A Framework for Formal Dynamic Dependability Analysis Using HOL Theorem Proving

Yassmeen Elderhalli(✉), Osman Hasan, and Sofiène Tahar

Electrical and Computer Engineering, Concordia University, Montréal, Canada
{y_elderh,o_hasan,tahar}@ece.concordia.ca

Abstract. Dependability analysis is an essential step in the design process of safety-critical systems, where the causes of failure and some other metrics, such as reliability, should be identified at an early design stage. The dynamic failure characteristics of real-world systems are usually captured by various dynamic dependability models, such as continuous time Markov chains (CTMCs), dynamic fault trees (DFTs) and dynamic reliability block diagrams (DRBDs). In order to conduct the formal dependability analysis of systems that exhibit dynamic failure behaviors, these models need to be captured formally. In this paper, we describe recent developments towards this direction along with a roadmap on how to be able to develop a framework for formal reasoning support for DFTs, DRBDs and CTMCs in a higher-order-logic theorem prover.

Keywords: Dynamic dependability analysis · Dynamic fault trees · Dynamic reliability block diagrams · Continuous time Markov chains · HOL theorem proving

1 Introduction

Dependability is a general concept that encompasses many attributes, such as reliability, availability, security and safety [1]. Reliability is the ability of a system to provide a correct service within a given period of time [1] and it is quantified by evaluating the probability of delivering such service. On the other hand, availability is the probability of a system or component to provide its correct service at a given moment of time [1]. Many real-world systems exhibit sequential failures and dependencies among system components that cannot be captured using traditional dependability models, such as static fault trees (SFTs) and static reliability block diagrams (SRBDs). Therefore, dynamic dependability models are used to capture the dynamic failure behavior of these systems. These models include Continuous time Markov chains (CTMCs) [2], dynamic fault trees (DFTs) [3] and dynamic reliability block diagrams (DRBDs) [4].

Dynamic dependability analysis identifies the sequences of failure of system components and their effect on the overall system behavior. This helps devising

C. Benzmüller and B. Miller (Eds.): CICM 2020, LNAI 12236, pp. 105–122, 2020.
https://doi.org/10.1007/978-3-030-53518-6_7

solutions to enhance the overall system dependability. Therefore, this analysis process should be handled carefully in a sound environment to produce accurate results. Traditionally, dependability models are analyzed using paper-and-pencil based proof methods or using simulation. The former provides a flexible way to model and analyze systems, but it is prone to human errors. On the other hand, simulation provides an easy automated method to conduct the analysis, which justifies its common use in analyzing a wide range of applications. However, due to the high computational cost of simulation, only part of the space could be analyzed, and thus the results cannot be termed as accurate or complete.

Formal methods, such as model checking [5] and theorem proving [6], have been used for the analysis of dependability models, to overcome the inaccuracy limitations of the above-mentioned techniques. For example, the STORM model checker [7] has been successfully used in the safety analysis of a vehicle guidance system [8] using DFTs. Although probabilistic model checkers (PMCs) provide an automatic way to conduct the analysis of dependability models, the state space explosion problem often limits its scope especially when analyzing complex systems. Moreover, the reduction algorithms embedded in these tools are usually not formally verified, which questions the accuracy of the reduced models [9]. This becomes an issue when analyzing safety-critical systems, where the smallest error cannot be tolerated. More importantly, PMCs inherently assume the failures to be exponentially distributed for system components [10], and thus cannot capture, for example, their aging factor. Although higher-order logic (HOL) theorem proving has been used in the analysis of traditional (static) dependability models, such as SFTs [11] and SRBDs [12], these models cannot capture the dynamic aspects of real-world systems and thus cannot fulfill the objective of the projected work. However, using a theorem prover in the analysis allows having verified, within a sound environment, generic expressions of dependability that are independent of the distribution of system components. Thus, the results are not limited to exponential distributions as with PMCs.

In this paper, we describe an ongoing project for building a complete framework to formally analyze dynamic dependability models using HOL theorem proving. The project had started at the Hardware Verification Group of Concordia University in 2017 with a clear roadmap. In this regard, we formalized DFTs in the HOL4 theorem prover [13], which allows formally verifying generic expressions of the probability of failure of DFTs [14] and thus can be used to conduct formal dynamic dependability analysis. Furthermore, we proposed a novel algebra to analyze DRBDs with certain structures and verified their mathematical foundations using the HOL4 theorem prover [15]. On the other hand, a formalization for general CTMCs [16] is available in Isabelle/HOL [17]. However, this formalization has not been used in the context of dynamic dependability analysis. Therefore, we plan to develop a CTMC formalization to be used in this context. The present paper mainly summarizes our developed formalization and shares our plans for the development of a complete framework for the formal dependability analysis of real-world systems using a theorem prover.

The rest of the paper is structured as follows: Sect. 2 presents our proposed framework for the formal dynamic dependability analysis. Section 3 provides a brief description of our formalization of DFTs. In Sect. 4, we present DRBDs and our developed DRBD algebra. Section 5 presents the required mathematical foundations of the CTMC dependability analysis. We report the current status of the project and the remaining milestones in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Proposed Framework

Figure 1 shows an overview of our proposed framework for formal dynamic dependability analysis. This framework provides verified generic expressions of dependability in the HOL4 theorem prover using DFTs, DRBDs and CTMCs. The analysis starts by having a system description with some dependability requirements, such as a certain expression of reliability. The dependability of this system can be modeled using a DFT, DRBD or CTMC model according to its description. For the case of the DFTs and DRBDs, we need, respectively, a library of formalized DFT gates and DRBD constructs besides their simplification theorems and verified probabilistic behavior. For the CTMC formal analysis, it is required to have both formal transient and steady state analyses. The formal DFT and DRBD models can be analyzed qualitatively or quantitatively. In the former, the sources of vulnerabilities of the system are verified by identifying the cut sets and cut sequences. In the latter, we prove generic failure and reliability expressions of DFT and DRBD based systems, respectively. It is worth mentioning that unlike PMC approaches, the formally verified generic expressions of DFT and DRBD are independent of the probability distributions of the system

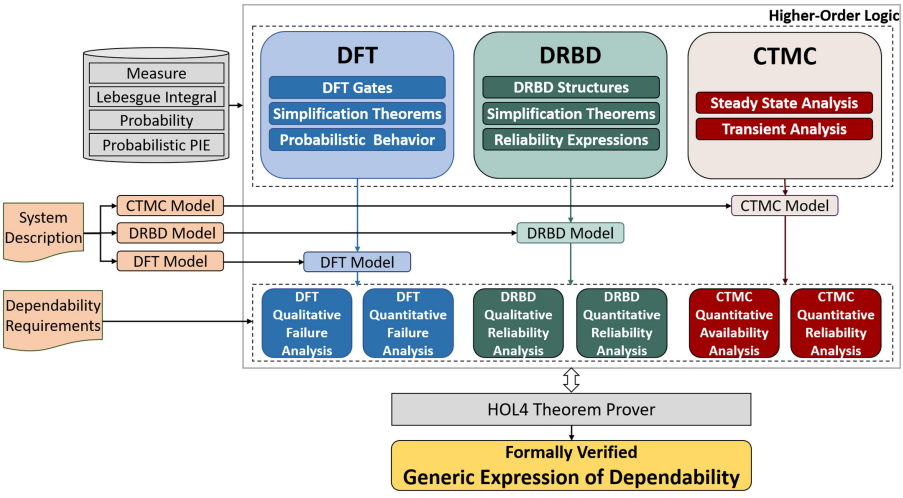
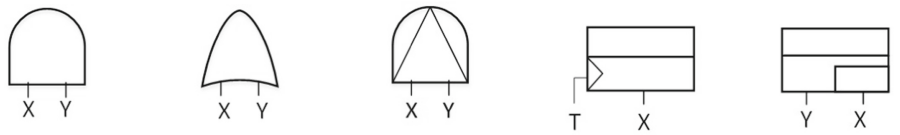


Fig. 1. Overview of the formal dependability analysis framework



(a) AND (b) OR (c) PAND (d) FDEP (e) SPARE

Fig. 2. DFT gates

components. For CTMC based models, the proposed framework formally analyzes availability and reliability metrics by proving generic expressions of these dependability metrics that are independent of the failure rates of system components. We choose HOL4 in the development of the formalization of dynamic dependability models as this would facilitate using some of the available theories, such as the probabilistic PIE [18], Lebesgue integral [19] and probability [20].

Our ultimate goal in this project is to develop a tool that accepts the dependability model in either a graphical or simple textual format. Then, using a parser, the tool creates the HOL formal models of these formats that can be used in the formal analysis using a HOL theorem prover. The aim of this tool is to reduce the user interaction with the theorem proving environment, which would facilitate the usage of this framework by users who are not familiar with HOL theorem proving or the underlying mathematical foundations of the dependability models. This requires invoking several techniques, such as machine learning, to automatically (to a certain extent) verify the required expressions. Therefore, this proposed framework will allow conducting the dynamic dependability analysis of many real-world systems to provide generic expressions. We highlight the details of the proposed framework in the following sections including the current status of the formalization and provide some insights about the remaining steps.

3 Dynamic Fault Trees

Dynamic fault trees (DFTs) [3] model the failure dependencies among system components that cannot be captured using traditional SFTs. A DFT is a graphical representation of the sources of failure of a given system. The modeling starts by an undesired top event that represents the failure of a system or a subsystem. DFT inputs represent basic events that contribute to the occurrence (failure) of the top event. The relationships and dependencies among these basic events are modeled using DFT gates (Fig. 2). For example, the output event of the Priority-AND (PAND) gate occurs when both input events occur in sequence.

Fault tree analysis (FTA) can be generally carried out qualitatively and quantitatively [3]. In the *qualitative* analysis, the combinations and sequences of basic events that contribute to the occurrence of the top event (failure of the system) are identified. These combinations and sequences represent the cut sets and cut sequences [21], respectively. In the *quantitative* analysis, attributes, such as the mean-time-to-failure (MTTF) and the probability of failure, can be evaluated

based on the failure distribution of the basic events and their relationships. Dynamic FTA has been commonly conducted using some sort of a DFT algebra (e.g., [22]) or by analyzing the corresponding CTMC of the given DFT [3]. In the former method, an algebra similar to the ordinary Boolean algebra is defined with some temporal operators and simplification properties that allow reducing the structure function of the top event. Based on this function, both the qualitative and quantitative analyses can be carried out, where the probability of failure of the DFT's top event can be expressed based on the failure distribution of the basic events. On the other hand, the given DFT can be converted into its equivalent CTMC, which can then be analyzed to find the probability of failure of the top event [3]. Complex systems can generate CTMCs with a large state space that can be handled by applying a modulerization approach, where the DFT is divided into static and dynamic parts. The static part can be analyzed using one of the conventional methods, such as binary decision diagrams (BDDs) [21]. The dynamic part can then be analyzed by converting it to its corresponding CTMC. This kind of modulerization is implemented in the Galileo tool [23].

The arithmetic foundations of the algebraic approach of [22] were not formally verified, which puts a question mark on the soundness of the reported results. In [24], we proposed to formalize this DFT algebra in higher-order logic theorem proving and developed an integrated methodology to conduct DFT's qualitative analysis using the HOL4 theorem prover and quantitative analysis using the STORM model checker. However, generic expressions of probability of failure cannot be obtained based on this methodology as a PMC is involved in the quantitative analysis. Moreover, our definitions in [24] could not cater for the DFT probabilistic analysis. Therefore, in [14,25], we improved our definitions of DFT gates to conduct both the DFT qualitative and quantitative analyses in the form of generic expressions in a theorem prover. Next, we provide the description of the DFT algebra and its formalization in order to have a better understanding of the first part of our proposed framework of Fig. 1.

3.1 DFT Operators and Simplification Properties

The DFT algebraic approach of [22] deals with the inputs of a DFT based on their time of failure. Therefore, all elements, operators and gates are defined based on this time of failure. It is assumed that the failure of a certain component causes the occurrence of its corresponding basic event. Moreover, it is also assumed that the components are non-repairable [22]. The algebraic approach defines two identity elements, which facilitate the simplification process of the structure function of a given DFT. These are the *ALWAYS* and *NEVER* elements. The *ALWAYS* element represents an event that always occurs, i.e., from time 0, while the *NEVER* element is an event that can never occur, i.e., the time of failure is $+\infty$. In order to capture the dynamic failure in DFTs, three temporal operators are introduced; *Before* ($\lhd$), *Simultaneous* (Δ), and *Inclusive Before* ($\trianglelefteq$) [22]. The output of the before operator fails when the first input event occurs before the second. The output of the simultaneous operator fails when both input events happen at the same time. Finally, the output of the inclusive before operator

fails when the first input occurs before or at the same time of the second input. We formally defined these elements and operators in HOL4 as extended-real functions of time [14]. The purpose of choosing extended-real numbers, which are real numbers besides $\pm\infty$, is to be able to model the NEVER event that returns $+\infty$ as its time of failure. Several simplification properties are introduced in the algebraic approach [22] to simplify the structure function of DFTs (the function of the top event). This reduced structure function can then be used in the probabilistic analysis. We verified over 80 simplification theorems [24] that vary from simple theorems to more complex ones. This enables having formally verified reduced cut sets and cut sequences, i.e., formal qualitative DFT analysis.

3.2 DFT Gates

DFTs use the ordinary FT gates, i.e., AND and OR gates, besides the dynamic gates (Fig. 2). AND ($\cdot$) and OR ($+$) are used in the algebraic approach as operators as well as FT gates. The output of the AND gate fails when both inputs fail. This means that the time of occurrence of the output event of the AND gate is the maximum time of occurrence of both input events. The output of the OR gate fails when at least one of the input events occurs. Therefore, the time of occurrence equals the minimum time of occurrence of its inputs. The Priority-AND (PAND) gate is similar to the AND gate, where the output fails when both inputs fail. However, the input events should occur in a certain sequence, conventionally, from left to right. The Functional DEPendency (FDEP) gate is used to model a situation when the failure of one system component triggers the failure of another. For the FDEP gate of Fig. 2, the occurrence of T triggers the occurrence of X. Finally, the spare gate models spare parts in systems, where the main part is replaced by the spare part after failure. In [14], we formally defined these gates as functions of time to enable the verification of their failure probabilistic expressions, as will be explained in the following section.

3.3 DFT Failure Analysis

In order to conduct the formal probabilistic failure analysis of DFTs, it is required first to formally verify the probability of failure of DFT gates. It is assumed that the basic events of DFT are independent. However, in case of the spare gate, the input events are not independent as the failure of the main part affects the failure behavior of the spare. In order to perform the failure analysis, we first formally define a DFT event that represents the set of time until which we are interested in finding the probability of failure [14].

In case of independent events, four expressions are used to determine the probability of failure of DFT gates [22].

$$Pr\{X \cdot Y\}(t) = F_X(t) \times F_Y(t) \tag{1a}$$

$$Pr\{X + Y\}(t) = F_X(t) + F_Y(t) - F_X(t) \times F_Y(t) \tag{1b}$$

$$Pr\{Y \cdot (X \lhd Y)\}(t) = \int_0^t f_Y(y)\ F_X(y)\ dy \tag{1c}$$

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

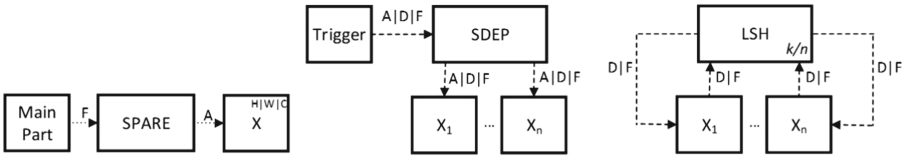
where F_X and F_Y are the cumulative density functions of random variables X and Y, respectively, and f_X and f_Y are their probability density functions.

Equation (1a) represents the probability of the AND gate. In order to verify this equation, we first verified that the event of the output of the AND gate equals the intersection of the individual input events. Then, based on the independence of these events, the probability of their intersection equals the multiplication of their probabilities, i.e., their cumulative density functions [14]. Equation (1b) represents the probability of the OR gate. We verified this equation using the fact that the event of the OR gate equals the union of the individual input events. Equations (1c) and (1d) represent the probability of two inputs events occurring one after the other or one before the other, respectively. For the first case, it is required that both events occur in sequence. Whereas the second case requires that the first input event occurs before the second. So, it is not necessary that the second input event occurs. Using these expressions, the probability of the AND gate is expressed using Eq. (1a). The probabilities of failure of the OR and FDEP gates are expressed using Eq. (1b). Moreover, Eq. (1c) represents the probability of failure of the PAND gate for basic input events. Finally, Eq. (1d) represents the probability of failure of the before operator.

Similarly, the probability of failure of the spare gate can be expressed, but it requires dealing with conditional density functions as the time of failure of the main part affects the activation time of the spare part. This means that the input events are no longer independent. We verified these expressions in HOL4 by first defining a conditional density function and then proving the probability of failure of the spare gates [14]. In our verification process, we used the measure, Lebesgue integral, Lebesgue-Borel measure and the probability theories in order to verify a generic expression of failure of a given DFT. Based on this formalization, we conducted the formal dependability analysis of several safety-critical systems, such as a cardiac assist system [26] and a drive-by-wire system [27].

4 Dynamic Reliability Block Diagrams

A dynamic reliability block diagram (DRBD) models the paths of success in a given system. System components are represented as blocks that are connected in the traditional series, parallel, series-parallel and parallel-series structures.



(a) Spare (b) State Dependency (c) Load Sharing

Fig. 3. Dynamic DRBD constructs

Additional constructs are used to model the dynamic dependencies among system blocks. The main dynamic constructs are: spare, state-dependencies and load sharing. The last two constructs enable modeling more realistic scenarios in system reliability that include the effect of activation/deactivation of one component on the rest of the components. This behavior cannot be captured using DFTs [28] as they can only capture the failure without considering the activation/deactivation effect.

Due to the dynamic nature of DRBDs, they can be analyzed by converting them into a state-space model, i.e., a Markov chain. Then, the resultant Markov chain can be analyzed using one of the traditional techniques, including analytical methods or simulation. Some tools, such as BlockSim [29], enable DRBD analysis by providing a graphical user interface to model DRBDs and conduct the analysis either analytically or using discrete event simulation. As mentioned previously, complex systems can generate Markov chains with a large number of states, which hinders the analysis process. Decomposition can be applied to divide the DRBD into a dynamic part that can be solved using Markov chains and a static part that can be analyzed using static RBD analysis techniques [30]. Although this decomposition would reduce the state space, such simulation based analysis cannot provide accurate and complete results.

The formal semantics of DRBDs were introduced in [31] using the Object-Z formalism [32]. Then, this DRBD is converted into a Colored Petri net (CPN), where it can be analyzed using existing Petri net tools. However, since the given DRBD is converted into a CPN, only state-based properties can be analyzed. In addition, generic expressions of reliability cannot be obtained, which represents our target in the proposed framework. HOL theorem proving has been only used for the analysis of traditional SRBDs [12], which cannot support the scope of the proposed framework, i.e., dynamic dependability. To the best of our knowledge, there is no support of DRBD analysis using a HOL theorem prover that can cater for the analysis of real-world systems that exhibit dynamic behavior. The main challenge towards this direction is the absence of a formal DRBD algebra that can provide similar analysis like DFTs. Therefore, we developed a novel algebra that allows conducting both the qualitative and quantitative analyses based on the structure function of DRBDs with spare constructs [15]. The formalization of this algebra in HOL facilitates the analysis using a theorem prover. Below, we provide an overview of DRBD constructs and structures.

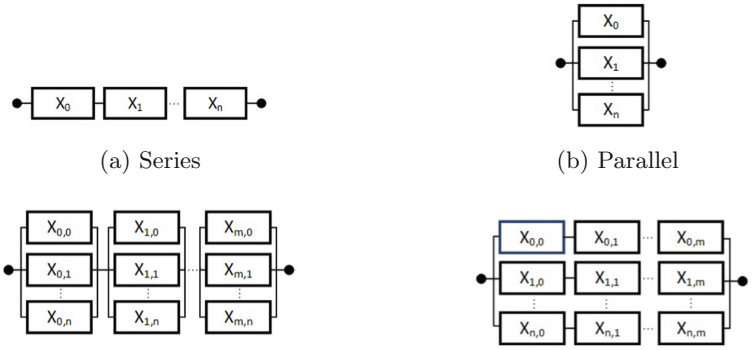
4.1 DRBD Constructs and Structures

The main dynamic DRBD constructs are shown in Fig. 3 [33]. The *spare* construct is used to model spare parts in systems, similar to the DFT spare gate. The *state dependencies* are used to model the effect of activation(A)/deactivation(D)/failure(F) among system components. In Fig. 3(b), the A/D/F of the trigger will cause the state dependency controller (SDEP) to signal the A/D/F of components $X_1 ... X_n$. Finally, the *load sharing* (LSH) construct is used to model the effect of sharing a load on the overall system failure. For example, the LSH in Fig. 3 models a load that is shared among n components. It is required that at least k out of these n components to be working in order for the successful functionality of the overall system. Therefore, the D/F of some of these components may cause the D/F of the rest of the components.

Besides the dynamic DRBD constructs, system components are represented as blocks that can be connected in series, parallel, series-parallel and parallel-series fashion, as shown in Fig. 4 [34]. Each block in Fig. 4 represents either a simple system component or one of the DRBD dynamic constructs.

4.2 DRBD Algebra

We developed a DRBD algebra to perform both qualitative and quantitative analyses in the HOL theorem proving environment of a given DRBD. The developed algebra can model traditional DRBD structures, i.e., series and parallel, besides the spare construct by dealing with the time-to-failure functions. In the developed algebra, we defined DRBD operators, like the DFT algebra, to model the various relationships among system components. These operators are: 1) AND ($\cdot$) to model a situation where two system components are required to work for a successful system behavior (connected in series); 2) OR ($+$) to model system components that are connected in parallel; 3) After operator ($\triangleright$) which



(a) Series (b) Parallel

(c) Series-Parallel (d) Parallel-Series

Fig. 4. DRBD structures

captures the situation where one system component is required to continue working after the failure of a second one; 4) Simultaneous operator (Δ) which is similar to the DFT simultaneous operator; and 5) Inclusive after ($\trianglerighteq$) that combines the behavior of both the after and simultaneous operators. In [15] we provided mathematical expressions for these operators, and expressed the DRBD structures and spare construct based on their mathematical expressions.

4.3 DRBD Reliability Analysis

In our algebra, we assume that each system block is represented by a random variable, which is the time-to-failure function of this block. We also assume that the system components are non-repairable. Based on this time-to-failure function, the reliability of a single block is defined as [34]:

$$R_X(t) = Pr(X > t) = 1 - Pr(X \leq t) = 1 - F_X(t) \tag{2}$$

The DRBD blocks can be connected in several ways depending on the success paths of the modeled system. The definitions and reliability expressions of the structures of Fig. 4 are listed in Table 1 [34]. In the *series* structure, it is required that all blocks are working for the system to work. Therefore, the series structure can be modeled as the intersection of the individual block events, as listed in Table 1, where X_i represents the DRBD event of the i^{th} block. This structure can be also modeled by ANDing the functions of these blocks. The reliability of this structure equals the multiplication of the reliability of the individual blocks. The *parallel* structure requires at least one of the blocks to be working for a successful system behavior. Hence, it is modeled as the union of the events of the individual blocks and it can be also modeled by ORing these functions. The *series-parallel* structure (Fig. 4(c)) represents a series structure of blocks each of which is a parallel structure. Therefore, it is modeled as the intersection of unions. The *parallel-series* structure (Fig. 4(d)), is a parallel structure of several series structures. It is modeled as the union of intersection of the individual block events. In [15], we formally verified these expressions besides the reliability of the spare construct. We plan to extend the DRBD algebra to model the remaining dynamic constructs, i.e., load sharing and state dependency. This requires modeling the deactivation state of system components and may include introducing new DRBD operators to capture such behavior.

Table 1. Mathematical and reliability expressions of DRBD structures

Structure	Mathematical expression	Reliability
Series	$\bigcap_{i=1}^{n} X_i$	$\prod_{i=1}^{n} R_{X_i}(t)$
Parallel	$\bigcup_{i=1}^{n} X_i$	$1-\prod_{1=1}^{n}(1-R_{X_i}(t))$
Series-parallel	$\bigcap_{i=1}^{m}\bigcup_{j=1}^{n} X_{(i,j)}$	$\prod_{i=1}^{m}(1-\prod_{j=1}^{n}(1-R_{X_{(i,j)}}(t)))$
Parallel-series	$\bigcup_{i=1}^{n}\bigcap_{j=1}^{m} X_{(i,j)}$	$1-(\prod_{i=1}^{n}(1-\prod_{j=1}^{m}(R_{X_{(i,j)}}(t))))$

5 Continuous Time Markov Chains

Continuous Time Markov Chains (CTMCs) are the most widely used stochastic processes in dynamic dependability analysis since they can capture the failure dependencies. A CTMC is a Markov process with discrete state space. The transition from one state to another can happen with a certain rate at any moment of time. Formal methods have been used in the analysis of CTMC based systems. For example, the PRISM model checker is utilized in [10] to model and analyze different case studies, such as dynamic voltage and molecular reactions. However, generic expressions cannot be obtained using such analysis. Utilizing the expressive nature of HOL, Markov chains are formalized in both HOL4 and Isabelle/HOL [17]. In [35], the formalization of discrete time Markov chains (DTMCs) is presented in HOL4 with some formalized attributes, such as steady state probabilities and stationary distribution. In [16], CTMCs are formally defined in Isabelle/HOL with the formalization of backward equations. Although Markov chains have been formalized in HOL, no work has been proposed yet regarding the dependability analysis of Markov chain based systems. Conducting the analysis of CTMCs to reason about dependability attributes, such as reliability and availability, would provide formally verified generic expressions of dependability, which is the scope of the third part of our proposed framework. In the sequel, we present some mathematical notations that are required for both CTMC transient and steady state analyses, which can be used for conducting the reliability and availability analyses of a given system.

5.1 CTMC Definition and Attributes

A stochastic process $\{X_t, t \in T\}$ is a collection of random variables indexed by $t \in T$ [36], where the time t can be continuous or discrete. The values that each random variable can take are called states and the set of these states is called the state space Ω. A Markov process is a stochastic process with the Markov property [35]. If the state space is finite or countably infinite, then the Markov process is called a *Markov Chain* (MC). The Markov property is defined as [2]:

$$\begin{aligned} &Pr\big(X(t) = x |\ X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, ..., X(t_0) = x_0\big) = \\ &Pr\big(X(t) = x |\ X(t_n) = x_n\big) \end{aligned} \tag{3}$$

If the transition can happen at any time, i.e., the time is continuous, then the MC is called a *Continuous Time Markov Chain* (CTMC). In the proposed framework, we are interested in CTMCs as they can capture the dynamic behavior at any instance of time. Once the process is in a certain state, i, the time it spends in this state is exponentially distributed with rate λ_i.

The probabilistic behavior of the CTMC is described by the initial state probability vector $\pi_k(t_0)$ [2], which is defined as $Pr\big(X(t_0 = k\big), k = 0, 1, 2, ...$ and the transition probabilities p_{ij} [2], where

$$p_{ij}(v, t) = Pr\big(X(t) = j \mid X(v) = i\big),\ 0 \leq v \leq t \text{ and } i, j\ =\ 0, 1, 2, ... \tag{4}$$

The CTMC is *time non-homogeneous* if the transition probabilities are functions of time, while it is a *time homogeneous* CTMC if the transition probabilities depend on the time difference $(t - v)$ and not on the actual value of time [2].

$$p_{ij}(t) = Pr\big(X(t+v) = j \mid X(v) = i\big),\ 0 \leq v \tag{5}$$

Each CTMC has an embedded DTMC with a probability transition matrix **P**. The matrix entries are the one step probabilities from state i to state j. This matrix specifies the behavior of the embedded DTMC, however, it does not provide information about the transition rates. The Chapman-Kolmogorov equation [2] provides the probability of the transition from state i to state j in time period from v to t, where the system is taken to an intermediate state k during the time v to u, and from the intermediate state to state j during the time u to t. This equation can be described as [2]:

$$p_{ij}(v,t) = \sum_{k \in \Omega} p_{ik}(v,u) p_{kj}(u,t), \quad 0 \leq v < u < t \tag{6}$$

The state probability or the unconditional probability of being in a certain state can be expressed using the total probability theorem as:

$$\pi_j(t) = \sum_{i \in \Omega} p_{ij}(v,t) \pi_i(v) \tag{7}$$

If we substitute v with 0, then, only the transition probabilities and the initial state probability vector are enough to describe the probabilistic behavior of the CTMC [2]. The state probability vector, π(t), is a vector with an entry for each unconditional state probability. The sum of the entries in the state probability vector at any time is equal to 1, as the MC should be in a certain state.

$$\sum_{j \in \Omega} \pi_j(t) = 1 \tag{8}$$

The infinitesimal matrix (or the generator matrix), **G**, is a core element in the CTMC analysis process. It has all the information about the transition rates. The elements of the matrix **G** are defined by:

$$g_{ij} = \begin{cases} \lambda_{ij} & , i \neq j \\ -\lambda_i & , i = j \end{cases} \tag{9}$$

where λ_{ij} is the transition rate from state i to state j.

5.2 CTMC Dependability Analysis

A CTMC can be used to model the dependability of a certain system. For example, a state machine can start with an initial state, where all system components are working. Then, several states can be used to model the varying failure conditions of system components. Note that it is not necessary that the failure of a

certain component in the system can lead to the failure of the whole system. So the failure of components in the system will cause the transition from one state in the CTMC to another. The transition rate depends on the failure rate of the component that failed. A fail state is used to model the fail state of the system. The CTMC quantitative analysis can be conducted using either the transient analysis or steady-state analysis depending on the dependability metric that we are interested in. These include the instantaneous availability, reliability and steady state availability, as will be described below:

Transient Analysis. The transition probabilities and the transition rates are related using Kolmogrov's forward or backward equations [2]. The backward Kolmogrov's equations are defined as:

$$p'_{ij}(t) = \sum_{k \neq i} \lambda_{ik} p_{ki}(t) - \lambda_i p_{ij}(t) \tag{10}$$

Equation 10 can be rewritten using the matrix form as:

$$\mathbf{P}'(t) = \mathbf{G}\mathbf{P}(t) \tag{11}$$

where $\mathbf{P}(t)$ is the matrix transition probability function. In a similar manner, the generator matrix can be used to find $\boldsymbol{\pi}(t)$ [2]:

$$\boldsymbol{\pi}'(t) = \boldsymbol{\pi}(t)\mathbf{G}(t) \tag{12}$$

Starting from a CTMC that models the failure behavior of a given system, we can find the probability of being available at a certain moment of time, i.e., instantaneous availability or the system reliability using this transient analysis. This is achieved by finding the probability of being in a fail or a working state.

Steady State Analysis. A stationary distribution is the vector of unconditional state probabilities that satisfies the following condition [36]:

$$\pi = \pi \mathbf{P}(t) \tag{13}$$

This means that if the CTMC starts with this initial stationary distribution, the unconditional state probabilities vector at any time will stay the same. The stationary distribution can be found by solving the following set of linear equations with the condition that $\sum_{j \in \Omega} \pi_j(t) = 1$ [36]:

$$\boldsymbol{\pi}\boldsymbol{G} = 0 \tag{14}$$

Using this stationary distribution, we can find the overall probability of system availability by finding the probability of being in a working state. This means that we can find the fraction of time where the system is available during its life cycle, which represents the steady state availability.

6 Current Status and Future Milestones

The final objective of the proposed project is to develop a tool that can be used for the formal dynamic dependability analysis of DFT, DRBD and CTMC models. To achieve this goal, we had to extend the properties of the Lebesgue integral and probability theory in HOL4. This enabled us to verify several probabilistic expressions that are concerned with sequences of random variables. For example, we verified $Pr(X < Y)$, which is required to model the failure behavior of some DFT gates and the DRBD spare construct. Furthermore, we verified several properties for the independence of random variables and sets that are required for verifying the probability of a nested hierarchy of union and intersection of sets. These properties are useful in the analysis of complex DFT and DRBD models. We encountered several challenges during the formalization process including the lack of mathematical proofs in the literature that clearly identify the required steps or theorems that can be utilized to verify the required properties.

We used our formalization to formally model and verify the mathematical foundations of DFTs in the HOL4 theorem prover [14,24,25]. In particular, we modeled the DFT gates and operators and verified several simplification theorems. We illustrated the applicability of the proposed framework by conducting the formal analysis of a cardiac assist system and a drive-by-wire system. Furthermore, we proposed a roadmap [37] to use machine learning techniques to conduct the DFT analysis, which reduces the user interaction with the theorem proving environment. Tables 2 summarizes these accomplished tasks.

We developed a new algebra that allows the formal qualitative and quantitative analyses of DRBDs with spare constructs within a theorem proving environment [15]. We illustrated the usefulness and utilization of the proposed algebra in the formal DRBD based analysis of shuffle-exchange networks, which are used in multiprocessor systems, and a drive-by-wire system, as listed in Table 2.

The remaining tasks of this project can be divided into two main categories: 1) the development and formalization of the mathematical models; and 2) the development of the tool itself. For the first category, we need to extend the DRBD algebra to model the state dependency and load sharing constructs. This requires considering the deactivation process of system components instead of dealing only with the activation and failure. By modeling this behavior, we can consider, in the future, the repairing scenarios that are not currently supported in our DRBD algebra. Furthermore, we have to mathematically model CTMCs in HOL4. For this purpose, we need to extend the properties of conditional density and distribution functions. Modeling exponential distributions and formalizing their properties are also needed as the time spent in each CTMC state is exponentially distributed. Finally, we plan to conduct the dependability analysis using CTMCs including reliability and availability. We plan to utilize this formalization in the dependability analysis of real-world systems, such as solid state drives-RAID systems [38].

Regarding the second category, we plan to develop a parser that creates the HOL dependability models based on a textual or graphical input format. Furthermore, the end-user should be able to specify the type of analysis required

(e.g., reliability or availability) and enter some requirements in the form of expressions. Thus, we have to develop a user friendly graphical user interface (GUI) to obtain these inputs. Then, we intend to invoke some machine learning (ML) algorithms that help in speeding up and automating the verification process. This includes classifying the useful theorems and choosing the proper tactics to be used in the verification of the input model. To achieve this step, we need to create training and test sets that can be used in developing the proper

Table 2. Roadmap

Task	Description	Duration
Accomplished tasks		
1	HOL Formalization of DFT algebra - Formal definitions of DFT gates and temporal operators [14] - Formalization of probabilistic behavior of dynamic gates [14,25]	14 Months
2	DFT applications - Quantitative analysis of CAS [14] - DFT qualitative and quantitative analyses of CAS and DBW [25]	2 Months
3	New DRBD algebra [15]	2 Months
4	HOL formalization of DRBD algebra [15] - Formal definitions of DRBD structures and spare construct - Formalization of reliability expressions	9 Months
5	DRBD applications - Formal reliability analysis of DBW and SEN [15]	1 Month
Future plan		
6	More DRBD dynamic constructs - Formalization of state dependencies construct - Formalization of load sharing construct	6 Months
7	Formalization of CTMCs - Homogeneous - Non-homogeneous	9 Months
8	Formal CTMC analysis - Transient analysis - Steady state analysis	6 Months
9	Applications (Reliability of SSD RAID)	2 Months
10	Using machine learning to automate the analysis - Create training and test sets - Develop ML models	6 Months
11	Tool Development - Develop a GUI - Develop a parser - Program the core of the tool	3 Months
	Total time	60 Months

ML models. Finally, we have to program the core of the tool that connects the pieces of the framework together to enable the automatic dynamic dependability analysis. As the development of this tool is an incremental process, which can be improved with time, we plan to conduct some tutorials for end-users that are not familiar with HOL to train them and consider their feedback. This step is also important for verification and reliability engineers that are interested in enriching the underlying theories of the proposed framework. This helps in the sustainability of the proposed framework by engaging many users with different goals and backgrounds in the development of the framework and its tool. A summary of this roadmap is provided in Table 2.

7 Conclusions

In this paper, we proposed a comprehensive framework to conduct the formal dynamic dependability analysis using HOL theorem proving. We provided the details of the mathematical foundations of each part of the proposed framework. The main contributions of this work is the development of the proposed framework in the HOL4 theorem prover that includes the formalization of DFTs and CTMCs besides the development of the DRBD algebra. These formalized models allow the dependability analysis of many real-world system that exhibit dynamic behavior. We described the future milestones to complete the proposed project including the final tool that enables the (semi) automation of the analysis.

References

1. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–33 (2004)
2. Trivedi, K.S.: Probability and Statistics with Reliability, Queuing and Computer Science Applications. Wiley, Hoboken (2002)
3. Stamatelatos, M., Vesely, W., Dugan, J., Fragola, J., Minarick, J., Railsback, J.: Fault Tree Handbook with Aerospace Applications. NASA Office of Safety and Mission Assurance (2002)
4. Distefano, S., Xing, L.: A new approach to modeling the system reliability: dynamic reliability block diagrams. In: Reliability and Maintainability Symposium, pp. 189–195. IEEE (2006)
5. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press, Cambridge (2008)
6. Gordon, M.J., Melham, T.F.: Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic. Cambridge University Press, Cambridge (1993)
7. Dehnert, C., Junges, S., Katoen, J.-P., Volk, M.: A storm is coming: a modern probabilistic model checker. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10427, pp. 592–600. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63390-9_31
8. Ghadhab, M., Junges, S., Katoen, J.-P., Kuntz, M., Volk, M.: Model-based safety analysis for vehicle guidance systems. In: Tonetta, S., Schoitsch, E., Bitsch, F. (eds.) SAFECOMP 2017. LNCS, vol. 10488, pp. 3–19. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66266-4_1

9. Elderhalli, Y., Volk, M., Hasan, O., Katoen, J.-P., Tahar, S.: Formal verification of rewriting rules for dynamic fault trees. In: Ölveczky, P.C., Salaün, G. (eds.) SEFM 2019. LNCS, vol. 11724, pp. 513–531. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30446-1_27
10. Kwiatkowska, M., Norman, G., Parker, D.: Quantitative analysis with the probabilistic model checker PRISM. Electron. Notes Theor. Comput. Sci. **153**(2), 5–31 (2006)
11. Ahmed, W., Hasan, O.: Formalization of fault trees in higher-order logic: a deep embedding approach. In: Fränzle, M., Kapur, D., Zhan, N. (eds.) SETTA 2016. LNCS, vol. 9984, pp. 264–279. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47677-3_17
12. Ahmed, W., Hasan, O., Tahar, S.: Formalization of reliability block diagrams in higher-order logic. J. Appl. Logic **18**, 19–41 (2016)
13. HOL4 (2020). https://hol-theorem-prover.org/
14. Elderhalli, Y., Ahmad, W., Hasan, O., Tahar, S.: Probabilistic analysis of dynamic fault trees using HOL theorem proving. J. Appl. Logics **6**, 467–509 (2019)
15. Elderhalli, Y., Hasan, O., Tahar, S.: A formally verified algebraic approach for dynamic reliability block diagrams. In: Ait-Ameur, Y., Qin, S. (eds.) ICFEM 2019. LNCS, vol. 11852, pp. 253–269. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32409-4_16
16. Hölzl, J.: Markov processes in Isabelle/HOL. In: ACM SIGPLAN Conference on Certified Programs and Proofs, pp. 100–111 (2017)
17. Isabelle (2020). https://isabelle.in.tum.de/
18. Ahmed, W., Hasan, O.: Towards formal fault tree analysis using theorem proving. In: Kerber, M., Carette, J., Kaliszyk, C., Rabe, F., Sorge, V. (eds.) CICM 2015. LNCS (LNAI), vol. 9150, pp. 39–54. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20615-8_3
19. Mhamdi, T., Hasan, O., Tahar, S.: On the formalization of the lebesgue integration theory in HOL. In: Kaufmann, M., Paulson, L.C. (eds.) ITP 2010. LNCS, vol. 6172, pp. 387–402. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14052-5_27
20. Mhamdi, T., Hasan, O., Tahar, S.: Formalization of entropy measures in HOL. In: van Eekelen, M., Geuvers, H., Schmaltz, J., Wiedijk, F. (eds.) ITP 2011. LNCS, vol. 6898, pp. 233–248. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22863-6_18
21. Ruijters, E., Stoelinga, M.: Fault tree analysis: a survey of the state-of-the-art in modeling. Anal. Tools Comput. Sci. Rev. **15–16**, 29–62 (2015)
22. Merle, G.: Algebraic modelling of dynamic fault trees, contribution to qualitative and quantitative analysis. Ph.D. thesis, ENS, France (2010)
23. Sullivan, K.J., Dugan, J.B., Coppit, D.: The galileo fault tree analysis tool. In: IEEE Symposium on Fault-Tolerant Computing, pp. 232–235 (1999)
24. Elderhalli, Y., Hasan, O., Ahmad, W., Tahar, S.: Formal dynamic fault trees analysis using an integration of theorem proving and model checking. In: Dutle, A., Muñoz, C., Narkawicz, A. (eds.) NFM 2018. LNCS, vol. 10811, pp. 139–156. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77935-5_10
25. Elderhalli, Y., Hasan, O., Tahar, S.: A methodology for the formal verification of dynamic fault trees using HOL theorem proving. IEEE Access **7**, 136176–136192 (2019)
26. Boudali, H., Crouzen, P., Stoelinga, M.: A rigorous, compositional, and extensible framework for dynamic fault tree analysis. IEEE Trans. Dependable Secure Comput. **7**, 128–143 (2010)

27. Altby, A., Majdandzic, D.: Design and implementation of a fault-tolerant drive-by-wire system. Master's thesis, Chalmers University of Technology, Sweden (2014)
28. Distefano, S., Puliafito, A.: Dynamic reliability block diagrams vs dynamic fault trees. In: Reliability and Maintainability Symposium, pp. 71–76. IEEE (2007)
29. BlockSim (2020). https://www.reliasoft.com/products/reliability-analysis/blocksim
30. Distefano, S.: System dependability and performances: techniques, methodologies and tools. Ph.D. thesis, University of Messina, Italy (2005)
31. Xu, H., Xing, L.: Formal semantics and verification of dynamic reliability block diagrams for system reliability modeling. In: International Conference on Software Engineering and Applications, pp. 155–162 (2007)
32. Smith, G.: The Object-Z Specification Language, vol. 1. Springer, Boston (2012). https://doi.org/10.1007/978-1-4615-5265-9
33. Xu, H., Xing, L., Robidoux, R.: Drbd: dynamic reliability block diagrams for system reliability modelling. Int. J. Comput. Appl. **31**(2), 132–141 (2009)
34. Hasan, O., Ahmed, W., Tahar, S., Hamdi, M.S.: Reliability block diagrams based analysis: a survey. In: International Conference of Numerical Analysis and Applied Mathematics, vol. 1648, p. 850129.1-4. AIP (2015)
35. Liu, L., Hasan, O., Tahar, S.: Formal reasoning about finite-state discrete-time Markov chains in HOL. J. Comput. Sci. Technol. **28**(2), 217–231 (2013)
36. Grimmett, G., Stirzaker, D., et al.: Probability and Random Processes. Oxford University Press, Oxford (2001)
37. Elderhalli, Y., Hasan, O., Tahar, S.: Using machine learning to minimize user intervention in theorem proving based dynamic fault tree analysis. In: Conference on Artificial Intelligence and Theorem Proving, pp. 36–37 (2019)
38. Li, Y., Lee, P.P.C., Lui, J.C.S.: Stochastic analysis on RAID reliability for solid-state drives. In: IEEE International Symposium on Reliable Distributed Systems, pp. 71–80 (2013)