM

The DFT of the drive-by-wire (DBW) system is shown in Fig. 6 [19]. We chose to analyze the brake and the throttle parts of this system, which consists of the following parts: the brakes control unit (BC), the throttle (TF), two sensors; the brake sensor (BS) and the throttle sensor (TS), the engine (EF) and finally the primary central control (PC) unit with its spare part (SC_d and SC_a for both the dormant and active

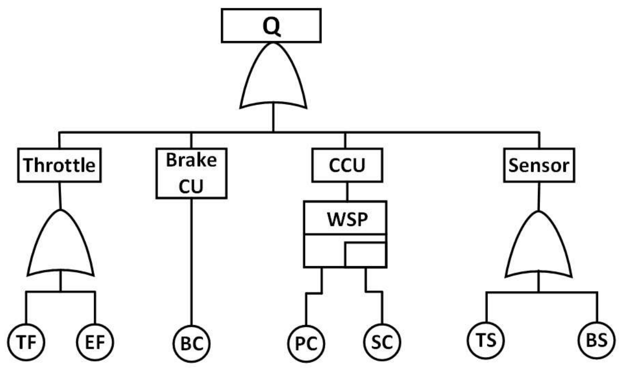


FIGURE 6. DFT of Drive-by-wire system.

states, respectively). We modeled the spare part of the central control unit as a warm spare, as this is the general case for the spare. In addition, this is the most convenient way to model it as the spare control unit can be working in the sleep mode, and it will only be activated after the main unit fails.

We proceed with the analysis of the drive-by-wire system following the steps outlined in our proposed methodology. We first start by verifying the reduction of the structure function of the top event to utilize it in both the qualitative and quantitative analyses.

Lemma 1: Reduced DBW

```
⊢ ∀BS TS PC SCa SCd BC EF TF.
   (∀s. ALL_DISTINCT
     [BS s; TS s; PC s; SCa s; SCd s;
      BC s; EF s; TF s]) ⇒
     ((TF + EF) + WSP PC SCa SCd + BC +
      (TS + BS) =
      TF + EF + BC + SCa ·(PC ◁ SCa) +
      PC · (SCd ◁ PC) + TS + BS)
```

From this expression, we can find a reduced form of the cut sequences:

$$[PC,\ SC_a], [SC_d, PC]$$

which means that the top event can fail due to two different sequences of input failures. The first one is the failure of the main control unit (PC) followed by the failure of the spare in its active state (SC_a). The second sequence is when the spare part fails in its dormant state (SC_d) followed by the failure of the main control unit.

In a similar way, a reduced form of the cut sets can be extracted from the reduced top event expression as:

$$\{TF\},\ \{EF\},\ \{BC\},\ \{TS\},\ \{BS\}$$

We choose to use a single event for the WSP as this will reduce the intermediate steps required to reach our final goal for the probabilistic expression and would result in expressing the top event as the union of six events. We verify that the `DFT_event` of the drive-by-wire is equal to the union of six events as:

Lemma 2: DBW Union of Events

```
⊢ ∀BS TS PC SCa SCd BC EF TF p t.
```

```
DFT_event p ((TF + EF) + WSP PC SC_a SC_d +
     BC + (TS + BS)) t =
union_list
 [DFT_event p TF t; DFT_event p EF t;
  DFT_event p (WSP PC SC_a SC_d) t;
  DFT_event p BC t;
  DFT_event p TS t; DFT_event p BS t]
```

We apply the probabilistic PIE to perform the formal quantitative analysis of the top event, by incorporating the existing verified properties. We verify the probabilistic failure expression of the drive-by-wire system as Theorem 9. In the following, we are presenting the formalization in mixed formal and standard math notations to make the results more understandable for the reader.

Theorem 9: Probability of Failure of DBW

```
⊢ ∀BS TS PC SC_a SC_d BC EF TF p t
  f_PC f_(SC_a|PC) f_SC_aPC. 0 ≤ t ∧
  All_distinct_events p
    [TF; EF; BC; WSP PC SC_a SC_d; BS; TS] t ∧
  rv_gt0_ninfinity [BS; TS; PC; SC_a; SC_d;
                    BC; EF; TF] ∧
  DISJOINT_WSP PC SC_a SC_d t ∧
  (∀y. cond_density lborel lborel p
      (λs. real (SC_a s)) (λs. real (PC s))
       y f_SC_aPC f_PC f_(SC_a|PC)) ∧
  den_gt0_ninfinity f_SC_aPC f_PC f_(SC_a|PC) ∧
  cont_CDF p (λs. real (SC_d s)) ∧
  measurable_CDF p (λs. real (SC_d s)) ∧
  indep_vars_sets_drive
   [BS; TS; PC; SC_a; SC_d; BC; EF; TF] p t ⇒
  (prob p (DFT_event p Q_DBW t) =
  F_TF(t) + F_EF(t) + F_BC(t) +
```

$$\int_0^t f_{PC}\ (pc) \times \int_{pc}^t f_{(SC_a|PC=pc)}\ (sc_a)\ dsc_a\ dpc\ +$$

$$\int_0^t f_{PC}(pc)\ \times\ F_{SC_d}(pc)\ dpc + F_{BS}(t) + F_{TS}(t) -$$

$$\ldots + \ldots - F_{TF}(t) \times F_{EF}(t) \times F_{BC}(t) \times$$

$$\Bigg[\bigg(\int_0^t f_{PC}(pc) \times \bigg(\int_{pc}^t f_{(SC_a|PC=pc)}\ (sc_a)\ dsc_a\bigg)\ dpc\bigg) + \int_0^t f_{PC}(pc)\ \times\ F_{SC_d}(pc)\ dpc\Bigg] \times F_{BS}(t) \times F_{TS}(t))$$

where `All_distinct_events` ensures that all event sets are distinct. As listed earlier, since the events of the WSP are disjoint, we used the WSP event directly to reduce the proof steps. It is necessary that all random variables that represent the input events to be positive or equal to 0, since they represent the time of failure. This condition is added by `rv_gt0_ninfinity`. It is also required to ensure the proper behavior of the WSP by adding the condition `DISJOINT_WSP PC` SC_a SC_d `t`, which ascertains that the events of the WSP are disjoint, i.e., until time t, the spare part can fail in one of its states only. A conditional density $f_{(SC_a|PC)}$ of SC_a given that $PC = pc$ is defined using `cond_density`. The function `indep_vars_sets_drive` adds the condition that the input events and their sets are s-independent, and finally we need to ensure that the CDF of SC_d is continuous and measurable. It is worth mentioning that since the union list of the drive-by-wire system has six events, applying the PIE results in the generation of 63 different terms, and a truncated version of the final expression is given above.

The verification engineer working on the analysis of the drive-by-wire system just needs to ensure that the mentioned conditions hold in order to use the results of the analysis. After formally ensuring that the probability of failure expression is correct, this expression can be used to evaluate the probability of failure using any tool with any distribution and density functions that satisfy the listed conditions. Assuming exponential distributions for the inputs with failure rates as listed in Table 4 [19], we evaluated the probability of failure using MATLAB until 1,000,000 working hours with dormancy factor of 0.5, as shown in Fig. 7. The proof script for the drive-by-wire system is around 4950 lines long.

TABLE 4. Failure rates for the DBW system ($\times 10^{-7}$).

TF	EF	BC	PC	SC	TS	BS
1	4	5	2	3	1	2

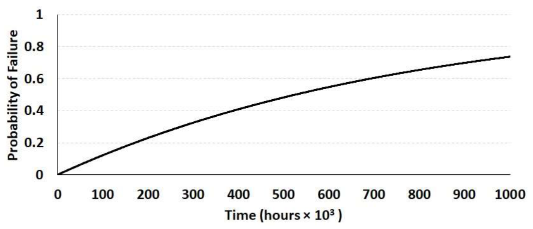


FIGURE 7. Probability of Failure of the Drive-by-wire System.

B. FORMAL VERIFICATION OF CARDIAC ASSIST SYSTEM

The DFT for the cardiac assist system (CAS) is shown in Fig. 8 [20]. The system consists of three sub-systems: pumps, motors and CPUs. There are two main pumps *PA* and *PB*. After the failure of one of these pumps, a shared spare *PS* replaces the failed one. There are two motors *MA* and the spare *MB* and a switch *MS*. The motor sub-system fails if *MS* then *MA* fail in sequence or if *MA* and the spare *MB* fail. Finally, there is one main CPU *P* and its spare *B*. Both CPUs are functionally dependent on the union of a crossbar switch (*CS*) and the system supervisor (*SS*).

We consider here different variations of spare gates, to make this case study more general and inclusive to all the formalized gates, as shown in Fig. 8. A simplified version of this DFT, where we assumed that all spare gates are HSPs gates, was verified in [17]. However, the variations that we assume here for the spares allow modeling and verifying the

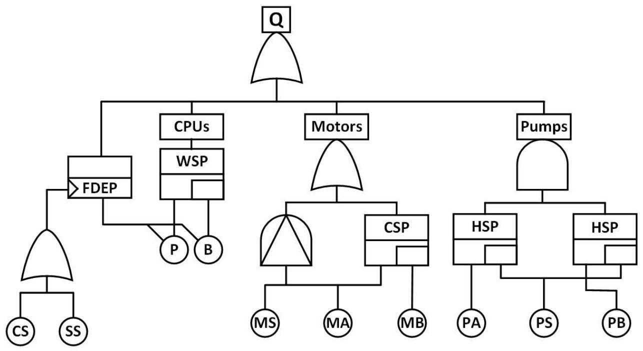


FIGURE 8. **DFT of Cardiac Assist System.**

probability of failure of the given system, while the independence of some of the events does not hold anymore.

We start first by verifying a reduced version of the structure function of the top event. This enables us to perform both the qualitative and quantitative analyses on a reduced function.

We formally verify this reduction in HOL4 as:

Lemma 3: Reduced CAS

```
⊢ ∀PA PB PS MS MA MB Ba Bd CS SS P.
 (∀s. ALL_DISTINCT
    [MA s; MS s; PA s; PB s; PS s;
     MB s; P s; Bd s; Ba s; CS s; SS s] ∧
 (Ba ◁ P = NEVER) ⇒
 ((shared_spare PA PB PS PS) ·
  (shared_spare PB PA PS PS) +
  ((PAND MS MA) + (CSP MA MB)) +
   (WSP (FDEP (CS + SS) P)(FDEP (CS + SS) Ba)
        (FDEP (CS + SS) Bd)) =
  CS + SS + MA · (MS ◁ MA) + MB · (MA ◁ MB) +
  Ba · (P ◁ Ba) + P · (Bd ◁ P) + PA · PB · PS)
```

where ALL_DISTINCT ensures that the inputs do not occur at the same time, and $(B_a \lhd P = \text{NEVER})$ ascertains that the spare part *B* in its active state cannot fail before *P*. This ensures the proper behavior of the WSP gate. It is worth mentioning that such condition is not required for the HSP gate as the spare part exhibits the same failure behavior in both of its states.

It is important to mention that since the inputs of the WSP gate are functionally dependent on the union of *CS* and *SS*, we use (FDEP (CS+SS) P), (FDEP (CS+SS) B_a) and (FDEP (CS+SS) B_d) for the main, the spare in its active state and the spare in its dormant state, respectively.

From the verified reduced top event, we can conclude a reduced form of cut sequences as follows:

$$[MS; MA], [MA; MB], [P; B_a], [B_d; P] \tag{6}$$

Moreover, a reduced form of the cut sets is deducted as:

$$\{CS\},\ \{SS\},\ \{PA, PB, PS\} \tag{7}$$

In a similar way to the drive-by-wire system, we verify that the DFT_event of the cardiac assist system equals the union of events as:

Lemma 4: CAS Union of Events

```
⊢ ∀PA PB PS MS MA MB CS SS P Ba Bd p t.
 DFT_event p
 (CS + SS + MA · (MS ◁ MA) + MB · (MA ◁ MB) +
   Ba · (P ◁ Ba) + P · (Bd ◁ P) +
   PA · PB · PS) t =
 union_list
  [DFT_event p CS t; DFT_event p SS t;
   DFT_event p (MA · (MS ◁ MA)) t;
   DFT_event p (MB · (MA ◁ MB)) t;
   DFT_event p
     (Ba · (P ◁ Ba) + P · (Bd ◁ P)) t;
   DFT_event p (PA · PB · PS) t]
```

Verifying a generic expression of the probability of failure for the cardiac assist system requires dealing with different conditions of s-independence for the input events, where we considered different configurations for the spare gates in the cardiac assist system from [17]. In particular, the outputs of the PAND and the CSP gates are no longer independent because of having *MA* in common. Therefore, it is required to use conditional probabilities to verify the probability of intersection that results from applying the probabilistic PIE. We verify the probability of failure of this system in HOL4:

Theorem 10: Probability of Failure of CAS

```
⊢ ∀CS SS MA MS MB P Ba Bd PA PB PS p t
  fMA fBaP fP fBa|P fMBMA fMB|MA fMS.
 0 ≤ t ∧ prob_space p ∧ (Ba ◁ P = NEVER)∧
 DISJOINT_WSP P Ba Bd t ∧
 ALL_DISTINCT_RVg
   [PA; PB; PS; MS; MA; MB; CS; SS; P;
    Ba; Bd] p t ∧
 indep_vars_setsg
   [PA; PB; PS; MS; MA; MB; CS; SS; P;
    Ba; Bd] p t ∧
 (∀y. cond_density lborel lborel p
    (λs. real (Ba s)) (λs. real (P s))
     y fBaP fP fBa|P) ∧
 (∀y. cond_density lborel lborel p
    (λs. real (MB s)) (λs. real (MA s))
    y fMBMA fMA fMB|MA ∧
 den_gt0_ninfinity fBaP fP fBa|P ∧
 den_gt0_ninfinity fMBMA fMA fMB|MA ∧
 cont_CDF p (λs. real (Bd s)) ∧
 measurable_CDF p (λs. real (Bd s)) ∧
 (∀z. 0 ≤ fMS(z)) ∧ (∀x. fMBMA(x) ≠ PosInf)∧
 distributed p lborel (λx. real (MS x)) fMS∧
 cont_CDF p (λs. real (MS s)) ∧
 measurable_CDF p (λs. real (MS s)) ⇒
```

$$
\begin{aligned}
&\texttt{(prob p (DFT\_event p } Q_{CAS} \texttt{ t)=}\\
&F_{CS}(t) + F_{SS}(t) + \int_0^t f_{MA}(ma) \times F_{MS}(ma)\, dma +\\
&\int_0^t f_{MA}(ma) \times \left(\int_{ma}^t f_{MB|MA=ma}(mb)\ dmb \right) dma +\\
&\left(\int_0^t f_P(pp) \times \left(\int_{pp}^t f_{B_a|P=pp}(b_a)\ db_a \right) dpp +\right.\\
&\left. \int_0^t f_P(pp) \times F_{B_d}(pp)\ dpp \right) +\\
&F_{PA}(t) \times F_{PB}(t) \times F_{PS}(t) - \ldots + \ldots -\\
&F_{CS} \times F_{SS} \times\\
&\int_0^t f_{MA}(ma) \times F_{MS}(ma) \times \left(\int_{ma}^t f_{MB|MA=ma}(mb)\ dmb \right) \times\\
&\left[\int_0^t f_P(pp) \times \left(\int_{pp}^t f_{B_a|P=pp}(b_a)\ db_a \right) dpp +\right.\\
&\left. \int_0^t f_P(pp) \times F_{B_d}(pp)\ dpp \right] \times\\
&F_{PA}(t) \times F_{PB}(t) \times F_{PS}(t)
\end{aligned}
$$

where $(B_a \lhd P = \texttt{NEVER}) \wedge \texttt{DISJOINT\_WSP P } B_a\ B_d \texttt{ t}$ are required to ensure that the spare part B_a cannot fail before the main part P and that the events of the WSP are disjoint, i.e., until time *t*, the spare part can fail in either the dormant or the active states. ALL_DISTINCT_RVg is a predicate required to ascertain that the inputs and their event sets are not equal and that the inputs are greater than or equal to 0 but not equal to $+\infty$. indep_vars_setsg ensures the s-independence of the random variables and the event sets. It is also required to define conditional density functions for $f_{B_a|P}$ and $f_{MB|MA}$ using cond_density. den_gt0_ninfinity ensures the proper values for the joint, marginal and conditional density functions that are used with cond_density. For example, the conditional density functions cannot be equal to 0. In addition, the density functions cannot equal to $+\infty$. It is also required to ensure that the CDFs of random variables B_d and MS are continuous and measurable using cont_CDF and measurable_CDF, respectively. distributed p lborel (λx. real (MS x)) f_{MS} is used to indicate that MS has a density function f_{MS}. It is worth mentioning again that the usage of the function real is required here as the random variables return extreal, while they are required to be used with the Lebesgue-Borel measure, which is defined over the real line. The first six elements of the conclusion of Theorem 10 represent the probability of the individual terms of the union list of Lemma 4, which result from applying the probabilistic PIE. While the rest of the elements represent the probability of the intersection of all the combination of the events. The last term represents the probability of the intersection of the six elements of the cardiac assist system. It took around 8000 lines of proof script to prove the probability of failure of the cardiac assist system.

As with the drive-by-wire system, we assume exponential distributions for the inputs of the cardiac assist system with failure rates listed in Table 5 [45]. We evaluated the probability of failure for this generic expression using MATLAB with a dormancy factor of 0.5 for the spare part *MB* until 400,000 working hours, as shown in Fig. 9.

TABLE 5. Failure rates ($\times 10^{-6}$).

CS	SS	P	B	MS	MA	MB	PA	PS	PB
1	2	4	4	1	5	5	5	5	5

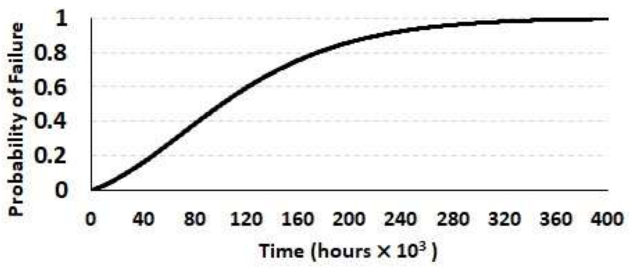


FIGURE 9. Probability of Failure of the Cardiac Assist System.

We have illustrated in this section the application of our proposed methodology to conduct the formal failure analysis of the drive-by-wire and the cardiac assist systems. We have created the HOL formal DFT models for these systems and verified a reduced form of the structure functions utilizing the verified simplification theorems. We then conducted the qualitative and the probabilistic analyses to generate formally verified expressions of probability of failure. Building upon the expressive and sound nature of HOL theorem proving, generic intermediate lemmas are verified that are valid for the analysis of systems similar to the drive-by-wire and the cardiac assist systems. Leveraging upon the current formalization of DFTs, the existing lemmas and theorems can be extended to analyze more complex systems. In addition, the results obtained using our methodology, in particular the generic expressions, cannot be obtained formally using a PMC. Moreover, our proposed methodology overcomes the vulnerability of the paper-and-pencil analysis results due to human errors, as it inherits the soundness of HOL theorem proving. Although, providing the formalization of this methodology is costly in terms of time and lines of script, the results obtained are usable by the verification engineer without the need to go through all the steps of the formalization. The verification engineer only needs to use the results of the theorems after ensuring that all the required conditions hold, which provides him/her with a formal proof that the analysis results can apply to his system if the conditions are met.

IX. CONCLUSION

In this paper, we proposed a novel methodology to conduct the formal analysis of DFTs using HOL theorem proving. This methodology supports accurate qualitative and quantitative analyses of DFTs based on the soundness and expressive nature of HOL theorem proving. The proposed approach represents a complementary one that overcomes the shortcomings of ordinary DFT analysis methods, such as the approximation based nature of simulation that results from sampling. Based on the proposed methodology, we identified an error in one of the proposed DFT analysis approaches.

In this paper, we described our HOL formalization of DFT gates, simplification theorems and probabilistic failure behavior of these gates. In addition, we presented our new HOL formalization of the probabilistic failure behavior of the output of a spare gate with a shared spare, which represents another novel contribution. Furthermore, we have been able to verify several intermediate generic lemmas that can be used in the analysis of DFTs based on our formalization in a quite straightforward manner. Finally, we utilized our formalization to conduct both the qualitative analysis in the form of reduced cut sets and cut sequences and the quantitative analysis of two safety-critical systems to generate generic expressions of probability of failure that can be instantiated later to evaluate the probability of failure. As a future work, we plan to develop some machine learning algorithms in order to facilitate the user interaction in the theorem proving process for the formal DFT analysis.

REFERENCES

[1] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools," *Comput. Sci. Rev.*, vols. 15–16, pp. 29–62, Mar. 2015.
[2] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback, *Fault Tree Handbook with Aerospace Applications: NASA Office of Safety and Mission Assurance*. Washington, DC, USA: NASA, 2002.
[3] J. Dugan, S. J. Bavuso, and M. A. Boyd, "Fault trees and sequence dependencies," in *Proc. Annu. Rel. Maintainability Symp.*, Jan. 1990, pp. 286–293.
[4] H. Boudali, P. Crouzen, and M. Stoelinga, "Dynamic fault tree analysis using input/output interactive Markov chains," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2007, pp. 708–717.
[5] G. Merle, J. M. Roussel, J. J. Lesage, and A. Bobbio, "Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events," *IEEE Trans. Rel.*, vol. 59, no. 1, pp. 250–261, Mar. 2010.
[6] C. Mooney, *Monte Carlo Simulation*. Newbury Park, CA, USA: Sage,, 1997.
[7] L. Pullum and J. Dugan, "Fault tree models for the analysis of complex computer-based systems," in *Proc. Annu. Rel. Maintainability Symp.*, Jan. 1996, pp. 200–207.
[8] L. Xing and S. Amari, *Binary Decision Diagrams and Extensions for System Reliability Analysis*. Hoboken, NJ, USA: Wiley, 2015.
[9] G. Merle, "Algebraic modelling of dynamic fault trees, contribution to qualitative and quantitative analysis," Ph.D. dissertation, ENS, Cachan, France, 2010.
[10] G. Merle, J.-M. Roussel, J.-J. Lesage, V. Perchet, and N. Vayatis, "Quantitative analysis of dynamic fault trees based on the coupling of structure functions and Monte Carlo simulation," *Qual. Rel. Eng. Int.*, vol. 32, no. 1, pp. 7–18, 2016.
[11] O. Hasan and S. Tahar, "Formal Verification Methods," in *Encyclopedia of Information Science and Technology*. Hershey, PA, USA: IGI Global, 2015, pp. 7162–7170.
[12] C. Baier and J. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.
[13] M. J. Gordon and T. F. Melham, *Introduction to HOL: A Theorem-Proving Environment for Higher-Order Logic*. Cambridge, U.K.: Univ. Cambridge Press, 1993.
[14] C. Dehnert, S. Junges, J. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," in *Computer Aided Verification* (Lecture Notes in Computer Science), vol. 10472. Cham, Switzerland: Springer, 2017, pp. 592–600.
[15] M. Ghadhab, S. Junges, J. Katoen, M. Kuntz, and M. Volk, "Safety analysis for vehicle guidance systems with dynamic fault trees," *Rel. Eng. Syst. Saf.*, vol. 186, pp. 37–50, Jul. 2019.
[16] Y. Elderhalli, O. Hasan, W. Ahmad, and S. Tahar, "Formal dynamic fault trees analysis using an integration of theorem proving and model checking," in *NASA Formal Methods* (Lecture Notes in Computer Science), vol. 10811. Cham, Switzerland: Springer, 2018, pp. 139–156.
[17] Y. Elderhalli, W. Ahmad, O. Hasan, and S. Tahar, "Probabilistic analysis of dynamic fault trees using HOL theorem proving," *J. Appl. Logics*, vol. 6, no. 3, p. 469, 2019.
[18] J. Ni, W. Tang, and Y. Xing, "A simple algebra for fault tree analysis of static and dynamic systems," *IEEE Trans. Rel.*, vol. 62, no. 4, pp. 846–861, Dec. 2013.
[19] A. Altby and D. Majdandzic, "Design and implementation of a fault-tolerant drive-by-wire system," M.S thesis, Chalmers Univ. Technol., Göteborg Sweden, 2014.
[20] H. Boudali, P. Crouzen, and M. Stoelinga, "A rigorous, compositional, and extensible framework for dynamic fault tree analysis," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 2, pp. 128–143, Apr. 2010.
[21] K. J. Sullivan, J. B. Dugan, and D. Coppit, "The Galileo fault tree analysis tool," in *Proc. 29th Annu. Int. Symp. Fault-Tolerant Comput.*, Jun. 1999, pp. 232–235.
[22] F. Arnold, A. Belinfante, F. V. der Berg, D. Guck, and M. Stoelinga, "DFTCalc: A tool for efficient fault tree analysis," in *Computer Safety, Reliability, and Security* (Lecture Notes in Computer Science), vol. 8153. Cham, Switzerland: Springer, 2013. pp. 293–301.
[23] X. Song, Z. Zhai, P. Zhu, and J. Han, "A stochastic computational approach for the analysis of fuzzy systems," *IEEE Access*, vol. 5, pp. 13465–13477, 2017.
[24] D. Codetta-Raiteri, "The conversion of dynamic fault trees to stochastic Petri nets, as a case of graph transformation," *Electron. Notes Theor. Comput. Sci.*, vol. 127, no. 2, pp. 45–60, 2005.
[25] G. Franceschinis, S. Donatelli, and R. Gaeta, "GreatSPN 2.0: Graphical editor and analyzer for timed and stochastic Petri Nets," in *Proc. Petri Nets*, 2000, p. 43.
[26] *PRISM*. (2019). [Online]. Available: http://www.prismmodelchecker.org/
[27] M. Volk, S. Junges, and J.-P. Katoen, "Fast dynamic fault tree analysis by model checking techniques," *IEEE Trans Ind. Informat.*, vol. 14, no. 1, pp. 370–379, Jan. 2018.
[28] J. Hurd, "Formal verification of probabilistic algorithms," Ph.D. dissertation, Dept. Comput. Sci., Univ. Cambridge, Cambridge, U.K., 2002.
[29] O. Hasan, "Formal probabilistic analysis using theorem proving," Ph.D. dissertation, Dept. Elect. Comput. Eng., Concordia Univ. Montreal, QC, Canada, 2008.
[30] T. Mhamdi, "Information-theoretic analysis using theorem proving," Ph.D. dissertation, Dept. Elect. Comput. Eng., Concordia Univ., Montreal, QC, Canada, 2012.
[31] J. Hòlzl, "Construction and stochastic applications of measure spaces in higher-order logic," Ph.D. dissertation, Dept. Comput. Sci., Technische Universität München, Berlin, Germany, 2012.
[32] N. Abbasi, O. Hasan, and S. Tahar, "Formal lifetime reliability analysis using continuous random variables," in *Proc. Int. Workshop Logic, Lang. Inf. Comput.* Cham, Switzerland: Springer, 2010, pp. 84–97.
[33] O. Hasan, S. Tahar, and N. Abbasi, "Formal reliability analysis using theorem proving," *IEEE Trans. Comput.*, vol. 59, no. 5, pp. 579–592, May 2010.
[34] W. Ahmad and O. Hasan, "Formalization of fault trees in higher-order logic: A deep embedding approach," in *Dependable Software Engineering: Theories, Tools, and Applications* (Lecture Notes in Computer Science), vol. 9984. Cham, Switzerland: Springer, 2016, pp. 264–279.
[35] W. Ahmad and O. Hasan, "Towards formal fault tree analysis using theorem proving," in *Intelligent Computer Mathematics* (Lecture Notes in Computer Science), vol. 9150. Cham, Switzerland: Springer, 2015, pp. 39–54.
[36] *HOL4*. (2019). [Online]. Available: https://hol-theorem-prover.org/
[37] *Isabelle*. (2019). [Online]. Available: https://isabelle.in.tum.de/
[38] *Coq*. (2019). [Online]. Available: https://coq.inria.fr/
[39] T. Mhamdi, O. Hasan, and S. Tahar, "On the formalization of the Lebesgue integration theory in HOL," in *Interactive Theorem Proving* (Lecture Notes in Computer Science) vol. 6172. Cham, Switzerland: Springer, 2010, pp. 387–402.
[40] T. Mhamdi, O. Hasan, and S. Tahar, "Formalization of entropy measures in HOL," in *Interactive Theorem Proving* (Lecture Notes in Computer Science), vol. 6898. Cham, Switzerland: Springer, 2011, pp. 233–248.
[41] M. Qasim, O. Hasan, M. Elleuch, and S. Tahar, "Formalization of normal random variables in HOL," in *Intelligent Computer Mathematics* (Lecture Notes in Computer Science), vol. 9791, Springer, 2016, pp. 44–59.

[42] H. Boudali and J. Bechta Dugan, "A continuous-time Bayesian network reliability modeling, and analysis framework," *IEEE Trans. Rel.*, vol. 55, no. 1, pp. 86–97, Mar. 2006.
[43] G. Merle, J.-M. Roussel, and J.-J. Lesage, "Improving the efficiency of dynamic fault tree analysis by considering gate FDEP as static," in *Proc. Eur. Saf. Rel. Conf.*, 2010, p. pp–845.
[44] Y. Elderhalli, O. Hasan, W. Ahmad, and S. Tahar, "Dynamic fault trees analysis using an integration of theorem proving and model checking," Dec. 2017, *arXiv:1712.02872*. [Online]. Available: https://arxiv.org/abs/1712.02872
[45] G. Merle, J.-M. Roussel, and J.-J. Lesage, "Quantitative Analysis of Dynamic Fault Ttrees based on the Structure Function," *Qual. Rel. Eng. Int.*, vol. 30, no. 1, pp. 143–156, 2014.
[46] *MATLAB 2017a*, The MathWorks, Natick, MA, USA, 2017.
[47] Y. Elderhalli. (2019). *DFT Formal Analysis: HOL4 Script*. [Online]. Available: http://hvg.ece.concordia.ca/code/hol/DFT_method/index.php

YASSMEEN ELDERHALLI (S'19) received the B.Sc. degree (Hons.) in computer engineering from Al-Ahliyya Amman University, Amman, Jordan, in 2007, and the M.Sc. degree (with distinction) in electrical and computer engineering from the New York Institute of Technology. She is currently pursuing the Ph.D. degree with Concordia University, working on the formal dynamic reliability analysis. She worked as a full-time Lecturer with Al-Ahliyya Amman University, from 2009 to 2015.

OSMAN HASAN (M'07–SM'14) received the B.Eng. degree (Hons.) from the N-W.F.P University of Engineering and Technology, Pakistan, in 1997, and the M.Eng. and Ph.D. degrees from Concordia University, Montreal, QC, Canada, in 2001 and 2008, respectively. He worked as an ASIC Design Engineer with LSI Logic Corporation, Ottawa, ON, Canada, from 2001 to 2003. He is currently an Associate Professor with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Pakistan. He is the Founder and Director of SAVe Lab, NUST, which mainly focuses on the design and formal verification of embedded systems. He has received several awards and distinctions, including Pakistan's Higher Education Commission's Best University Teacher (2010), the Best Young Researcher Award (2011), and the President's gold medal for the Best Teacher of the University from NUST, in 2015. He is a member of the Association for Automated Reasoning (AAR) and a member of the Pakistan Engineering Council (PEC).

SOFIÈNE TAHAR (M'96–SM'07) received the Diploma degree in computer engineering from the University of Darmstadt, Darmstadt, Germany, in 1990, and the Ph.D. degree (Hons.) in computer science from the University of Karlsruhe, Karlsruhe, Germany, in 1994. He is currently a Professor and the Research Chair in formal verification of systems-on-chip with the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada, where he is also the Founder and the Director of the Hardware Verification Group. His research interests include formal hardware verification, system-on-chip verification, AMS circuit verification, and probabilistic, statistical, and reliability analysis of systems. He earned the title of the University Research Fellow upon receiving Concordia University's Senior Research Award, in 2007. He is a Senior Member of the ACM and a Professional Engineer in the Province of Quebec.