# Probabilistic Analysis of Dynamic Fault Trees using HOL Theorem Proving

Yassmeen Elderhalli, Waqar Ahmad, Osman Hasan, Sofiène Tahar
*Electrical and Computer Engineering*
*Concordia University, Montreal, QC, Canada*
{y_elderh, waqar, o_hasan, tahar}@ece.concordia.ca

**Abstract**

Dynamic Fault Trees (DFTs) is a widely used failure modeling technique that allows capturing the dynamic failure characteristics of systems in a very effective manner. Simulation and model checking have been traditionally used for the probabilistic analysis of DFTs. Simulation is usually based on sampling and thus its results are not guaranteed to be complete, whereas model checking employs computer arithmetic and numerical algorithms to compute the exact values of probabilities, which contain many round-off errors. Leveraging upon the expressive and sound nature of higher-order-logic (HOL) theorem proving, we propose, in this paper, a formalization of DFT gates and their probabilistic behaviors as well as some of their simplification properties in HOL based on the algebraic approach. This formalization would allow us to conduct the probabilistic analysis of DFTs by verifying generic mathematical expressions about their behavior in HOL. In particular, we formalize the AND, OR, Priority-AND, Functional DEPendency, Hot SPare, Cold SPare and the Warm SPare gates and also verify their corresponding probabilistic expressions in HOL. Moreover, we formally verify an important property, $Pr(X < Y)$, using the Lebesgue integral as this relationship allows us to reason about the probabilistic properties of the Priority-AND gate and the *Before* operator in HOL theorem proving. We also formalize the notion of conditional densities in order to formally verify the probabilistic expressions of the Cold SPare and the Warm SPare gates. In order to illustrate the usefulness of our formalization, we use it to formally analyze the DFT of a Cardiac Assist System.

## 1 Introduction

A Fault Tree (FT) [22] represents an effective way of graphically modeling the causes of failure in a system in the form of a rooted failure tree. A typical FT consists of

a *top event* representing system failure, basic failure events modeling the components failure and the FT gates, which combine the basic failure events and allow components failure to propagate to the top event. FTs are categorized as: Static FTs (SFTs) and Dynamic FTs (DFTs). SFTs capture the causes of failure in a system without considering the failure dependencies or sequences between the system components. DFTs, on the other hand, capture the failure dependencies in systems, which represent a more realistic approach to model the behavior of real-world systems.

Fault Tree Analysis (FTA) can be used to examine the failure characteristics of the given system qualitatively and quantitatively. In the former analysis, the combinations and sequences of basic failure events, associated with the system components, are determined in the form of cut sets and cut sequences [22]. While the quantitative analysis allows estimating the failure probability of the system based on component's failure probabilities among other metrics. Usually, Markov chain (MC) based analysis or algebraic approaches are used to perform DFT analysis. In the Markov chain based analysis, the DFT is first converted into its equivalent MC and then the analysis is conducted on the resulting MC. Complex systems often lead to a MC with a large number of states. The MCs of such complex systems can be analyzed using a modularization approach that divides the corresponding FT into SFT and DFT parts [19]. The SFT part is analyzed using traditional combinatorial analysis methods, such as Binary Decision Diagrams (BDDs) [22], while the DFT part is analyzed using MCs [23]. This kind of modularization approach has been implemented in the Galileo tool [24]. In the algebraic approach, an algebra similar to the ordinary Boolean algebra is used to reduce the structure function (expression) of the *top event* of the DFT [12]. This reduced expression is then used to derive the failure probability of the given system based on the failure probabilities of DFT gates.

Traditionally, DFTs are either analyzed by analytically deriving the system failure probability expression or using computer-based simulation tools. In the former method, firstly cut-sequences consisting of basic failure events are obtained and then the probabilistic Principle of Inclusion-Exclusion (PIE) [12] is used to manually derive the probability of failure of the overall system. This kind of manual manipulation is prone to human errors and can produce erroneous results especially when dealing with large DFTs. The latter method is more extensively used due to its scalability and user friendliness. Several simulation tools are available that provide GUI editors that obtain the system FT model from the user and return the analysis results based on the assigned failure distribution to the system components at a given instant of time. However, simulation cannot be guaranteed to produce complete and accurate results due to the involvement of numerical tech-

niques, such as Monte Carlo simulation [17], and pseudo random variables. Due to the above-mentioned inaccuracies, both analytical and simulation based methods are not suitable to conduct the failure analysis of safety-critical systems.

As an accurate alternative, formal methods have been recently utilized for analyzing FTs. Probabilistic model checkers (PMC), such as STORM [6], have been used to perform the quantitative analysis of DFTs [9]. However, due to the state-based nature of PMCs, they cannot be used to verify generic expressions for probability of failure. In addition, their usage is only limited to exponential distributions, which in the context of reliability analysis, for example, do not consider the aging of systems components. Due to the sound nature of higher-order-logic (HOL) theorem proving, it has been successfully used to formalize basic SFT gates [1], which have been in turn used to conduct the SFT-based analysis of several systems, including an air traffic management system [2]. However, this formalization is only limited to SFTs. So far, there is no formalization in HOL that supports the probabilistic failure analysis of DFTs. Recently, we have presented a hybrid methodology based on both interactive theorem proving and model checking for formal analysis of DFTs [8]. The main idea is to first conduct the qualitative analysis of a given DFT, based on the algebraic approach [12], using theorem proving and then quantitatively analyze the simplified DFT model using the STORM model checker. Since a PMC is involved in estimating the probabilities quantitatively, this methodology cannot provide generic expressions for probability and its usage is only limited to exponential distributions. Moreover, the formal definitions of DFT gates in [8] cannot cater for conducting the probabilistic analysis using HOL theorem proving as the behavior of the DFT gates has been captured using numbers instead of random variables.

In order to perform the complete probabilistic analysis of DFTs within a higher-order-logic theorem prover by verifying generic expressions of probability of failure, we propose to improve our formalization of the DFT gates in higher-order logic that uses the algebraic approach, presented in [12], as its foundation. The choice to formalize an algebraic approach to conduct the DFT analysis is motivated by the fact that HOL is well known for modeling systems that can be mathematically expressed. In addition, using HOL theorem proving we can formally verify generic expressions that cannot be obtained and verified using other formal tools and the algebraic approach fits perfectly with these features of HOL theorem proving. The foremost task in this work is to identify an algebraic approach to formalize DFTs so that the formal DFT analysis can be constructed within a theorem prover. In this respect, we have to consider the availability of foundational theories, like measure and probability, and their compatibility with the chosen approach. We identified the algebraic approach, initially proposed by Merle [12], to formalize DFTs among other options (e.g., [18]). Despite the fact that the presented formalization is based

on an existing algebraic approach [12], it bears its own research challenges. For example, it is well-known that a theorem proving based proof requires many intricate proof guidance and explicit reasoning that a mathematician doing a paper based proof would sometimes ignore. Thus, in our experience with the formalization of the algebraic approach, we also had to take many modeling decisions, choose appropriate data types, identify missing assumptions for the validity of results, devise proof strategies and verify many helper theorems to facilitate the process of formal DFT analysis in a theorem prover. The paper highlights these details while we present our formalization. Based on this novel formalization, we also formally verify the DFT algebraic reduction properties. Then, using the available probability theory formalization [13], we also formally verify the failure probability relationships of all commonly used DFT gates, i.e., AND, OR, Priority-AND (PAND), Functional DEPendency (FDEP), Hot SPare gate (HSP), Cold SPare gate (CSP) and Warm SPare gate (WSP). In order to verify the failure probability relationship of some of these DFT gates, we are required to formalize the $Pr(X < Y)$ describing the effect of one system component failing before the other or one after the other. This property is mainly verified by using Lebesgue integral properties [15, 21]. In addition, we formalize the notion of conditional density functions, which is necessary to formally verify the probabilistic relationships of the spare gates. The HOL4 theorem prover [10] was a natural choice for this formalization as it has the required theories such as: the probability theory and the Lebesgue integral [15]. In addition, we use the existing formalization of the probabilistic PIE in HOL4 [1]. The above-mentioned formalizations can be utilized to conduct the DFT-based failure analysis of a variety of real-world systems within the sound core of a theorem prover. For illustration purposes, we present the formal DFT-based failure analysis of a Cardiac Assist System (CAS) [5], which is a safety-critical DFT benchmark. We first reduce the original structure function of the system's top event using the formally verified simplification theorems. Then, we utilize the probabilistic PIE [1] to formally verify a generic failure probability expression of the Cardiac Assist System whereas the failure characteristics of its components are represented as generic probability distribution and density functions.

### 1.1 Contributions of the Paper

The main contributions of the paper are summarized as:

- Providing a framework for the probabilistic analysis of DFT within a theorem prover, which offers a sound and rigorous method for conducting DFT analysis by providing formally verified generic expressions of probability of failure.

- Development of reasoning steps for the verification of DFT gate properties, which, to the best of our knowledge, are not available in the literature or even in [12].
- Providing the formal definitions of DFT gates, which are somewhat different than the expressions provided in [12].
- Verifying a generic expression for the probabilistic failure behavior of a cardiac assist system in HOL theorem proving, which involves identifying the required conditions for the generic expression to hold.

### 1.2 Paper Organization

The rest of the paper is structured as follows: Section 2 presents some preliminaries about the probability theory and the Lebesgue integral in HOL4 that will facilitate the understanding of the rest of the paper. In Section 3, we present our HOL formalization of DFT gates and the corresponding simplification properties. Section 4 provides the verification details of the probabilistic behavior of the DFT gates. Section 5 presents the formalization of the probabilistic failure behavior of the Cardiac Assist System. Finally, we conclude the paper in Section 6.

## 2 Preliminaries

In this section, we present some preliminaries that are required for the understanding of the proposed formalization.

### 2.1 Probability Theory

The probability theory is formalized based on the measure theory in HOL4 [15]. A measurable space is represented as a pair $(\mathcal{X}, \mathcal{A})$, where $\mathcal{X}$ represents a space and $\mathcal{A}$ a set of measurable sets. The functions `space` and `subsets` are defined in HOL to return $\mathcal{X}$ and $\mathcal{A}$, respectively of a measurable space $(\mathcal{X}, \mathcal{A})$. A measure is generally a function that designates a certain number to a set, which represents the size of this set [13]. It is defined as the triplet $(\mathcal{X}, \mathcal{A}, \mu)$, where $\mathcal{X}$ represents the space, $\mathcal{A}$ represents the measurable sets and finally $\mu$ represents the measure. Three functions, `m_space, measurable_sets` and `measure`, are defined in HOL to return the space $(\mathcal{X})$, measurable sets $(\mathcal{A})$ and measure $(\mu)$ of a measure space, respectively [16]. A probability space is defined as a measure space, with the added condition that the probability measure for the entire space is equal to 1.

Random variables are formalized as measurable functions that map events from the probability space to some other $\sigma$- algebra space $s$. Random variables are defined in HOL4 as [13]:

**Definition 2.1.**

```
⊢ ∀X p s.  random_variable X p s ⇔
           prob_space p ∧ X ∈ measurable (p_space p, events p) s
```

where `prob_space p` ensures that `p` is a probability space with `p_space` as its space and `events` as its measurable sets. `X ∈ measurable (p_space p, events p) s` ensures that `X` belongs to the set of measurable functions from the probability space `p` to $\sigma$-algebra space `s` [16]. Measurable spaces `s` and `(p_space p, events p)` are ensured to be $\sigma$-algebra spaces using the `measurable` function.

The probability distribution of a random variable $X$ represents the probability that the random variable $X$ belongs to a set $A$. This is equivalent to finding the probability of the event $\{X \in A\}$, which can also be represented using the preimage as $X^{-1}(A)$. The probability distribution is defined in HOL4 as [13]:

**Definition 2.2.**

```
⊢ ∀p X. distribution p X = (λs.  prob p (PREIMAGE X s ∩ p_space p))
```

where `s` is a set of elements of the space that the random variable $X$ maps to. For a random variable that maps the probability space ($p$) into another measurable space, the push forward measure is a measure that uses the space and subsets of the measurable space as its space and measurable sets and uses the distribution of the random variable as its measure part [11]. In general, the push forward measure for any measurable function $X$ from measure $M$ to measure $N$ can be expressed as:

**Definition 2.3.**

```
⊢ ∀ M N f.  distr M N f =
    (m_space N, measurable_sets N,
         λA. measure M (PREIMAGE f A ∩ m_space M))
```

A density measure is used to define a density function, $f$, over the measure space $M$ as [11]:

**Definition 2.4.**

```
⊢ ∀ M f.  density M f =
    (m_space M, measurable_sets M,
         λA. pos_fn_integral M (λ x.  f x * indicator_fn A x ) )
```

where `pos_fn_integral` represents the Lebesgue integral of positive functions as will be described in the following section.

The cumulative distribution function (CDF) of a random variable $X$ is usually used when we are interested in finding the probability that the random variable is less than or equal to a certain value. It is formally defined for real values as [1]:

**Definition 2.5.**

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

It is worth mentioning that the CDF can be defined for extended-real (`extreal`) random variables as well, where `extreal` is a HOL data-type that includes the real numbers beside $\pm\infty$. However, in our formalization we will use the CDF of real random variables, as it is required to integrate their density functions over the real line.

When dealing with multiple random variables, the probabilistic *Principle of Inclusion and Exclusion* (PIE) provides a very interesting relationship between the probability of the union of different events. It can be expressed as:

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

It has been formally verified in HOL4 as follows [1]:

**Theorem 2.1.**

```
⊢ ∀p L.
    prob_space p ∧ (∀ x.  MEM x L ⇒ x ∈ events p) ⇒
    (prob p (union_list L = sum_set {t | t ⊆ set L ∧ t ≠ {}}
    (λt.  -1 pow (CARD t+1) * prob p (BIGINTER t))
```

where L is the list of events that we are interested in expressing the probability of their union.

In order to be able to handle multiple random variables, a pair measure (often called binary product measure) is required to be able to model joint distribution measures. This pair measure can be used also in a nested way to model the joint distribution measure of multiple random variables. The pair measure is defined as the product of two measures. It was initially formalized in Isabelle/HOL [11] and was then ported to HOL4 [20]. The space and the measurable sets of this pair measure are generated using the Cartesian product of the spaces and the measurable sets of the participating measures, while the measure part is defined using the Lebesgue integral.

Since there are real and extended-real data-types in HOL4, there exist two Borel spaces, one over the real line (`borel`) [21] and the second over the extended-real line (`Borel`) [14]. The Lebesgue-Borel measure is required to integrate over the real line. In particular, we need the Lebesgue-Borel measure in this work to integrate the density functions of the random variables over the real line. The Lebesgue-Borel measure is a measure defined over the real line, which uses the real line as its space and the Borel sets as its measurable sets. The Lebesgue-Borel measure is defined in HOL4 as `lborel`, which uses the real borel sigma algebra (`borel`) generated by the open sets of the real line as well as the Lebesgue measure [21].

The independence of random variables is an important property when dealing with multiple random variables. In general, for any two random variables $X$ and $Y$, the probability of the intersection of their events is equal to the multiplication of the probability of the individual events. The independence of random variables is defined as `indep_vars` [20]:

**Definition 2.6.**

```
⊢ indep_vars p M X ii =
   (∀i.  i ∈ ii ⇒
      random_variable (X i) p
        (m_space (M i), measurable_sets (M i))) ∧
   indep_sets p
      (λi.  {PREIMAGE f A ∩ p_space p |
         (f = X i) ∧ A ∈ measurable_sets (M i)}) ii
```

where p is the probability space and M is the measure space that the random variable X maps to. In this case, M and X are indexed by a number from the set of numbers ii, which gives the possibility of defining the independence for multiple random variables that map from the probability space to different spaces. The

function `indep_vars` defines the independence by first ensuring that the group of input functions `X` are random variables and that their event sets are independent using `indep_sets`. Using `indep_sets`, the probability of the intersection of any sub-group of events of the random variables is equal to the multiplication of the probability of the individual events.

Using `indep_vars`, the independence of two random variables is defined as [20]:

**Definition 2.7.**

```
⊢ indep_var p M_x X M_y Y =
  indep_vars p (λi.  if i = 0 then M_x else M_y)
               (λi.  if i = 0 then X else Y) {x | (x = 0) ∨ (x = 1)}
```

We define several functions that facilitate handling our formalization. The first function is `measurable_CDF`, which is defined as:

**Definition 2.8.**

```
⊢ ∀p X. measurable_CDF p X = (λx.  CDF p X x) ∈ measurable borel Borel
```

This function ensures that the CDF of random variable X is measurable from the `borel` space to the `Borel` space. In other words, it ensures that the CDF is measurable from the real line to the extended-real line. This implies that the domain for this CDF is the real line and the range is the extended-real line.

We define another function, `cont_CDF`, which ensures that the CDF is continuous. It is formally defined as:

**Definition 2.9.**

```
⊢ ∀p X. cont_CDF p X = ∀z.  (λx.  real (CDF p X x)) contl z
```

where the function `real` typecasts the value of CDF from extended-real to real data-type, and `contl` ascertains that the function is continuous over all values in its domain. It is worth mentioning that $X$ is a real valued random variable. However, the CDF returns extended-real. As the continuity of functions is defined in HOL4 for real valued functions, it is required to typecast the value of the CDF from extended-real to real. In addition, since the values of the CDF range from 0 to 1, as it represents a probability, this function is the same in both cases but with different datatypes. Therefore, if the function is continuous in the extended-real, then it is

continuous using the real datatype. Furthermore, later we will use extended-real random variables, therefore, it is required to typecast their values using the `real` function.

Next, we define a function, `rv_gt0_ninfinity`, to ensure that the input random variables of a DFT can only have the range $[0, +\infty)$:

**Definition 2.10.**

```
⊢ (rv_gt0_ninfinity [] = T) ∧
  (rv_gt0_ninfinity (h::t) = (∀s.  0 ≤ h s ∧ h s ≠ PosInf) ∧
     (rv_gt0_ninfinity t))
```

Finally, we define a function, `den_gt0_ninfinity` to ensure the proper values for the marginal, joint and conditional density functions:

**Definition 2.11.**

```
⊢ ∀f_xy f_y f_cond.
    den_gt0_ninfinity f_xy f_y f_cond ⇔
    ∀x y.
      0 ≤ f_xy (x,y) ∧ 0 < f_y y ∧ f_y y ≠ PosInf ∧ 0 ≤ f_cond y x
```

where `f_xy` is the joint density function, `f_y` is the marginal density function, and finally `f_cond` is the conditional density function of X given Y. This function can be used to assign the mentioned conditions to other functions and not necessarily only the density functions.

## 2.2 Lebesgue Integral

The Lebesgue integral is defined in HOL4 using positive simple functions, which are measurable functions defined as a linear combinations of indicator functions of measurable sets representing a partition of the space $X$ [15]. A positive simple function, $g$, can be represented using the triplet $(s,\ a,\ x)$ as [15]:

$$\forall t \in X,\ g(t) = \sum_{i \in s} x_i \mathbf{1}_{a_i}(t), \qquad x_i \geq 0 \tag{2}$$

where $s$ is a finite set of partition tags, $x_i$ is a sequence of positive `extreal` numbers, $a_i$ is a sequence of measurable sets and $\mathbf{1}_{a_i}$ is the indicator function of measurable set $a_i$ and is defined as [15]:

**Definition 2.12.**
⊢ ∀A. indicator_fn A = (λx. if x ∈ A then 1 else 0)

The Lebesgue integral is first defined for positive simple functions and then extended for positive functions for measure $\mu$ as [14]:

$$\int_X f d\mu = sup\{\int_X g\ d\mu \mid g\ \leq\ f\ and\ g\ positive\ simple\ function\} \tag{3}$$

It is usually required that the probability of an event for a random variable to be expressed using the integration of the random variable's distribution. This is verified in HOL4 as [13]:

**Theorem 2.2.**
```
⊢ ∀X p s A.
    random_variable X p s ∧ A ∈ subsets s ⇒
    (distribution p X A =
     integral (space s, subsets s, distribution p X)(indicator_fn A))
```

In the above theorem, $X$ can be a continuous or discrete random variable. However, in our DFT formalization, we are only interested in continuous random variables as they represent the time of failure of system components.

# 3 Formalization of Dynamic Fault Trees in HOL

Our previous formalization of DFT gates and operators was based on the algebraic approach [12], where the DFT events are treated based on their time of occurrence (failure of corresponding components) [8]. However, these formal definitions cannot cater for the probabilistic analysis of system failures, which is the scope of the current paper. Therefore, we provide an improved formalization of DFT gates and operators using functions of time that can be represented as random variables when carrying out the formal probabilistic analysis of the given DFT based on the algebraic approach presented in [12]. However, there are some missing gaps in the paper-and-pencil proofs available in [12] that we were able to fill using our formalization, particularly that we had to build our formalization on top of some existing HOL theories, such as the Lebesgue integral and probability theories. In [12], there is no direct description on how to build the DFT analysis based on the above-mentioned theories. Besides this, we also had to use different strategies for some proofs. All these differences will be highlighted throughout Sections 3 and 4.

### 3.1 Identity Elements and Temporal Operators

Similar to ordinary Boolean algebra, the DFT algebraic approach defines identity elements that are important in the simplification process of the DFT [12]. The DFT identity elements are: the *ALWAYS* element representing an event that always occurs (fails) from time 0 and the *NEVER* element, which describes an event that never occurs (fails). The formal definitions of these elements are shown in Table 1, where `PosInf` represents $+\infty$ in HOL4. We define the time of failure of the events as lambda abstracted functions that accept an arbitrary data-type that represents an element from the probability space and return the time. so that they can be later treated as random variables. For example, the time of failure of a component is a random variable $X$ and can be expressed in lambda abstraction form as `(λs. X s)`.

Temporal operators are also required to model the DFT gates in the algebraic approach [12]. These operators are: *Before* ($\lhd$), *Simultaneous* ($\Delta$) and *Inclusive Before* ($\trianglelefteq$). Each one of these operators accepts two inputs, which can be subtrees or basic events that represent faults of system components. The output event of the operator occurs according to a certain sequence of occurrence for the input events, i.e., the time of occurrence of the first (left) input is less than, equal to or less than or equal to the occurrence time of the second input (right) for the *Before*, the *Simultaneous* and the *Inclusive Before* operators, respectively. The time of occurrence of the output event of all operators is equal to the time of occurrence of the first input event (left). The mathematical expressions of these operators as well as their corresponding HOL formalizations are shown in Table 1, where `X` and `Y` represent the time of occurrence of events X and Y, respectively.

It is worth mentioning that if the inputs of the *Simultaneous* operator are basic events with continuous failure distributions, then the output of this operator can never fail [12]. This is because the time of failure is continuous, and the possibility that two system components failing at the same time can be neglected. As a consequence, it is assumed in the algebraic approach that any two *different* basic events

Table 1: Definitions of Identity Elements and Temporal Operators

| Element/Operator | Mathematical Expression | Formalization |
|---|---|---|
| `Always element` | $d(ALWAYS) = 0$ | `⊢ ALWAYS = (λs. (0:extreal))` |
| `Never element` | $d(NEVER) = +\infty$ | `⊢ 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. D_BEFORE 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. D_SIMULT X Y = (λs. if X s = Y s then X s else PosInf)` |
| `Inclusive Before` | $d(X \trianglelefteq Y) = \begin{cases} d(X), & d(X) \leq d(Y) \\ +\infty, & d(X) > d(Y) \end{cases}$ | `⊢ ∀ X Y. D_INCLUSIVE_BEFORE X Y = (λs. if X s ≤ Y s then X s else PosInf)` |

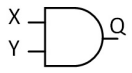
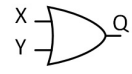
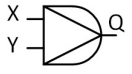
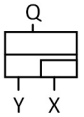
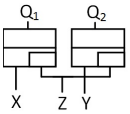
can never fail at the same time. This can be expressed for basic failure events of the inputs of the given DFT as [12]:

$$d(X \Delta Y) = NEVER \tag{4}$$

### 3.2 Formalization of FT Gates and Simplification Theorems

Our formalization of all FT gates; static and dynamic, and their mathematical expressions [12] are presented in Table 2.

Table 2: DFT Gates

| Gate | Mathematical Expression | Formalization |
|---|---|---|
|   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)) |
|   Spare | $d(Q_{CSP}) = \begin{cases} d(X), & d(Y) < d(X) \\ +\infty, & d(Y) \geq d(X) \end{cases}$ | ⊢ ∀X Y. CSP Y X = (λs. if Y s < X s then X s else PosInf) |
| | $d(Q_{HSP}) = max(d(Y), d(X))$ | ⊢ ∀X Y. HSP Y X = (λs. max (Y s)(X s)) |
| | $d(Q_{WSP}) = d(Y \cdot (X_d \lhd Y)+ X_a \cdot (Y \lhd X_a)+ Y \Delta X_a + Y \Delta X_d$ | ⊢ ∀Y X_a X_d. WSP Y X_a X_d = D_OR(D_OR(D_OR (D_AND Y (D_BEFORE X_d Y)) (D_AND X_a (D_BEFORE Y X_a))) (D_SIMULT Y X_a))(D_SIMULT Y X_d) |
|   Shared Spare | $d(Q_1) = d(X \cdot (Z_d \lhd X)+ Z_a \cdot (X \lhd Z_a)+ X \cdot (Y \lhd X))$ | ⊢ ∀X Y Z_a Z_d. shared_spare X Y Z_a Z_d = D_OR (D_OR (D_AND X (D_BEFORE Z_d X)) (D_AND Z_a (D_BEFORE X Z_a))) (D_AND X (D_BEFORE Y X))) |

### 3.2.1 AND and OR Gates

The AND ($\cdot$) and OR ($+$) gates can be modeled based on the time of occurrence of their output events. For the AND gate, the output occurs when both of its input events occur and the time of occurrence of the output is modeled as the maximum time of occurrence of both input events [12]. For the OR gate, the output occurs once one of its input events occurs. Therefore, we formalize it as the minimum time of occurrence of the inputs [12]. In Table 2, `max` and `min` are the HOL4 functions that represent the maximum and the minimum functions, respectively. It is important to notice that we define the AND and OR gates as lambda abstracted functions that accept two inputs that are also functions. This would enable defining the inputs later as random variables to represent the time of failure function of system components. This also applies to the formal definitions of the rest of DFT gates.

### 3.2.2 Priority AND Gate (PAND)

The PAND gate, shown in Table 2, captures the sequence of occurrence (failure) of its inputs. The output event of this gate occurs if all input events occur in a certain sequence (conventionally from left to right). In Table 2, we provide both the mathematical and formal definitions of the PAND gate. Then we verify that the behavior of the PAND can also be represented using the temporal operators as [12]:

$$Q = Y \cdot (X \trianglelefteq Y) \tag{5}$$

We verify the above relationship in HOL4 as follows:

**Theorem 3.1.** `⊢ ∀X Y. PAND X Y = D_AND Y (D_INCLUSIVE_BEFORE X Y)`

This result ascertains that the behavior of PAND gate is correctly captured in our formal definition. It is worth mentioning that in [12] the PAND gate is defined as Equation (5). However, we define it using a mathematical expression as in Table 2, which represents its actual behavior, and then verify that this definition is equal to the definition provided in [12] as in Theorem 3.1.

### 3.2.3 Functional DEPdency Gate (FDEP)

The FDEP is used to model the dependencies in the failure behavior between the system components. In other words, it is used when the failure of one component triggers the failure of another. For the FDEP gate, shown in Table 2, event $X$ can occur if it is triggered by the failure of $T$ or if it occurs by itself. As a result, the

occurrence time of $X_T$ (triggered $X$) equals the minimum time of occurrence of $T$ and $X$. From the FDEP definition, we can notice that its behavior is equivalent to the behavior of the OR gate.

### 3.2.4 Spare Gates

Modeling spare parts in real systems is necessary when analyzing the probability of failure of the overall system, as these spares are used to replace the main parts after their failure. The main part $Y$ of the spare gate, shown in Table 2, is replaced by the spare part $X$ after a failure occurs. The spare gate has three variants depending on the type of the spare:

- ***Cold SPare Gate (CSP)***: The spare part can only fail while it is active.
- ***Hot SPare Gate (HSP)***: The spare part can fail in both the active and the dormant states with the same probability.
- ***Warm SPare Gate (WSP)***: The spare part can fail in both the dormant and active states with different probabilities.

While manipulating the structure function of the DFT, it is required to distinguish between the two states of the spare part, i.e., the active state and the dormant state, therefore a different variable is assigned to each state. For example, for the spare gate in Table 2, variable $X$ is assigned $X_d$ and $X_a$ for the dormant and active states, respectively [12]. This is required in case of a WSP gate, where the spare part has two different states. Recall that in the case of a $CSP$ gate, it is not necessary to use these subscripts, since the spare part in the CSP gate does not work in the dormant state. Therefore, the active state only affects the DFT behavior and is included in the expressions. In the HSP gate, the spare part has the same behavior for both states and no subscript is required to distinguish between these two.

It can be noticed from the definition of the $WSP$ gate that the output of the spare occurs in two cases; if the spare fails in its dormant state, then the main part fails or the main part fails then the spare is activated and then it fails in its active state. The last two terms in the WSP definition cover the possibility that the spare and the main part fail at the same time. This can happen if the main part and the spare are functionally dependent on the same trigger. The $WSP$ represents the general case for the spare gates, while the $CSP$ and $HSP$ represent special cases of the $WSP$, where the spare cannot fail or is fully functioning in its dormant state. We have defined mathematical expressions for both the $CSP$ gate for basic events and the $HSP$ gate to facilitate using their expressions in DFT analysis. However, as will be seen shortly, we have verified that the behavior of our expressions is equivalent to

a $WSP$ under certain conditions. For the $CSP$ gate, the output occurs if the main part fails then the spare is activated and then the spare fails while it is active. Since the spare part of the $HSP$ has the same failure distribution in both of its states, the output of the $HSP$ occurs when both inputs (main and spare) fail. Therefore, its behavior is equivalent to an AND gate. We formally verify that the WSP gate is equivalent to an HSP gate when the spare part in its dormant state is equal to its active state.

**Theorem 3.2.** ⊢ ∀X Y. WSP Y X X = HSP Y X

Moreover, we formally verify that the WSP gate is equivalent to a CSP gate, if the spare part cannot fail in its dormant state. We formally verify this as:

**Theorem 3.3.** ⊢ ∀X_a X_d Y. (X_d = NEVER) ∧
(∀s. ALL_DISTINCT [Y s; X_a s]) ⇒ WSP Y X_a X_d = CSP Y X_a

where `X_d = NEVER` indicates that the spare part cannot fail in its dormant state, and `ALL_DISTINCT` ensures that the inputs cannot fail at the same time. This is because we defined the $CSP$ gate for basic events. As can be seen from the above theorem, the $CSP$ gate only deals with the active state of the spare, therefore, when dealing with a CSP there is no need to use the subscript.

In some real-world applications, a spare part can replace one of two main parts. This case is represented using shared spare gates as shown in Table 2 [8]. The expression of the output $Q_1$ of the first gate is listed in Table 2 [12]. This expression implies that the output $Q_1$ of this gate occurs in three different situations: *(i)* if the main part $X$ fails, then the spare fails while it is active ($Z_a$), *(ii)* if the spare part fails in its dormant state $Z_d$, then the main part fails, or *(iii)* if the second main part (of the other gate) $Y$ fails before $X$, and thus the spare is not available to replace $X$ when it fails. We use the DFT operators to model the behavior of this gate, as shown in Table 2.

In the DFT algebraic approach, many simplification theorems exist and are used to reduce the structure function of the top event [12]. In [8], we verified over 80 simplification theorems. However, these theorems were based on our old definitions of the DFT gates and operators that cannot cater for probabilistic analysis. We verify all these theorems for the new definitions, presented in this paper, and the details can be accessed from [7]. These simplification theorems range from simple ones, such as commutativity of the AND, OR and Simultaneous operator, to more complex ones that include combinations of all the operators. Table 3 includes some of these verified properties.

Table 3: Examples of Formally Verified Simplification Theorems

| DFT Algebra Theorems | HOL Theorems |
|---|---|
| $X{+}Y{=}Y{+}X$ | ⊢ ∀X Y. D_OR X Y = D_OR Y X |
| $X.NEVER{=}NEVER$ | ⊢ ∀X. D_AND X NEVER = NEVER |
| $X{\lhd}(Y{+}Z){=}(X{\lhd}Y).(X{\lhd}Z)$ | ⊢ ∀ X Y Z. D_BEFORE X (D_OR Y Z) = D_AND (D_BEFORE X Y)(D_BEFORE X Z) |
| $X{\trianglelefteq}(Y{+}Z){=}(X{\trianglelefteq}Y).(X{\trianglelefteq}Z)$ | ⊢ ∀ X Y Z. D_INCLUSIVE_BEFORE X (D_OR Y Z) = D_AND (D_INCLUSIVE_BEFORE X Y) (D_INCLUSIVE_BEFORE X Z) |
| $(X{\trianglelefteq}Y){+}(X\Delta Y){=}X{\trianglelefteq}Y$ | ⊢ ∀X Y. D_OR (D_INCLUSIVE_BEFORE X Y) (D_SIMULT X Y) = D_INCLUSIVE_BEFORE X Y |

# 4 Formal Verification of DFT Probabilistic Behavior

In order to formally verify the probability of failure of the top event of a DFT, it is required to formally model and verify the probability of failure expression for each DFT gate. We assume that the basic events of the DFT are independent. However, in some cases these events can be dependent; in particular in the case of CSP and WSP, where the failure of the main part affects the operation and failure of the spare part. We handle this by first introducing the probabilistic behavior of the gates for independent events, then we present the probabilistic behavior of the $WSP$ and the $CSP$ gates, which deal with dependent events. At the end of this section, we present a summary of the challenges that we faced during the formalization of the probabilistic failure behavior of DFT gates.

## 4.1 Probabilistic Behavior of Gates with Independent Events

Assuming that we are interested in finding the probability of failure until time t, the following four expressions can be used to express the probability of any DFT gate with independent basic events [12]:

$$Pr\{X \cdot Y\}(t) = F_X(t) \times F_Y(t) \tag{6a}$$

$$Pr\{X + Y\}(t) = F_X(t) + F_Y(t) - F_X(t) \times F_Y(t) \tag{6b}$$

$$Pr\{Y \cdot (X \lhd Y)\}(t) = \int_0^t f_Y(y)\ F_X(y)\ dy \tag{6c}$$

$$Pr\{X \lhd Y\}(t) = \int_0^t f_X(x)(1 - F_Y(x))\ dx \tag{6d}$$

where $F_X$ and $F_Y$ represent the CDFs of the random variables $X$ and $Y$, respectively, and $f_X$ and $f_Y$ represent their corresponding PDFs.

Equation (6a) represents the probability of the AND and HSP gates, which results from the probability of intersection of two independent events. Equation (6b) describes the probability of the OR and FDEP gates, which corresponds to the probability of union of two independent events. Equation (6c) represents the probability of having two basic events occurring in sequence one *after* the other until time $t$, i.e., $Pr(X < Y)$ until time $t$ or $Pr(X < Y \wedge Y \leq t)$, which is the failure probability of the PAND for basic events. Finally, the probability of the *Before* operator is represented by Equation (6d), which is the probability of having event $X$ occurring *before* event $Y$ until time $t$, i.e., $Pr(X < Y \wedge X \leq t)$. The difference between the last two events (*before* and *after*) is that in the *before* event, we are just interested in finding the probability of failure of $X$ until time $t$ with the condition that $X$ fails before $Y$. So, it is not necessary that $Y$ fails. While in the *after* event, we find the probability of failure of $Y$ until time $t$ with the condition that $Y$ fails after $X$. So, it is required that both $X$ and $Y$ fail in sequence.

Since the probability is applied for sets that belong to the events of the probability space, we define a `DFT_event` that satisfies the condition that the input function is less than or equal to time $t$, which represents the moment of time until which we are interested in finding the probability of failure. Without this `DFT_event`, there is no possible way to apply the probability directly to DFT gates. We first need to create the `DFT_event` for the time-to-failure function of the output event of any gate or DFT, then apply the probability to it.

**Definition 4.1.**

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

where `Normal` typecasts the type of $t$ from `real` to `extreal`, $p$ represents the probability space and $X$ represents the time-to-failure function.

We formally verify the equivalence between the probability of the `DFT_event` of an extended real function and its equivalent CDF of the real version of the function as:

**Theorem 4.1.**
```
⊢ ∀X p t.  (∀s.  X s ≠ PosInf ∧ 0 ≤ X s) ⇒
    (CDF p (λs.  real (X s)) t = prob p (DFT_event p X t))
```

where `real` is mirror opposite to the typecasting `Normal` operator. This typecasting is required as the `DFT_event` is defined for `extreal` data-type, and the CDF is defined for real random variables only. Therefore, it is required to ensure that the input function does not equal $+\infty$ and is greater than or equal to 0 since it represents the time of failure of a system component.

### 4.1.1 Probabilistic Behavior of AND, HSP, OR and FDEP Gates

To formally verify Equations (6a) and (6b), we verify the equivalence of the DFT event of the AND gate to the intersection of two events and the OR as the union:

**Lemma 4.1.**
```
⊢ ∀p t X Y.
    DFT_event p (D_AND X Y) t = DFT_event p X t ∩ DFT_event p Y t
```

**Lemma 4.2.**
```
⊢ ∀p t X Y.
    DFT_event p (D_OR X Y) t = DFT_event p X t ∪ DFT_event p Y t
```

Based on the independence of random variables and using Theorem 4.1, we formally verify Equation (6a) in HOL4 as:

**Theorem 4.2.**
```
⊢ ∀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 (D_AND X Y) t) =
     CDF p (λs.  real (X s)) t * CDF p (λs.  real (Y s)) t
```

where `indep_var` ensures the independence of the random variables, $X$ and $Y$, over the Lebesgue-Borel (`lborel`) measure [20]. `rv_gt0_ninfinity` is required since we

are dealing with the real versions of the random variables. It is a logical condition, since any real-world component will eventually fail, so we are interested only in dealing with the time of failure that is not $\infty$.

In Theorem 4.2, the random variables are type-casted as real-valued, using the operator `real`, to function over the Lebesgue-Borel (`lborel`) measure. `lborel` is purposely used here to facilitate the Lebesgue integration over the real line when expressing the probabilities of the *before* and *after* events. Theorem 4.2 represents the probability of the AND gate and the $HSP$ gate, since the behavior of the $HSP$ is equivalent to the behavior of the AND gate.

We formally verify Equation (6b) based on the probabilistic PIE and the independence of random variables and using Theorem 4.1 as:

**Theorem 4.3.**

```
⊢ ∀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 (D_OR X Y) t) =
     CDF p (λs.  real (X s)) t + CDF p (λs.  real (X s)) t -
     CDF p (λs.  real (X s)) t × CDF p (λs.  real (Y s)) t)
```

where `All_distinct_events` ascertains that the event sets are not equal. We formally define it as:

**Definition 4.2.**

```
⊢ All_distinct_events p L t =
  ALL_DISTINCT (MAP (λx.  DFT_event p x t) L
```

where `ALL_DISTINCT` is a HOL4 predicate, which ensures that the elements of its input list are not equal, `MAP` is a function that applies the input function `(λx. DFT_event p x t)` to all the elements in the list `L` and returns a list. This condition is required for the probabilistic PIE.

Theorem 4.3 provides the probability of the OR gate as well as the FDEP gate, since the behavior of the FDEP is equivalent to the OR gate.

It is worth noting that in [12], Equations (6a) and (6b) were just presented without any information on how to link them to the definitions of the AND and OR gates. We should recall that the AND and OR gates are defined as the maximum and minimum of their operands. Looking at these definitions does not give any knowledge about how the probability of the AND gate is equivalent to the probability of the

intersection or how the probability of the OR gate is equal to the probability of the union. However, using our formalization and utilizing our formal definition of `DFT_event`, we are able to verify that the `DFT_event` of the AND gate is equal to the intersection of the input events and that the `DFT_event` of the OR gate is equal to the union of the input events. Based on this, we can ensure that the probability of the AND and OR gates are represented using Equations (6a) and (6b), respectively.

### 4.1.2 Probabilistic Behavior of PAND Gate and Before Operator

We verify Equations (6c) and (6d) as Theorems 4.4 and 4.5, respectively.

**Theorem 4.4.**

```
⊢ ∀X Y p fy 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)) fy ∧ (∀y.  0 ≤ fy y) ∧
    cont_CDF p (λs.  real (X s)) ∧
    measurable_CDF p (λs.  real (X s)) ⇒
    (prob p (DFT_event p (Y·(X◁Y)) t) =
     pos_fn_integral lborel
        (λy.  fy y *
             (indicator_fn {w | 0 ≤ w ∧ w ≤ t} y *
              CDF p (λs.  real (X s)) y)))
```

**Theorem 4.5.**

```
⊢ ∀X Y p fy 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)) fx ∧ (∀x.  0 ≤ fx x) ∧
    measurable_CDF p (λ s.  real (Y s)) ⇒
    (prob p (DFT_event p (X ◁ Y) t) =
     pos_fn_integral lborel
        (λx.  fx x *
              (indicator_fn {u | 0 ≤ u ∧ u ≤ t} x *
               (1- CDF p (λs real (Y s)) x)))
```

where `pos_fn_integral` is the Lebesgue integral for positive functions [15], `fy` and `fx` are the PDF of random variables of the real version of functions $Y$ and $X$, respectively. `cont_CDF` is required in Theorem 4.4 as we need to prove that $Pr(X \leq t)$ and

$Pr(X < t)$ are equal, and this is not valid unless the CDF function is continuous (`cont`).

Verifying Theorems 4.4 and 4.5 is not a straightforward task due to the involvement of Lebesgue integration. To the best of our knowledge, this is the first time that these proofs are formally verified in a theorem prover, where we are able to identify the exact steps to reach the final form of Theorems 4.4 and 4.5. In addition, in [12], Equation (6c) is presented without any proof, while a proof is presented for Equation (6d) that is based mainly on the probability of disjoint events and utilizes derivatives to reach the final expression. However, we have been able to verify the same expression of Equation (6d), but following a different and simpler proof, which is similar to the proof of Equation (6c) to reach the final form of Theorem 4.5 without using derivatives. We first prove the probability of sets of real random variables in the form of integration before extending the proofs to extended real functions.

### Proof Strategy for Theorem 4.4

To verify Theorem 4.4, we first express the event set that corresponds to the integration in Equation (6c) as:

$$(X,Y)^{-1}\{(u,w) \mid u \ < \ w \ \wedge \ 0 \leq \ w \wedge \ w \ \leq \ t\} \tag{7}$$

Then we verify that the probability of this set can be written using the integration as in Equation (6c). Therefore, we verify the relationship between the distribution and the integration of positive functions using the push forward measure (`distr`):

**Theorem 4.6.**

```
⊢ ∀X p M A.
    measure_space M ∧
    random_variable X p (m_space M, measurable_sets M) ∧
    A ∈ measurable_sets M ⇒
    (distribution p X A =
     pos_fn_integral (distr p M X) (indicator_fn A))
```

It is worth mentioning that this theorem can be used in the verification process of other applications and not only for DFT analysis. We use Theorem 4.6 to verify the relationship between the probability and the integration of the joint distribution $F_{XY}$ of two independent random variables as:

$$Pr(X,Y)^{-1}(A) = \int \mathbf{1}_A \, dF_{XY} \tag{8}$$

We formalize this relationship in HOL4 and use a property, which converts the distribution of a pair measure of independent measures into the pair measure of the individual distributions [20], to split the integral of joint distributions into two integrals of the individual distributions ($\int\int \mathbf{1}_A dF_X dF_Y$). In order to reach the final form of Equation (6c), we express it in the form of two integrals:

$$\int_0^t f_Y(y) \times F_X(y)\ dy = \int_0^t \int_{-\infty}^{y} f_Y(y) \times f_X(x)\ dx\ dy \tag{9a}$$

$$= \int_0^t f_Y(y)\ \Big( \int_{-\infty}^{y} f_X(x)\ dx\ \Big) dy \tag{9b}$$

The problem in Equations (9a) and (9b) lies in the fact that the outer integral is a function of the inner integral, i.e., for the inner integral we are integrating until $y$ which is the variable of the outer integral. To be able to handle this formally, we verify that the indicator function of the set in Equation (7) can be written in the form of the multiplication of two indicator functions, where one is a function of the other.

**Lemma 4.3.**

```
⊢ ∀x y t.
    indicator_fn {(u,w) | u < w ∧ 0 ≤ w ∧ w ≤ t}(x,y) =
    indicator_fn {w| 0 ≤ w ∧ w ≤ t} y * indicator_fn {u|u < y} x
```

In order to use the above-mentioned lemma and the set on the left hand side, we need to verify that this set is measurable in the two dimensional borel space, i.e., the set belongs to the measurable sets of `pair_measure lborel lborel`. This property can be verified based on the fact that the countable union of measurable sets is also measurable. We verify this fact on the rational numbers $\mathbb{Q}$ as follows:

**Theorem 4.7.**

```
⊢ ∀m s.
    measure_space m ∧ (∀n.  n ∈ Q_set ⇒ s n ∈ measurable_sets m) ⇒
    BIGUNION (IMAGE s Q_set) ∈ measurable_sets m
```

where $m$ in our case is `pair_measure lborel lborel`. This theorem is generic and can be used in other contexts than DFTs.

The purpose of using the set of rational numbers is that we need a countable set that can be used to express the set in Lemma 4.3 as the union of borel rectangles. We verify this in HOL4 as:

**Lemma 4.4.**

```
⊢ ∀t.  BIGUNION
          {{u | u < real q} × {w | real q < w ∧ 0 ≤ w ∧ w ≤ t} |
            q ∈ Q_set} =
       {(u,w) | u < w ∧ 0 ≤ w ∧ w ≤ t}
```

Besides this, we also verify a lemma that the sets in the union of Lemma 4.4 are measurable sets in the `pair_measure lborel lborel` as:

**Lemma 4.5.**

```
⊢ ∀t q.  {u | u < real q} × {w | real q < w ∧ 0 ≤ w ∧ w ≤ t} ∈
         measurable_sets (pair_measure lborel lborel)
```

We can use the proof steps of the previous lemmas to verify the same properties for similar sets, which is essential for other gates expressions. This facilitates dealing with other events in the future, by following the steps in our proof.

By using the above lemmas, we can reason that the original set is a measurable set in the `pair_measure lborel lborel` as:

**Lemma 4.6.**

```
⊢ ∀t.  {(u,w) | u < w ∧ 0 ≤ w ∧ w ≤ t} ∈
       measurable_sets (pair_measure lborel lborel)
```

We use Lemmas 4.3 and 4.6 to verify that the expression given in Equation (9b) is equal to $\int_A dF_X dF_Y$, where $A$ is the set that specifies the boundaries of the integral. We verify this in HOL4 using the push forward measure as:

**Lemma 4.7.**

```
⊢ ∀X Y p t.
    prob_space p ∧ indep_var p lborel X lborel Y ⇒
    (pos_fn_integral (pair_measure (distr p lborel X)
             (distr p lborel Y))
       (λ(x,y).  indicator_fn{(u,w) |u < w ∧ 0 ≤ w ∧ w ≤ t }(x,y) =
     pos_fn_integral (distr p lborel Y)
        (λy.  indicator_fn {w|0 ≤ w ∧ w ≤ t} y *
           pos_fn_integral(distr p lborel X)
              (λx.  indicator_fn {u | u < y} x)))
```

where `pair_measure (distr p lborel X) (distr p lborel Y)` represents the joint distribution of the push forward measures of random variables $X$ and $Y$ over the borel space.

We verify several essential properties for CDF in order to prove that the inner integral of Lemma 4.7 is equal to $F_X(y)$ or formally to `(CDF p X y)`. In order to have the PDF of random variable $Y$ in the integral, we assume that the random variable $Y$ has a PDF by defining a density measure for $Y$. We ported the following definition, `distributed`, from Isabelle/HOL [11], where $f$ in this definition is the PDF of random variable $X$, and the measure part of the density measure is the integral of this PDF. Using this definition, the integral of $f$ is equal to the distribution of the random variable $X$.

**Definition 4.3.**

```
⊢ ∀p M X f.
    distributed p M X f ⇔
    X ∈
    measurable(m_space p,measurable_sets p)
      (m_space M,measurable_sets M) ∧
    f ∈ measurable(m_space M,measurable_sets M) Borel ∧
    AE M {x | 0 ≤ f x} ∧ (distr p M X = density M f)
```

where `density` is the density measure, and `AE M {x | 0 ≤ f x }` ensures that the PDF `f` is almost everywhere positive over the measure `M`. We also use a theorem that replaces the integration with respect to the density measure by the PDF with respect to the original measure (`lborel` in our case) [11]. In addition to the previously verified theorems, we also prove some additional properties, such as sigma finite measure for the push forward measure over the borel space (`sigma_finite_measure (distr p lborel X)`). We also verify that the space generated by the pair measure of two distributions over the borel space is sigma algebra (`sigma_algebra (m_space (pair_measure (distr p lborel X)(distr p lborel Y)), measurable_sets (pair_measure (distr p lborel X)(distr p lborel Y)))`). Moreover, we verify that the space generated by the space and the measurable sets of the pair measure of `lborel` is also a sigma algebra (`sigma_algebra (m_space (pair_measure lborel lborel), measurable_sets (pair_measure lborel lborel))`). Finally, we prove that the set of the left-hand side of Equation (6c) is equal to the set that corresponds to the integration of the right-hand side of the same equation as:

**Lemma 4.8.**
```
⊢ ∀p t X Y.
    rv_gt0_ninfinity [X; Y] ∧ 0 ≤ t ⇒
    (DFT_event p (Y·(X◁Y)) t =
     PREIMAGE (λx.  (real (X x), real (Y x)))
        {(u,w) | u < w ∧ 0 ≤ w ∧ w ≤ t} ∩ p_space p
```

Based on all the above mentioned lemmas, we are able to verify the original goal for Equation (6c) as in Theorem 4.4.

### Proof Strategy for Theorem 4.5

For the verification of Theorem 4.5, we follow almost the same steps for the previous proof. We start by first writing the event set for the integration as:

$$(X,Y)^{-1}\{(u,w) \mid 0 \leq u \wedge u \leq t \wedge u < w\} \tag{10}$$

Then, we describe the indicator function of this set as the multiplication of two indicator functions as:

**Lemma 4.9.**
```
⊢ ∀x y t.
    indicator_fn {(u,w) | 0 ≤ u ∧ u ≤ t ∧ u < w}(x,y) =
    indicator_fn {u | 0 ≤ u ∧ u ≤ t} x * indicator_fn {w | x < w} y
```

Similar to the procedure, explained previously for the set of the after event in Lemmas 4.4, 4.5 and 4.6, we verify that the set of the before event is a measurable set in the `pair_measure lborel lborel`.

Finally, we rewrite Equation (6d) as:

$$\begin{aligned} Pr\{X \lhd Y\}(t) &= \int_0^t \int_x^\infty f_X(x)\ f_Y(y)\ dy\ dx \\ &= \int_0^t f_X(x) \left( \int_x^\infty f_Y(y)\ dy \right) dx \end{aligned} \tag{11}$$

We verify some additional properties for the CDF in order to complete the proof. For example, we verify that $\int_x^\infty f_Y(y)\ dy$ is equal to $1 - F_Y(x)$. Similarly, we also formally verify that the event of the left-hand side of Equation (6d) is equal to the set that corresponds to the integration of the right-hand side of the same equation. We use the set in Equation (10) to verify this as:

**Lemma 4.10.**

```
⊢ ∀p t X Y.
    rv_gt0_ninfinity [X; Y] ∧ 0 ≤ t ⇒
    (DFT_event p (X◁Y) t =
     PREIMAGE (λs.  (real (X s),real (Y s)))
        {(u,w) | 0 ≤ u ∧ u < w ∧ u ≤ t} ∩ p_space p
```

Based on all these verified theorems, we are able to formally verify Theorem 4.5.

So far, we presented the formal verification of the probabilistic behavior of:

- The AND and HSP gates using Theorem 4.2 (since they are equivalent).
- The probability of the OR and FDEP gates using Theorem 4.3 (since they are equivalent).
- The probability of the PAND gate for basic events using Theorem 4.4.
- The probability of the *Before* operator using Theorem 4.5.

There is no probability of failure for the *Simultaneous* operator as it is eliminated for basic events according to Equation (4). This implies that the probability of the *Inclusive Before* operator is equal to the probability of the *Before* operator for basic events.

## 4.2 Probabilistic Behavior of Gates with Dependent Events

The probabilistic behavior of the CSP and WSP requires dealing with dependent events, as the failure of the main part affects the behavior of the spare part. Therefore, it is required to approach the proof in a different manner.

For the $CSP$, the failure distribution of the spare part is affected by the failure time of the main part, as the cold spare starts working after the failure of the main part. Hence, the failure distribution of the spare part is dependent on the failure of the main part. The probability of failure for the output event of a CSP with $Y$ as the main part and $X$ as the spare part is given by [12]:

$$Pr(Q_{CSP})(t) = \int_0^t \Big( \int_v^t f_{(X_a|Y=v)}(u)du \Big) f_Y(v)dv \tag{12}$$

where $f_{(X_a|Y=v)}$ is the conditional probability density function for the spare part in its active state ($X_a$) given that the main part($Y$) has failed at time $v$. As mentioned previously, the subscript of $X_a$ can be omitted, since the spare part of the CSP

gate does not work in its dormant state and we are only concerned with the active state, so using $X$ directly with CSP means that we are dealing with the active state and not the dormant one. It can be noticed from Equation (12) that the failure distribution of the spare part is affected by the failure of the main part. Hence, these two input events are not independent, and we cannot utilize the previously verified relationships in Section 4.1 to verify the probabilistic behavior of the CSP gate.

For the WSP gate with two basic events, the output fails in two cases, Case 1: when the main part fails, then the spare fails in its active state (this case is similar to the CSP case); Case 2: when the spare part fails in its dormant state, then the main part fails with no spare to replace it. In the latter case, the failure distribution of the spare part in its dormant state is independent of the main part. Hence, we can use the previously verified expressions for this case. The probability expression for a WSP with $X$ as the spare part ($X_a$ for the active state and $X_d$ for the dormant state) and $Y$ as the main part is expressed as [12]:

$$Pr(Q_{WSP})(t) = \int_0^t \Big( \int_v^t f_{(X_a|Y=v)}(u)du \Big) f_Y(v)dv + \int_0^t f_Y(u) F_{X_d}(u)du \qquad (13)$$

where $F_{X_d}$ is the CDF of $X$ in its dormant state. The first part of Equation (13) represents the probability of a CSP and the second part represents the probability when the spare fails before the main part. For the second part, $Y$ and $X_d$ are considered to be independent as the failure of one of them does not affect the failure of the second and hence we can use Equation (6c) for this case.

We verify Equations (12) and (13) as Theorems 4.8 and 4.9, respectively.

**Theorem 4.8.**

```
⊢ ∀p X Y f_xy f_y f_cond 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_cond) ∧
    prob_space p ∧ den_gt0_ninfinity f_xy f_y f_cond ⇒
    (prob p (DFT_event p (CSP Y X) t) =
     pos_fn_integral lborel
        (λy.
           indicator_fn {u | 0 ≤ u ∧ u ≤ t} y * f_y y *
           pos_fn_integral lborel
              (λx.  indicator_fn {w | y < w ∧ w ≤ t} x * f_cond y x )))
```

**Theorem 4.9.**

```
⊢ ∀p Y X_a X_d t f_y f_xy f_cond.
    prob_space p ∧ (∀s.  ALL_DISTINCT [X_a s; X_d s; Y s]) ∧
    (D_AND X_a X_d = NEVER) ∧ 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_cond) ∧
    den_gt0_ninfinity f_xy f_y f_cond ∧
    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) =
     pos_fn_integral lborel
        (λy.
           indicator_fn {u | 0 ≤ u ∧ u ≤ t} y * f_y y *
           pos_fn_integral lborel
              (λx.  indicator_fn {w | y < w ∧ w ≤ t} x * f_cond y x ))+
     pos_fn_integral lborel
        (λy.
           f_y y *
           (indicator_fn {u | 0 ≤ u ∧ u ≤ t} y *
            CDF p (λs.  real (X_d s)) y )))
```

where $p$ is the probability space, `f_xy` is the joint density function for $X$ and $Y$, `f_y` is the marginal density function for $Y$, `cond_density` defines the conditional density function (`f_cond`) for $X$ given that $(Y = y)$ and `den_gt0_ninfinity` ensures the proper values for the density functions as mentioned in Section 2.

It is noticed that the spare part in the CSP is used without any subscript, i.e., it is used as $X$, since the spare has only one state in the CSP, which is the active state. Therefore, there is no need to use any subscript to distinguish between the dormant and the active states. While in the WSP, we need to distinguish between the two states, i.e., active and dormant, hence the usage of $X_a$ and $X_d$. For Theorem 4.9, the condition `D_AND X_a X_d = NEVER` ensures that the spare part can only fail in one of its states but not both. This condition is different from `D_SIMULT X_a X_d = NEVER`, as the former means that if one of the inputs occurs, then the other cannot occur at all. While the latter means that both inputs cannot occur at the same time, they can occur at different times. This second condition is ensured in our case using `ALL_DISTINCT`. In addition, it is assumed that the spare part in the dormant $(X_d)$ state is independent of the main part $Y$ since the failure of the spare part in its dormant state is not affected by the failure of the main part. As with the previous

theorems in Section 4.1, we need to use the typecast operator `real` with the random variables, since the random variables are of type `extreal` and the integral over the `lborel` requires real random variables.

In [12], a proof has been introduced for the above expressions, which is based mainly on the total expectation theorem [4]. However, we have been able to conduct the same proof in a simpler manner based on conditional density functions as explained below.

### Proof Strategy for Theorem 4.8 (CSP Gate)

In order to verify Theorem 4.8, we formalize the conditional density function as [3]:

**Definition 4.4.**

```
⊢ ∀M1 M2 p X Y y f_xy f_y f_cond.
    cond_density M1 M2 p X Y y f_xy f_y f_cond ⇔
    random_variable X p (m_space M1, measurable_sets M1) ∧
    random_variable Y p (m_space M2, measurable_sets M2) ∧
    distributed p (pair_measure M1 M2) (λx.  (X x, Y x)) f_xy ∧
    distributed p M2 Y f_y ∧ (f_cond y = (λx.  f(x,y) / f_y y))
```

where `p` is the probability space, `M1` and `M2` are the measure spaces that the random variables $X$ and $Y$ map to, respectively (we will use `lborel` in our case), `f_xy` is the joint density function for $X$ and $Y$, `f_y` is the marginal density function of $Y$ and finally, `f_cond` is the conditional density function of $X$ given $(Y = y)$.

The conditional density function definition ensures that $X$ and $Y$ are random variables with joint density function `f_xy` and a marginal density function `f_y`. It is noticed from the definition of the conditional density function `f_cond` that it is a function of $x$ only, and it can have different variants depending on the value of $Y$ that we are conditioning at, i.e., $y$. This is why `f_cond` takes $y$ as a parameter.

From Definition 4.4, we formally verify the following relationship between the conditional density, the joint density and the marginal density functions, given that $f_Y(y) \neq 0$:

$$f_{XY}(x, y) = f_{X|Y=y}(x) \times f_Y(y) \tag{14}$$

The above equation can be formalised in HOL4 as:

**Theorem 4.10.**

```
⊢ ∀M1 M2 p X Y f_cond x y f_xy f_y.
    (∀y.  f_y y ≠ 0 ∧ f_y y ≠ PosInf ∧ f_y y ≠ NegInf) ∧
    cond_density M1 M2 p X Y y f_xy f_y f_cond ⇒
    (f_xy (x,y) = f_cond y x * f_y y)
```

The condition `f_y y` $\neq$ `0` is required, as this function will be used in the denominator of the conditional density and it cannot equal to 0. In addition, since we are dealing with extended-real numbers, `f_y y` cannot equal infinity. This theorem is applicable to any conditional density function that satisfies the given conditions.

The second step in verifying the expression of the CSP is by verifying that the probability of the joint random variables is equal to the iterated integrals of the joint density function. This can be expressed as:

$$Pr(X,Y)^{-1}(A) = \int\int \mathbf{1}_A \times f_{XY}(x,y)dx\ dy \tag{15}$$

We use Theorem 4.6 to verify this in HOL4 as:

**Theorem 4.11.**

```
⊢ ∀p X Y f_xy A.
    distributed p (pair_measure lborel lborel) (λx.  (X x, Y x)) f_xy ∧
    prob_space p ∧ (∀x.  0 ≤ f_xy x) ∧
    A ∈ measurable_sets (pair_measure lborel lborel)⇒
    (prob p (PREIMAGE (λx.  (X x, Y x)) A ∩ p_space p) =
     pos_fn_integral lborel
        (λy.
          pos_fn_integral lborel
             (λx.  indicator_fn A (x,y) * f_xy (x,y))))
```

Then, we express the probability of the joint random variables using the conditional density function as:

$$Pr(X,Y)^{-1}(A) = \int\int \mathbf{1}_A \times f_{(X|Y=y)}(x) \times f_Y(y)\ dx\ dy \tag{16}$$

We verify this in HOL4, using Theorems 4.10 and 4.11, as:

**Theorem 4.12.**

```
⊢ ∀p X Y f_xy f_y f_cond A.
    (∀y.  cond_density lborel lborel p X Y y f_xy f_y f_cond) ∧
    prob_space p ∧ (∀x.  0 ≤ f_xy x) ∧
    (∀y.  0 < f_y y ∧ f_y y ≠ PosInf) ∧
    A ∈ measurable_sets (pair_measure lborel lborel)⇒
    (prob p (PREIMAGE (λx.  (X x, Y x)) A ∩ p_space p) =
     pos_fn_integral lborel
        (λy.
          pos_fn_integral lborel
             (λx.  indicator_fn A (x,y) * f_cond y x * f_ y y )))
```

In order to be able to reach the final form of Equation (12), we need first to express the event set that corresponds to the integration in Equation (12) as:

$$(X,Y)^{-1}\{(x,y) \mid y < x \wedge x \leq t \wedge 0 \leq y \wedge y \leq t\} \tag{17}$$

We verify in HOL4 that this set corresponds to the `DFT_event` of the CSP gate as:

**Lemma 4.11.**

```
⊢ ∀X Y p t.
    rv_gt0_ninfinity [X; Y] ∧ 0 ≤ t ⇒
    (DFT_event p (CSP Y X) t =
     PREIMAGE (λs.  (real (X s), real (Y s)))
        {(x,y)| y < x ∧ x ≤ t ∧ 0 ≤ y ∧ y ≤ t} ∩ p_space p)
```

In addition, we verify that the event set in Lemma 4.11 is measurable in `pair_measure lborel lborel`. Finally, we verify that the indicator function of the set in Lemma 4.11 can be expressed as the multiplication of two indicator functions to determine the boundaries of the iterated integrals in Equation (12) as:

**Lemma 4.12.**

```
⊢ ∀x y t.
    indicator_fn {(w,u) | u < w ∧ w ≤ t ∧ 0 ≤ u ∧ u ≤ t} (x,y) =
    indicator_fn {w | y < w ∧ w ≤ t} x *
    indicator_fn {u | 0 ≤ u ∧ u ≤ t} y
```

Using all these verified theorems and lemmas, we formally verify Theorem 4.8.

**Proof Strategy for Theorem 4.9 (WSP Gate)**

For the verification of Theorem 4.9, it is evident that the probability expression involves the probability of the CSP gate in addition to the probability of the *after* expression of Theorem 4.4. Therefore, we choose to verify that the event of the WSP for basic events is equivalent to the union of two sets as:

**Lemma 4.13.**

```
⊢ ∀p Y X_a x_d t.
    (∀s.  0 ≤ Y s) ∧ (∀s.  ALL_DISTINCT [X_a s; X_d s; Y s]) ∧
    (D_AND X_a X_d = NEVER) ⇒
    (DFT_event p (WSP Y X_a X_d) t =
      {s | Y s < X_a s ∧ X_a s ≤ Normal t ∧
          0 ≤ Y s ∧ Y s ≤ t} ∩ p_space p ∪
      {s | X_d s < Y s ∧ Y s ≤ Normal t } ∩ p_space p)
```

Then, we verify that the above two sets are disjoint. As these two sets are disjoints then the probability of the original set is equivalent to the sum of the probabilities of the disjoint sets. Based on this, we verify that the probability of the first set (`{s | Y s < X_a s ∧ X_a s ≤ Normal t ∧ 0 ≤ Y s ∧ Y s ≤ t} ∩ p_space p`) is equal to the probability of the CSP gate, which will result in the first term in the addition of the conclusion of Theorem 4.9. We also verify that the probability of the second set in Lemma 4.13 (`{s | X_d s < Y s ∧ Y s ≤ Normal t} ∩ p_space p`)) is expressed using Theorem 4.4, which will result in the second term of the addition of the conclusion of Theorem 4.9. As a result, we have the probability of the WSP as in Theorem 4.9.

In this section, we formally verified the probabilistic behavior of the DFT gates: AND, OR, HSP, FDEP, PAND, CSP, WSP and the Before operator besides the formalization of expressions for $Pr(X < Y \wedge Y \leq t)$ and $Pr(X < Y \wedge X \leq t)$.

These verified properties are generic, i.e., universally quantified for all distribution and density functions, and can be used to formally verify the probability of failure expression of any DFT. The HOL4 proof script for this verification as well as the gate definitions is available at [7].

## 4.3 Summary of Formalization Challenges

In this section, we summarize the main challenges that we faced during our formalization of the DFT gates, which allows us to formally analyze DFTs in a theorem prover.

The first challenge is resolving the data-types issue. The problem in the data-types is that the gates and operators are defined as functions that return `extreal`. This is mainly required because we need to model $+\infty$ that represents the NEVER condition. However, this data-type cannot be used to represent random variables over the `lborel` measure. Any random variable defined from a probability space to the `lborel` measure should return `real` data-type. This is required because we need to integrate the density and distribution functions over the real line. Therefore, we need random variables that return `extreal` to model the gates but at the same return `real` to be used with `lborel`. We resolved this issue by using `extreal` to model the gates, but when we are conducting the probabilistic analysis we use the real version of the random variable `(λ. real (X s))`.

Secondly, after modeling the DFT and expressing the structure function of the top event using the DFT gates and operators, it is required to conduct the probabilistic failure analysis of the top event. However, the structure function cannot be used directly since it is a time-to-failure function not a set. Furthermore, in [12], there is no clear information on how to create the DFT event and link it to the structure function of the DFT top event or any other event in the fault tree. Using our formalization, we have been able to clearly and formally define a `DFT_event` that is used to create the set of moments of time until the time of failure $t$, as explained in Definition 4.1.

Thirdly, the probabilities of the AND and OR gates are directly presented in [12] as the probability of the intersection and union (Equations (6a) and (6b), respectively). However, the AND and the OR are defined using the maximum and minimum of their input operands, respectively. There is no information in [12] on how the AND and OR gates are related to the intersection and union of the input events. Using our formalization, we have been able to verify the relationship between the AND and the interaction of the input events utilizing our defined `DFT_event`. In a similar way, we verified the relationship between the OR gate and the union of the input events.

Another contribution is represented by introducing a formal proof in a theorem prover for the probability of failure of the PAND and Before operator, which are represented by $Pr(X < Y)$ in both forms, i.e., $Pr(X < Y \wedge Y \leq t)$ and $Pr(X < Y \wedge X \leq t)$. As mentioned earlier, the first proof of these ($Pr(X < Y \wedge Y \leq t)$) is not provided in [12], while the second one ($Pr(X < Y \wedge X \leq t)$) is presented in a different manner that involves derivatives. In our formalization, we presented, for the first time, the formal proof for $Pr(X < Y)$ in both its formats, i.e., $Pr(X < Y \wedge Y \leq t)$ that represents the probability of the PAND gate for basic events; and $Pr(X < Y \wedge X \leq t)$ that represents the probability of the before operator. In addition, we presented a formal proof for the probability of the $WSP$ and $CSP$

gates based on conditional density functions, which we defined, while the proof of these gates is presented in [12] based on the law of total expectation.

Finally, while performing all of these formalizations and proofs in HOL, we identified several missing assumptions or conditions that were required to ensure the correctness of the theorems. For example, ensuring the proper values for the input random variables that represent the time-to-failure functions of the system components. These important assumptions were either unavailable in [12] or are not explicitly presented as a requirement in the final form of the theorems in [12].

It is important to highlight that the main benefit of having the formalization of DFT in higher-order logic is that it enables conducting the formal DFT analysis within the sound environment of a theorem prover, which is very useful in the context of safety-critical systems.

## 5 Formal Verification of the Cardiac Assist System

In order to illustrate the utilization of our formalized probabilistic behavior of the gates and operators in the last section, we present a DFT-based formal failure analysis of the Cardiac Assist System, shown in Figure 1 [5].

We first provide generic steps that can be followed in order to use our formalization of the DFTs to conduct the formal analysis of DFTs in the form of generic expressions of failure probabilities. These steps are:

1. Determine the structure function of the top event of the DFT.
2. Simplify the structure function and formally verify that the simplified version is equal to the original function obtained in step (1).
3. Create the `DFT_event` of the structure function.
4. Express the `DFT_event` of the top event as the union of multiple input events.
5. Apply the probabilistic PIE to the union of events generated in the previous step, then simplify the result of the PIE. This will result in having the summation of the probabilities of the intersection of the different events that contribute to the failure of the top event of the DFT.
6. Replace each term in the result of the PIE by its probabilistic expression based on the verified expressions in Section 4 for each gate and operator.

Step (5) requires proving many lemmas that are necessary for manipulating the result of the PIE. For example, we need to verify the associativity property of addition for a large group of numbers (in case of the Cardiac Assist system, we verified

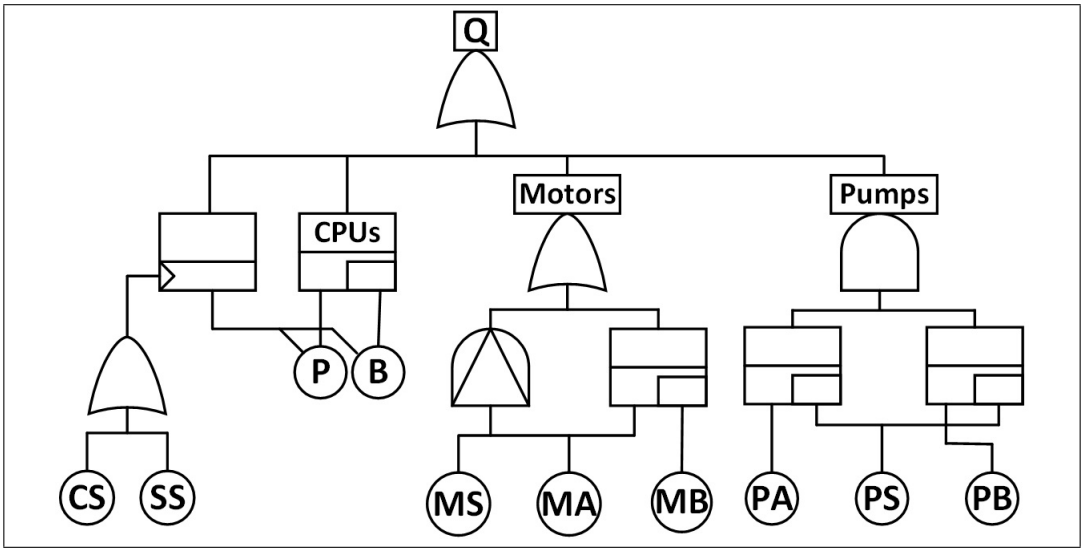



Figure 1: Cardiac Assist System

this property for 63 numbers). Although this seems a trivial task, it requires dealing with `extreal` numbers, which includes proving that for all combinations of the inputs, the result of the addition cannot equal to $\infty$. Step (5) also requires verifying the power set of events in a recursive way to generate a set of all combinations of the events, which is required by the PIE. Moreover, based on the independence of the input random variables, we need to verify the independence of several combinations of random variables (in the Cardiac Assist system, we verified that any two random variables out of the ten are independent, then three out of ten,... etc). We have verified these generic lemmas and they can be easily reused with other similar case studies and thus can reduce the proof efforts significantly.

In the rest of this section, we illustrate the utilization of the previous steps to perform the formal DFT analysis of the Cardiac Assist System to provide a generic expression for the probability of failure of the top event. The Cardiac Assist system consists of three main subsystems: pumps, motors and CPUs. The system has two main pumps ($PA$ and $PB$) with a shared spare $PS$. It has three motors $MS$, $MA$, and $MB$, where $MB$ replaces $MA$ after failure. Finally, the system has one main CPU ($P$) and a spare CPU ($B$). Both CPUs are functionally dependent on a trigger, which is the union of the crossbar switch ($CS$) and the system supervisor ($SS$). In this case study, we are assuming that the spare gates are HSPs.

Our goal is to verify the probability of failure of the Cardiac Assist system by applying the probabilistic PIE considering that the input events are independent.

This can be represented mathematically as:

$$\begin{aligned} Pr(Q) =& F_{CS}(t) \ + \ F_{SS}(t) \ + \ \int_0^t f_{MA}(y) \times F_{MS}(y)\ dy \ + \\ & F_{MA}(t) \times F_{MB}(t) \ + \ F_P(t) \times F_B(t) \ + \ F_{PA}(t) \times F_{PB}(t) \times F_{PS}(t) \\ & - \ldots + \ldots - \ F_{CS}(t) \ \times F_{SS}(t) \times \Big( \int_0^t f_{MA}(y) \times F_{MS}(y)\ dy \Big) \\ & \times F_{MA}(t) \times F_{MB}(t) \times F_P(t) \times F_B(t) \times F_{PA}(t) \times F_{PB}(t) \times F_{PS}(t) \end{aligned} \tag{18}$$

We verify Equation (18) for generic probability CDF and PDF in HOL4 as:

**Theorem 5.1.**

```
⊢ ∀CS SS MA MS MB P B PA PB PS p t f_MA.
    0 ≤ t ∧ prob_space p ∧
    ALL_DISTINCT_RV [CS;SS;MA;MS;MB;P;B;PA;PB;PS] p t ∧
    indep_vars_sets [CS;SS;MA;MS;MB;P;B;PA;PB;PS] p t ∧
    distributed p lborel (λs.  real (MA s)) f_MA ∧ (∀y.  0 ≤ f_MA y) ∧
    cont_CDF p (λs.  real (MS s)) ∧
    measurable_CDF p (λs.  real (MS s)) ⇒
    (prob p
       (DFT_event p
           ((shared_spare PA PB PS PS)·(shared_spare PB PA PS PS)+
           (PAND MS MA)+(HSP MA MB)+
           (HSP (FDEP(CS + SS) P)(FDEP(CS + SS) B))) t) =
     CDF p (λs.  real (CS s)) t + CDF p (λs.  real (SS s)) t +
     pos_fn_integral lborel
        (λy.
           f_MA y * (indicator_fn {u | 0 ≤ u ∧ u ≤ t} y *
            CDF p (λs.  real (MS s)) y)) +
     CDF p (λs.  real (MA s)) t * CDF p (λs.  real (MB s)) t +
     CDF p (λs.  real (P s)) t * CDF p (λs.  real (B s)) t +
     CDF p (λs.  real (PA s)) t * CDF p (λs.  real (PB s)) t *
     CDF p (λs.  real (PS s)) t - ....+...-
     CDF p (λs.  real (CS s)) t * CDF p (λs.  real (SS s)) t *
     pos_fn_integral lborel
        (λy.
           f_MA y * (indicator_fn {u | 0 ≤ u ∧ u ≤ t} y *
            CDF p (λs.  real (MS s)) y)) *
     CDF p (λs.  real (MB s)) t * CDF p (λs.  real (P s)) t *
     CDF p (λs.  real (B s)) t * CDF p (λs.  real (PA s)) t *
     CDF p (λs.  real (PB s)) t * CDF p (λs.  real (PS s)) t)
```

where `0 ≤ t` ensures that the time $t$ is greater than or equal to 0, `prob_space p` indicates that $p$ is a probability space, `ALL_DISTINCT_RV` is a predicate which ensures that all inputs and their event sets are not equal and their values are greater than or equal to 0 but they cannot equal $+\infty$. This assumption is a realistic one, since for any component in a system the time of failure will always be greater than or equal to 0 and the component will eventually fail. The predicate `indep_vars_sets` adds the condition that all random variables and their event sets are independent. The predicate (`distributed p lborel (λs. real (MA s)) f_MA`) indicates that the real random variable of $MA$ has the density function `f_MA`. The last two predicates in the goal ensures that the CDF of the real random variable of $MS$ is continuous and measurable from the real line to the extreal one (real-borel to extreal-borel).

We verify several intermediate lemmas to prove Theorem 5.1. We first verify a reduced form of the given DFT and, then we verify the probability expression of the verified reduced version.

**Lemma 5.1.**

```
⊢ ∀CS SS MA MS MB P B PA PB PS.
    (∀s.  ALL_DISTINCT [MA s; MS s; PA s; PB s; PS s]) ⇒
    ((shared_spare PA PB PS PS)·(shared_spare PB PA PS PS) +
     (PAND MS MA) +
     (HSP MA MB)+(HSP (FDEP(CS + SS) P)(FDEP(CS + SS) B)) =
     CS + SS + (MA·(MS ◁ MA)) + MA·MB + P·B + PA·PB·PS)
```

In the above lemma, `(shared_spare PA PB PS PS)·(shared_spare PB PA PS PS)` represents the pumps part of the DFT, `(PAND MS MA)+(HSP MA MB)` represents the motors parts and finally the CPUs part is represented by `(HSP (FDEP(CS + SS) P)(FDEP(CS + SS) B)`. The predicate `ALL_DISTINCT` ensures that all basic events cannot fail at the same time. Since we assumed that all spare gates are HSPs, the spare input $PS$ for the shared spare gates is the same for both the active and dormant states.

In order to find the probability of the top event, we utilize the formally verified reduced version of the structure function and encapsulate it into a `DFT_event`, as the probability can only be applied to sets. To utilize the probabilistic PIE, we express the `DFT_event` of the Cardiac Assist system as the union of events.

**Lemma 5.2.**

```
⊢ ∀PA PB PS MS MA MB CS SS P B p t.
    DFT_event p
       (CS + SS + (MA·(MS ◁ MA)) + MA·MB + P·B + 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 (MA·MB) t;
      DFT_event p (P·B) t; DFT_event p (PA·PB·PS) t]
```

From Lemma 5.2, we can notice that the top event is constructed from the union of six different sets. Applying the probabilistic PIE on the union of these sets (6 sets) generates 63 different terms (combinations). We verify several lemmas to be able to use the theorem of the probabilistic PIE [1] for the union list of these six sets. For example, we formally verify that:

**Lemma 5.3.**

```
⊢ ∀A B C D E K.
    {t | t SUBSET {A; B; C; D; E; k} ∧ t ≠ {}} =
    {{A}; {B}; {C}; {D}; {E}; {k}; {A; B}; {A; C};...;
     {A; B; C; D; E; k}}
```

The result of Lemma 5.3 is a set of 63 different sets. We had to apply the `SIGMA` function that results from the `sum_set` in the PIE theorem. Therefore we verify the following lemma for 63 sets.

**Lemma 5.4.**

```
⊢ ∀A B C D E K.
    ALL_DISTINCT [A;B;C;D;E;k] ∧
    (∀x.  x ∈{{A};{B};{C};{D};{E};{k};...;{A; B; C; D; E; k}} ⇒
       f x ≠ PosInf) ⇒
    (SIGMA f {{A};{B};...;{A; B; C; D; E; k}} =
     f {A} + f {B} +...+ f {A; B; C; D; E; k}
```

After verifying all these lemmas and based on the reduced DFT expression we are able to verify the probability of the Cardiac Assist system (Equation 18) into Theorem 5.1.

The first part of the conclusion of Theorem 5.1 corresponds to the original DFT (without reduction). In the verification of this theorem, we use Lemma 5.1 to replace the original DFT with the reduced one. Then, we use Lemma 5.2 to represent the `DFT_event` as a union list. After representing the left-hand side of the conclusion of Theorem 5.1 as a union list, we use Lemmas 5.3, 5.4 and the probabilistic PIE theorem [1] to prove this goal. After applying the probabilistic PIE, the resulting 63 subgoals should be proven based on the verified theorems of the probability of DFT gates. Therefore, applying the probabilistic PIE will not directly verify the current theorem, it is rather required to verify several intermediate subgoals after applying the PIE. In addition, after applying the PIE, it is necessary to apply the simplification theorems again since the application of the PIE results in intersecting the events. This means that further simplifications needed to be done. The first 6 terms in the right-hand side of the conclusion of Theorem 5.1 correspond to the probability of the elements of the list in Lemma 5.2. For example, `CDF p (λs. real (CS s)) t` represents the probability of `DFT_event p CS t`, which is $F_{CS}(t)$, according to Theorem 4.1. `pos_fn_integral lborel(λy. f_MA y *(indicator_fn {u |0 ≤ u ∧ u ≤ t} y * CDF p (λs. real (MS s)) y))` represents the probability of `DFT_event p (MA·(MS ◁ MA)) t`, which is $\int_0^t f_{MA}(y) \times F_{MS}(y)\, dy$, according to Theorem 4.4. The following terms in the conclusion of Theorem 5.1 correspond to finding the probability of the intersection of each pair in the list, then each 3 elements then 4 elements until we reach the last term in the right-hand side of the goal, which corresponds to the probability of the intersection of all elements in the list. Since all six elements in the union list are independent, the probability of their intersection is equal to the multiplication of the individual probabilities. The verification of Theorem 5.1 required around 6000 lines of proof script and took only 40 man-hours. This process was significantly facilitated thanks to the availability of the formally verified intermediate lemmas. The proof effort for the formal verification of these lemmas involved 12000 lines of code and about 320 man-hours. These lemmas can be easily reused for verifying similar systems that can be represented as the union of 6 events. In addition, these lemmas can also be useful in verifying larger systems based on similar proof steps identified above. As a future work, we plan to use machine learning techniques to automate the proof process of similar systems. This would facilitate the reusability of this work with users who are not much familiar with HOL or the underlying details of our verified theorems.

It is important to note that we have been able to verify the probability of the Cardiac Assist system for generic distributions and density functions, which can be instantiated later with specific functions according to the required constraints, without any need to repeat the whole process from the beginning. It is worth men-

tioning that such results cannot be obtained using PMCs, as they can only generate the probability of failure after specifying the failure rates of the components. In addition, PMCs are only limited to exponential distribution which does not consider the aging factor in any system. However, using our formalization for generic expression, it can be used with any probability distribution and density function as long as they are integrable, which makes it a more general and accurate alternative to the existing techniques.

## 6 Conclusions

In this paper, we proposed to conduct the probabilistic analysis of DFTs within the HOL4 theorem prover and thus obtain formally verified probability of failure expressions for generic probability distributions and density functions. We verified many simplification theorems for DFT gates and operators that allow formal reasoning about the reduction of the structure function of the DFT top event into a simpler form. In particular, we verified the probability of the intersection and the union of independent events to provide the probability of the AND, OR, FDEP and HSP gates. Moreover, we verified the probability of a sequence of two failing events ($Pr(X < Y)$) in two forms, i.e, $Pr(X < Y \wedge Y \leq t)$ and $Pr(X < Y \wedge X \leq t)$, which, to the best of our knowledge, is another novel contribution. These expressions are used to formally express the probability of the PAND gate and the before operator. Similarly, we also verified the probabilistic behavior of the spare gates, which required dealing with dependent events and conditional density functions. To illustrate the effectiveness of our formalization, we presented the formal analysis of the Cardiac Assist System, which is a safety-critical system. Using our formalization, we were able to provide generic results for the probability of failure of this system, i.e., for any distributions and density functions. It is evident that such results cannot be obtained using simulation nor using model checking. This highlights the importance of our proposed work, besides the fact that it inherits the sound and expressive nature of HOL theorem proving.

## References


[1] W. Ahmad and O. Hasan. Towards Formal Fault Tree Analysis using Theorem Proving. In *Intelligent Computer Maths.*, LNCS 9150, pages 39–54. Springer, 2015.

[2] 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*, LNCS 9984, pages 264–279. Springer, 2016.

[3] H. Bauer. *Probability Theory.* Walter de Gruyter, 1996.

[4] P. Billingsley. *Probability and Measure.* John Wiley & Sons, 2012.

[5] H. Boudali, P. Crouzen, and M. Stoelinga. A Rigorous, Compositional, and Extensible Framework for Dynamic Fault Tree Analysis. *IEEE Transactions on Dependable and Secure Computing*, 7:128–143, 2010.

[6] C. Dehnert, S. Junges, J.P. Katoen, and M. Volk. A Storm is Coming: A Modern Probabilistic Model Checker. In *Computer Aided Verification*, LNCS 10427, pages 592–600. Springer, 2017.

[7] Y. Elderhalli. DFT Formal Probabilistic Analysis: HOL4 Script, Concordia University, Montreal, QC, Canada, http://hvg.ece.concordia.ca/code/hol/DFT/index.php, 2018.

[8] 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*, LNCS 10811, pages 139–156. Springer, 2018.

[9] M. Ghadhab, S. Junges, J.P. Katoen, M. Kuntz, and M. Volk. Model-based Safety Analysis for Vehicle Guidance Systems. In *Computer Safety, Reliability, and Security*, LNCS 10488, pages 3–19. Springer, 2017.

[10] HOL4. https://hol-theorem-prover.org/, 2018.

[11] J. Hölzl. *Construction and Stochastic Applications of Measure Spaces in Higher-Order Logic.* PhD thesis, Technische Universität München, Germany, 2012.

[12] G. Merle. *Algebraic Modelling of Dynamic Fault Trees, Contribution to Qualitative and Quantitative Analysis.* PhD thesis, ENS, France, 2010.

[13] T. Mhamdi. *Information-theoretic Analysis using Theorem Proving.* PhD thesis, Concordia University, Montreal, QC, Canada, 2012.

[14] T. Mhamdi, O. Hasan, and S. Tahar. On the Formalization of the Lebesgue Integration Theory in HOL. In *Interactive Theorem Proving*, LNCS 6172, pages 387–402. Springer, 2010.

[15] T. Mhamdi, O. Hasan, and S. Tahar. Formalization of Entropy Measures in HOL. In *Interactive Theorem Proving*, LNCS 6898, pages 233–248. Springer, 2011.

[16] T. Mhamdi, O. Hasan, and S. Tahar. Formalization of Measure Theory and Lebesgue Integration for Probabilistic Analysis in HOL. *ACM Transactions on Embedded Computing Systems*, 12(1):13, 2013.

[17] C.Z. Mooney. *Monte Carlo Simulation.* Sage, 1997.

[18] J. Ni, W. Tang, and Y. Xing. A Simple Algebra for Fault Tree Analysis of Static and Dynamic Systems. *IEEE Transactions on Reliability*, 62(4):846–861, 2013.

[19] L. Pullum and J.B. Dugan. Fault Tree Models for the Analysis of Complex Computer-based Systems. In *IEEE Reliability and Maintainability Symposium*, pages 200–207, 1996.

[20] M. Qasim. Formalization of Normal Random Variables. Master's thesis, Concordia University, Montreal, QC, Canada, 2016.

[21] M. Qasim, O. Hasan, M. Elleuch, and S. Tahar. Formalization of Normal Random Variables in HOL. In *Intelligent Computer Mathematics*, LNCS 9791, pages 44–59. Springer, 2016.

[22] E. Ruijters and M. Stoelinga. Fault Tree Analysis: A Survey of the State-of-the-art in Modeling, Analysis and Tools. *Computer Science Review*, 15-16:29 – 62, 2015.

[23] M. Stamatelatos, W. Vesely, J.B. Dugan, J. Fragola, J. Minarick, and J. Railsback. *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, 2002.

[24] K. J. Sullivan, J. B. Dugan, and D. Coppit. The Galileo Fault Tree Analysis Tool. In *IEEE Symposium on Fault-Tolerant Computing*, pages 232–235, 1999.

 Received 20 July 2018