Date of publication SEP-30, 2024, date of current version SEP-10, 2024
www.computingonline.net/ computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.23.3.XXXX

Application of Binary Decision Diagrams in Time-Dependent Reliability Analysis

MICHAL MRENA[1], MIROSLAV KVASSAY[1]
[1] Faculty of Management Science and Informatics, University of Zilina, Zilina, Slovakia (e-mail: {michal.mrena,miroslav.kvassay}@fri.uniza.sk)
Corresponding author: Michal Mrena (e-mail: michal.mrena@fri.uniza.sk).

This work was supported by the Ministry of Education, Science, Research and Sport of the Slovak Republic under the grant VEGA 1/0858/21.

ABSTRACT Binary Decision Diagrams (BDDs) are often used in specific types of reliability analysis known as topological (or structure) analysis and time-independent analysis. The previous focuses only on the analysis of system topology, which is defined by structure function. The latter takes into account the structure function together with the time-independent reliabilities of components that the system is composed of. However, the most interesting type of reliability analysis is time-dependent analysis in which reliabilities of the components are time-dependent functions. In this paper, we first present the development of a mathematical model of a non-repairable system composed of independent non-repairable components and explain the properties of this model from the point of view of time-dependent, time-independent, and topological reliability analysis. In the second part of the paper, we present and experimentally compare two methods for time-dependent reliability analysis of the considered mathematical model. The first method is based on the direct application of BDDs and we label it as a basic approach. The second, symbolic approach, combines BDDs with expression trees. The experimental comparison implemented using open-source C++ libraries TeDDy and GiNaC shows that the first method based on the basic approach is much faster than the second method using expression trees.

KEYWORDS binary decision diagram; GiNaC; probabilistic decision diagram; structure function; symbolic expression; TeDDy; time-dependent reliability.

I. INTRODUCTION

A Binary Decision Diagram (BDD) is a graph structure that can represent any Boolean function [1], [2]. Due to its ability to efficiently store information about a Boolean function and to be simply processed on a computer, it has found applications in almost all areas dealing with Boolean functions. Nowadays, BDDs are used in the analysis of logic circuits [3], synthesis of optical circuits [4], game theory [1], mathematical programming [5], and many other areas. Along with all these areas they have found important applications in reliability analysis [6], [7]. In this case, they allow representing functions defining the structure of large systems composed of many components.

Although BDDs have found use in many areas, they are primarily used in solving time-independent problems. For example, in reliability analysis, they have been used in the evaluation of system reliability from the topological point of view or from the point of view of time-independent probabilistic analysis [7], but according to our knowledge, they have been used in time-dependent analysis only marginally. Furthermore, many papers dealing with applications of BDDs in reliability analysis are based on custom software solutions of researchers, which have not been made publicly available and were primarily used for solving a specific problem considered in the papers of researchers. Because of that, we have decided to develop a software solution that will be publicly available. This solution has a character of C++ software library named TeDDy (Templated Decision Diagram library) and is available under link https://github.com/MichalMrena/DecisionDiagrams.

Unlike well-known BDD packages, such as CUDD [8] or BuDDy [9], which are commonly used in logic design, TeDDy has been developed primarily for tasks related to reliability analysis of large systems. It allows us to

perform topological analysis as well as probabilistic time-independent analysis [10]. However, many tasks of reliability analysis are based on time-dependent analysis [11]–[14]. Most of these tasks are based on the direct application of state transition diagrams and the theory of stochastic processes [15]. If we want to use our library to solve such kinds of problems, we have to consider the incorporation of time processing into it. This can be done in several ways. Thus, in comparison to existing research, the main contribution of this paper lies in performing time-dependent probabilistic analysis using BDDs.

In this paper, we compare two approaches to how BDDs can be utilized in time-dependent reliability analysis. The first, simpler approach is based on the idea that elements (nodes) of the BDD will represent time-dependent functions defining reliabilities of the components of the system whose structure is expressed by the BDD. The second approach is based on the idea that the BDD defining the structure of the system is transformed into symbolic expression, which is stored in the form of an abstract syntax tree with one independent variable – time. Using a state-of-the-art library for manipulation of symbolic expressions GiNaC [16] we show that the first approach is quicker than the second one.

II. RELIABILITY ANALYSIS OF NON-REPAIRABLE SYSTEM WITH INDEPENDENT NON-REPAIRABLE COMPONENTS

Reliability is an important characteristic of most real-world systems. It is crucial in the development of integrated circuits based on field programmable gate arrays [17], in the deployment of unmanned aerial vehicle systems [18], [19], in the design of vehicular ad-hoc networks [13], in creation of smart buildings and homes [12] as well as in many other areas summarized in a nice way in [20]. Based on available data about the system and its components, we can create various mathematical models of the system that allow us to consider various properties of the system [21]–[23].

According to [21], the reliability of a system is defined as the ability of the system to perform a required function under given environmental and operational conditions for a given period of time. From the mathematical point of view, reliability R at time t agrees with the probability that the system does not fail by time t assuming that it works at time $t = 0$. If we introduce a random variable T that defines the time to failure of the system, then reliability at time t can be expressed as the following function:

$$R(t) = \Pr\{T > t\};\ R(0) = 1. \tag{1}$$

The complementary function is known as system unreliability. Using random variable T, it is defined at time t as the probability that a failure of the system occurs by time t given that the system works at time $t = 0$, i.e.:

$$F(t) = \Pr\{T \leq t\} = 1 - R(t);\ F(0) = 0. \tag{2}$$

The previous two definitions imply that the system can be in one of two states from the reliability point of view. These states are working and failed. Thus, for a given time point t, we can define a random variable Z_t that takes value 1 if the system is working and value 0 if it is failed at time point t. With respect to random variable T, random variable Z_t can be defined as follows:

$$Z_t = \mathrm{I}(T > t) = \begin{cases} 1, & \text{if } T > t \\ 0, & \text{otherwise} \end{cases}, \tag{3}$$

where $\mathrm{I}(.)$ is indication function that takes value 1 if the expression in the parentheses is true and value 0 otherwise. Since the random variable Z_t has only two possible values, it has a Bernoulli distribution.

If we consider uncountable collection of random variables Z_t for all possible values of t, i.e., for $t \in : [0, \infty)$, then we obtain continuous stochastic process $\{Z_t,\ 0 \leq t < \infty\}$. This process defines how the state of the system changes as time flows and can be described by system state function $Z(t)$:

$$\begin{aligned} Z(t) &= \{Z_t,\ 0 \leq t < \infty\} \\ &= \mathrm{I}(T > t) \colon [0, \infty) \to \{0, 1\}. \end{aligned} \tag{4}$$

It is worth noting that we use two similar symbols that are $Z(t)$ and Z_t. The difference between them lies in the fact that t in $Z(t)$ is an independent variable whose value is not known a priori, while t in Z_t denotes a time point that is known a priori. A notation in which a priori information is placed in the subscript of the considered mathematical object will also be used in the rest of the paper.

Using the system state function, event $\{T > t\}$ agrees with event $\{\forall \tau \in [0, t] : Z(\tau) = 1\}$. Therefore, the definition (1) of reliability can be expressed in the following form:

$$\begin{aligned} R(t) &= \Pr\{\{\forall \tau \in [0, t] : Z(\tau) = 1\} | Z(0) = 1\} \\ &= \frac{\Pr\{\forall \tau \in [0, t] : Z(\tau) = 1\}}{\Pr\{Z(0) = 1\}} \\ &= \frac{\Pr\{\forall \tau \in [0, t] : Z(\tau) = 1\}}{R(0)}. \end{aligned} \tag{5}$$

Since definition (1) implies that $R(0) = 1$, we can write:

$$R(t) = \Pr\{\forall \tau \in [0, t] : Z(\tau) = 1\}. \tag{6}$$

Using the conditional probability, we can rewrite this formula as the following multiplication:

$$\begin{aligned} R(t) = &\Pr\{\{\forall \tau \in [0, t) : Z(\tau) = 1\} | Z(t) = 1\} \\ &. \Pr\{Z(t) = 1\}. \end{aligned} \tag{7}$$

In what follows, we will assume that the system is non-repairable. Using this assumption event $\{Z(t) = 1\}$ implies that:

$$\Pr\{\{\forall \tau \in [0, t) : Z(\tau) = 1\} | Z(t) = 1\} = 1; \tag{8}$$

therefore, we can write:

$$R(t) = \Pr\{Z(t) = 1\}. \tag{9}$$

Next, let us assume that the system is composed of n components that are independent. Let us define a random variable T_i that specifies the time to failure of the i-th component, for $i = 1, 2, \ldots, n$. Using this random variable, we can define the reliability of the i-th component at time t as follows:

$$p_i(t) = \Pr\{T_i > t\} \tag{10}$$

and its unreliability in the following manner:

$$q_i(t) = \Pr\{T_i \leq t\} = 1 - p_i(t). \tag{11}$$

As in the case of the whole system, we can introduce the state function of the i-th component $Z_i(t)$:

$$\begin{aligned} Z_i(t) &= \{Z_{i,t},\ 0 \leq t < \infty\} \\ &= \mathrm{I}(T_i > t)\colon [0, \infty) \to \{0, 1\}, \\ &\quad \text{for } i = 1, 2, \ldots, n, \end{aligned} \tag{12}$$

which can be viewed as uncountable collection of random variables $Z_{i,t}$ modeling state of the i-th component at time point t. As in the case of random variable Z_t, random variables $Z_{i,t}$ have Bernoulli distribution.

If we assume that the components of the system are non-repairable, then the same procedure as above can be used to show that the reliability and unreliability functions of the i-th component have the following forms:

$$\begin{aligned} p_i(t) &= \Pr\{Z_i(t) = 1\}\colon [0, \infty) \to [0, 1], \\ q_i(t) &= 1 - p_i(t)\colon [0, \infty) \to [0, 1]. \end{aligned} \tag{13}$$

Using the same notation as in the case of stochastic process (12), these time-dependent functions can also be viewed as uncountable collections of time-independent probabilities (probabilities at a priori defined time points). These probabilities are labeled as $p_{i,t}$ and $q_{i,t} = 1 - p_i, t$ and they are together referred to as component state probabilities. Thus, we can write:

$$\begin{aligned} p_i(t) &= \{p_{i,t},\ 0 \leq t < \infty\}, \\ q_i(t) &= \{q_{i,t},\ 0 \leq t < \infty\} \\ &= \{1 - p_{i,t},\ 0 \leq t < \infty\}. \end{aligned} \tag{14}$$

Clearly, the state of a system composed of n components depends on the states of its components. For this purpose, it is useful to introduce one more function, which is structure function. The structure function is a mapping of the following form [21]:

$$\phi(\boldsymbol{x})\colon \{0, 1\}^n \to \{0, 1\}, \tag{15}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is the state vector, variable x_i for $i = 1, 2, \ldots, n$ defines state of the i-th component, and n is the number of components. Since we assume that states of the components are modeled by time-dependent state functions $Z_i(t)$, we can substitute state vector $\boldsymbol{x}$ by a vector of component state functions $\boldsymbol{Z}(t) = (Z_1(t), Z_2(t), \ldots, Z_n(t)$. Based on this substitution, we can write:

$$Z(t) = \phi(\boldsymbol{Z}(t)) = \phi(Z_1(t), Z_2(t), \ldots, Z_n(t)). \tag{16}$$

System state function (16) defined by structure function (15) together with component state probabilities (13) represents the basic mathematical model used in reliability analysis. It is a model of a non-repairable system composed of independent non-repairable components. Using this model, system reliability is a function depending on structure function $\phi(\boldsymbol{x})$, component reliabilities $\boldsymbol{p}(t) = (p_1(t), p_2(t), \ldots, p_n(t))$ and time t, i.e.:

$$R(\phi, \boldsymbol{p}, t) = \Pr\{\phi(\boldsymbol{Z}(t)) = 1\}. \tag{17}$$

The independence of system components implies that this function can be rewritten in the following form:

$$R(\phi, \boldsymbol{p}, t) = \sum_{\boldsymbol{x} \in \{0,1\}^n} \phi(\boldsymbol{x}) \prod_{i=1}^{n} \Pr\{Z_i(t) = x_i\}. \tag{18}$$

Function (18) can be analyzed in three basic ways that are time-dependent probabilistic analysis, time-independent probabilistic analysis, and topological analysis.

A. TIME-DEPENDENT PROBABILISTIC ANALYSIS

In time-dependent probabilistic analysis, we assume that structure function (15) and component reliabilities (13) are a priori given, and we study how time influences the reliability of the system. If we use notation such that a priori information is placed in the subscript of the considered measure, we can write:

$$R(\phi, \boldsymbol{p}, t) = R_{\phi, \boldsymbol{p}}(t), \tag{19}$$

which indicates that reliability is a function of time. In this case, reliability can be calculated using the formula (18).

B. TIME-INDEPENDENT PROBABILISTIC ANALYSIS

Sometimes it is useful to study how the reliability of the system depends on state probabilities of its components. In this case, we assume that component states are not modeled by time-dependent state functions $Z_i(t)$ but rather by random variables $Z_{i,t}$. According to (12) these random variables have Bernoulli distribution with the following two probabilities:

$$\begin{aligned} p_{i,t} &= \Pr\{Z_{i,t} = 1\}, \\ q_{i,t} &= 1 - p_{i,t}. \end{aligned} \tag{20}$$

This implies that system reliability is a function depending only on state probabilities of system components, i.e.:

$$\begin{aligned} R(\phi, \boldsymbol{p}, t) &= R_{\phi,t}(\boldsymbol{p}_t) \\ &= \sum_{\boldsymbol{x} \in \{0,1\}^n} \phi(\boldsymbol{x}) \prod_{i=1}^{n} \Pr\{Z_{i,t} = x_i\}, \end{aligned} \tag{21}$$

where $\boldsymbol{p}_t = (p_{1,t}, p_{2,t}, \ldots, p_{n,t})$.

Time-independent probabilistic analysis can also be performed without a priori knowledge of the time in which we want to perform such a kind of reliability analysis. In this case, subscript t can be omitted, random variables $Z_{i,t}$ have

a form of Z_i, and component state probabilities are labeled as p_i and q_i. The consequence of this is that the reliability function becomes a time-invariant function:

$$R(\phi, \boldsymbol{p}, t) = R_\phi(\boldsymbol{p}) = \sum_{\boldsymbol{x}\in\{0,1\}^n} \phi(\boldsymbol{x}) \prod_{i=1}^{n} \Pr\{Z_i = x_i\}, \tag{22}$$

where $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$.

This approach is suitable for analyzing how various values of component state probabilities influence the reliability of the whole system, which is a typical task of a special part of reliability analysis known as importance analysis [7], [24].

C. TOPOLOGICAL ANALYSIS

Finally, we can study system reliability from the topological point of view. This means that we focus on the layout of the system and connections between system components. In this case, we assume that the component reliabilities and time are fixed, and we analyze how system reliability changes depending on various structure functions. Therefore, in this case, the reliability of the system can be defined as a function depending on the structure function:

$$R(\phi, \boldsymbol{p}, t) = R_{\boldsymbol{p},t}(\phi) \tag{23}$$

and can be computed using formula (21).

A specific type of topological analysis is structure analysis, in which we assume that both states of each component occur with the same probability which is independent of time, i.e.:

$$\Pr\{Z_{i,t} = 1\} = \Pr\{Z_{i,t} = 0\} = \frac{1}{2}, \quad \text{for } i = 1, 2, \ldots, n \text{ and } t \in [0, \infty). \tag{24}$$

In this specific case, topological reliability (23) can be transformed using formula (21) in the following way:

$$\begin{aligned} R_{\boldsymbol{p},t}(\phi) &= R_{\frac{\mathbf{1}}{\mathbf{2}}}(\phi) \\ &= \sum_{\boldsymbol{x}\in\{0,1\}^n} \phi(\boldsymbol{x}) \prod_{i=1}^{n} \frac{1}{2} \\ &= \frac{1}{2^n} \sum_{\boldsymbol{x}\in\{0,1\}^n} \phi(\boldsymbol{x}). \end{aligned} \tag{25}$$

where vector $\frac{\mathbf{1}}{\mathbf{2}} = (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ used in the subscript means that each of n system components is reliable with the probability 0.5.

Please note that the structure function can be viewed as a Boolean function of n variables. In this case, the last row in the previous formula agrees with the relative number of points (state vectors with respect to reliability terminology) at which the structure function is true (takes value 1 on the output). With respect to Boolean functions, this characteristic agrees with the Truth Density (TD) of Boolean function, therefore, we can write:

$$R_{\frac{\mathbf{1}}{\mathbf{2}}}(\phi) = \mathrm{TD}(\phi(\boldsymbol{x})). \tag{26}$$

Clearly, this measure does not depend on the time point at which it is evaluated. In some works, e.g., [25], it is known as state frequency of system state 1 and is marked using the following notation:

$$\mathrm{Fr}^{=1}(\phi) = R_{\frac{\mathbf{1}}{\mathbf{2}}}(\phi). \tag{27}$$

An advantage of the structure analysis is that it requires only the structure function as its input. Hence, for example, it allows us to analyze systems that do not yet exist, we only need to know their topology. Because of that, this kind of analysis is used in the phase of the design of the system to find the topology that will be most reliable without respect to the reliabilities of the system components.

III. DECISION DIAGRAMS

As the previous section implies, one of the key elements of the model of a system is structure function. Evaluation of system reliability thus requires a suitable representation of this function. Choice of the representation is essential if we need to analyze a system with numerous components since simple approaches – like truth tables – fail for functions with a higher number of variables.

Decision diagrams are often the structure of choice when one needs to represent larger discrete functions. Even though the size of the diagram is still exponential in the number of variables in the worst case [26], they can often represent a function much more efficiently [2]. The literature describes various types of decision diagrams. In general, a decision diagram is a directed acyclic graph structure containing two types of nodes. *Terminal* nodes contain values of the function, and *internal* nodes represent variables with several outgoing edges, which is given by the size of the domain of the variable. Consequently, each internal node represents a decision based on the value of the variable hence the name *decision* diagram.

Reduced Ordered Multi-valued Decision Diagram (ROMDD or shortly just MDD) [27] is a common general type of decision diagram, which represents an integer function of the following form:

$$\begin{aligned} f(\boldsymbol{x})\colon \{0, 1, \ldots, m_1 - 1\} \times \{0, 1, \ldots, m_2 - 1\} \\ \times \cdots \times \{0, 1, \ldots, m_n - 1\} \to \{0, 1, \ldots, m - 1\}, \end{aligned} \tag{28}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, n denotes the number of variables, $m_i \in \mathbb{N} - \{1\}$ denotes the size of the domain of the variable x_i for $i = 1, 2, \ldots, n$, $\mathbb{N}$ denotes the set of natural numbers, and $m \in \mathbb{N} - \{1\}$ denotes the size of the codomain of the function. In Figure 1, we can see an example of an MDD representing an integer function of three variables. In the case when $m_i = m_j = m$ for $i, j = 1, 2, \ldots, n$ the definition (28) agrees with the definition of Multiple-valued Logic function [28]. Furthermore, if $m = 2$ the definition (28) agrees with the definition of Boolean function [1]. In such a case, the decision diagram is known as a Binary Decision Diagram (BDD) [2].

Historically, BDD is a decision diagram that precedes MDD – we can consider MDD as a generalization of BDDs.

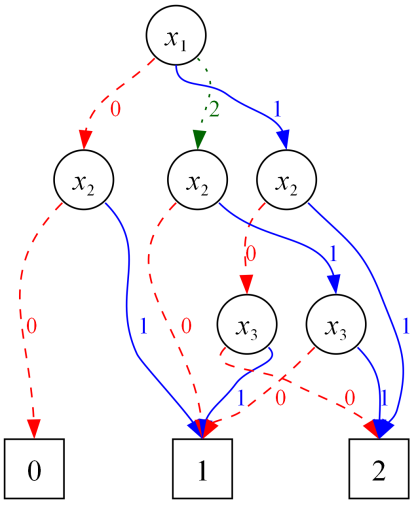

Figure 1. MDD representing an integer function where $m_1 = 3, m_2 = 2, m_3 = 2$.

Since BDDs can efficiently represent Boolean functions they have found their application in numerous areas dealing with applications of logic functions [1], [28], such as design and verification of logic circuits or artificial intelligence. Moreover, since the definition of the structure function (15) agrees with the definition of a Boolean function, which BDD can efficiently represent, BDDs have also found strong application in reliability analysis [6], [7], [10]. In the rest of the paper, we will focus on the application of BDDs in probabilistic analysis.

BDD has at most two terminal nodes representing values 0 and 1, which can be interpreted as *false* and *true* respectively. Each internal node of BDD has two outgoing edges representing two possible values of the variable, again, the values 0 and 1. In the paper we use the notation VALUE(T) to denote value stored in terminal node T, INDEX(A) to denote index of the variable associated with internal node A, and LeftSon(A) and RIGHTSON(A) to denote left and right son of an internal node A respectively.

IV. PROBABILISTIC ANALYSIS BY BDDS

Based on the construction and internal structure of BDD it can be shown that it is an orthogonal representation of a Boolean function. This makes it a convenient structure for performing probabilistic calculations. The literature offers two principal approaches to the calculation of output probabilities known as *bottom-up* and *top-down* [28] algorithms. Both approaches can calculate the so-called Node Traversing Probability (NTP) [29] of the terminal node (typically the node representing 1) using a traversal of the BDD, in which they associate intermediate results with the nodes. They differ in the order in which they traverse the diagram and in the node from which they read the resulting NTP. In the rest of the paper, we use the *bottom-up* approach, which was used by other researchers [30] in a probabilistic evaluation.

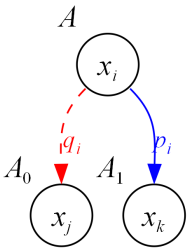

Figure 2. Internal node of a probabilistic BDD depicting the relation between the graph structure and component state probabilities.

However, the techniques we describe further in the paper are also applicable to the *top-down* approach and possibly to other approaches as well.

A. TIME-INDEPENDENT ANALYSIS

1) Bottom-up algorithm

Evaluation of system reliability requires us to evaluate the probability that the structure function (15) evaluates to 1 given component state probabilities (20) or (13). This agrees with the NTP of the terminal node representing value 1. To evaluate this NTP, the *bottom-up* algorithm assigns probability Prob(A) to each internal node A associated with variable x_i using the following recursive formula:

$$\text{Prob}(A) = q_i * \text{Prob}(A_0) + p_i * \text{Prob}(A_1), \qquad (29)$$

where A_0 and A_1 are nodes where the edges representing values 0 and 1 lead to and q_i and p_i are component state probabilities. The recursion terminates if A_k (for $k = 0, 1$) is a terminal node, in which case it returns the value k i.e., the value stored in the node. Finally, to obtain the system reliability, we evaluate the probability Prob($root$) of the root node.

Considering the relation (29), we may notice that it is closely tied with the structure of the internal node as we can see in Figure 2. Let us notice that in the figure, the component state probabilities q_i and p_i are placed on the edges. We refer to such a BDD as a probabilistic BDD.

2) Bottom-up algorithm implementation

Implementation of the *bottom-up* algorithm is relatively straightforward. It is a simple recursive algorithm that follows the relation (29). However, there are certain complications caused by node sharing that need to be addressed. Direct computation of (29) in each internal node would cause the recursion to visit shared nodes multiple times – recomputing the entire subgraph each time. This would make the algorithm considerably inefficient. Therefore, the implementation of the algorithm visits each node exactly once and it puts the value computed at each node into a cache (we refer to it as *memo*).

In Algorithm 1, we can see the pseudocode of the algorithm. Lines 2–4 implement the terminating case of

Algorithm 1 Bottom-up algorithm for the calculation of Node Traversing Probabilities in BDD.

```
1:  procedure BottomUpNTP(node, p)
2:      if IsTerminal(node) then
3:          return Value(node)
4:      end if
5:      if Contains(memo) then
6:          return Lookup(memo, node)
7:      end if
8:      i ← Index(node)
9:      prob_0 ← BottomUpNTP(LeftSon(node), p)
10:     prob_1 ← BottomUpNTP(RightSon(node), p)
11:     result ← (1 − p_i) * prob_0 + p_i * prob_1
12:     Put(memo, node, result)
13:     return result
14: end procedure
```

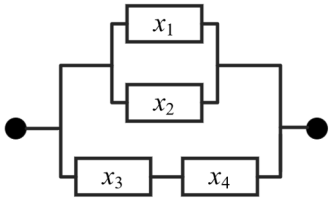

Figure 3. Reliability Block Diagram depicting simple series-parallel system made of four components.

the relation (29) and lines 9–11 implement the case for internal nodes. Lines 5–7 and line 12 marked with blue color implement the caching mechanism, which ensures that each node is visited exactly once. The actual data structure behind the variable *memo* is implementation-defined. Typically, it is a hash table that maps a pointer to a real number. In section V, we discuss another possibility which is to store the cached value directly in the node.

Logical view of probabilistic BDD places the component state probabilities q_i and p_i in the edges. However, implementing the BDD this way would not be reasonable since the same information would be stored duplicitly in multiple edges. In the case of time-independent analysis, q_i and p_i are real numbers – typically represented as floating point numbers. Hence, the input of the algorithm is usually a single vector $\boldsymbol{p}$ of floating point numbers representing probabilities p_i for $i = 1, 2, \ldots n$. There is no need to store q_i since they can be easily computed using the formula (20). Consequently, implementation of the time-independent analysis is relatively straightforward.

B. TIME-DEPENDENT ANALYSIS

Evaluation of the NTP gets complicated when we work with time-dependent component state probabilities. In this case, the probabilities are expressions containing variable t representing time. The expressions typically represent cumulative distribution functions of some probability distribution such as exponential or Weibull [21].

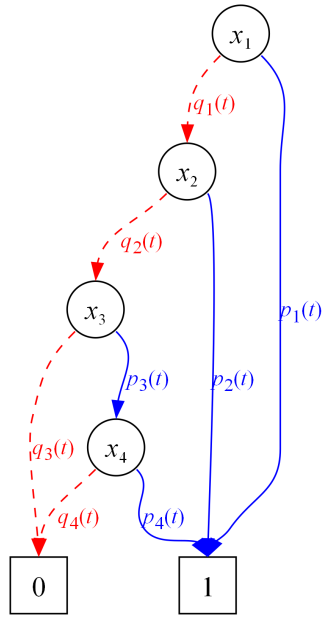

Figure 4. Probabilistic BDD representing structure function of the series-parallel system depicted in Figure 3.

We will illustrate the problem and, subsequently, our solution using a simple example. Let us consider a simple storage system analyzed in [31]. The system consists of two units connected in parallel. Each unit has two hard drives configured as RAID 1 and RAID 0 respectively. In Figure 3 we can see a reliability block diagram describing the system, which has the following structure function:

$$\phi(\boldsymbol{x}) = (x_1 \vee x_2) \vee x_3 x_4. \tag{30}$$

Using the function and the inclusion-exclusion principle we can calculate the system reliability using the following formula:

$$\begin{aligned} R_{\phi,\boldsymbol{p}}(t) = {} & p_1(t) + p_2(t) + p_3(t)p_4(t) \\ & - p_1(t)p_2(t) \\ & - p_1(t)p_3(t)p_4(t) \\ & - p_2(t)p_3(t)p_4(t) \\ & + p_1(t)p_2(t)p_3(t)p_4(t). \end{aligned} \tag{31}$$

In Figure 4, we can see a probabilistic BDD representing the structure function. By using the *bottom-up* algorithm and relation (29) we obtain the following formula:

$$R_{\phi,\boldsymbol{p}}(t) = q_1(t)(p_2(t) + q_2(t)p_3(t)p_4(t)) + p_1(t), \tag{32}$$

which after substituting $1 - p_i(t)$ for each $q_i(t)$ agrees with formula (31). Let us assume the same exponential distributions of component reliabilities as authors in [31] – we can see the distributions in Table 1. If we substitute

Table 1. Storage system component reliability distributions.

Component	Component reliability $p_i(t)$
1	$\exp(1/25,359) * t$
2	$\exp(1/6,246) * t$
3	$\exp(1/4,764) * t$
4	$\exp(1/44,360) * t$

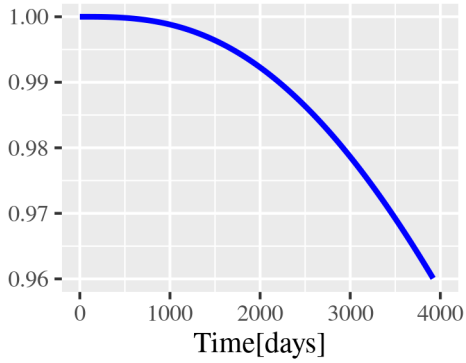

Figure 5. Reliability function of the series-parallel system depicted in Figure 3.

the distributions into expression (31) we can plot the system reliability function which we can see in Figure 5.

Now let us consider the task of evaluating system reliability at multiple time points using BDD. The input vector $\boldsymbol{p}$ is no longer a vector of simple floating-point numbers, but it is a vector of cumulative distribution functions describing the evolution of component reliabilities in time. We identified two approaches to the problem.

1) Basic approach

The first approach, which we call the *basic* approach, is straightforward. The idea is to, first, evaluate each element of $\boldsymbol{p}$ at time point t, transforming it into $\boldsymbol{p}_t$ – a simple vector of floating-point numbers representing component probabilities at time point t – and proceed with the calculation of system reliability using the standard time-independent algorithm such as the *bottom-up* algorithm described in the previous subsection. This approach requires no modification of the standard algorithm and therefore can be used with existing tools. However, it requires repeated evaluation of the *bottom-up* algorithm for each time point t.

The authors in [32] utilize this approach in the analysis of distributed generation power systems. In Algorithm 2 we can see a pseudocode that shows how to use this approach to evaluate system reliability at multiple time points. One can recognize that this algorithm is based on the application of the formula (21), which is used to compute system reliability for various component state probabilities defined by a priori known time points.

2) Symbolic approach

The second approach utilizes symbolic expressions – hence we refer to it as *symbolic* approach. Various computer algebra systems such as Matlab, GNU Octave, or wxMaxima allow manipulation, evaluation, and analysis of expressions represented by trees. Figure 6 shows an example of such a tree. Thus, the main idea of the *symbolic* approach is to perform the calculation on expressions rather than probabilities evaluated in time t. Therefore, the input vector $\boldsymbol{p}$ contains expressions representing component state probabilities $p_i(t)$ depending on a single variable t representing time.

Algorithm 2 Basic approach to the evaluation of system reliability in multiple time points.

```
1: function EvaluateBasic(bdd, timePoints, p)
2:     for ∀ t ∈ timePoints do
3:         p_t ←EvaluateDistributions(p, t)
4:         R ←BottomUp(bdd, p_t)
5:     end for
6: end function
```

The *symbolic* approach differs from the *basic* one in two principal aspects. First, it requires modification of *bottom-up* algorithm (or equivalent NTP evaluating algorithm) in a way that it performs addition and multiplication on symbolic expressions instead of floating-point numbers. Second, after the last step of the *bottom-up* algorithm, the result is a symbolic expression describing the NTP of terminal node 1. This expression contains a single variable – symbol t representing time. Then, to evaluate system reliability at time t, we evaluate the expression for a given value of t. Thus, with the *basic* approach we evaluate BDD using *bottom-up* algorithm for each time point evaluation whereas with the *symbolic* approach we run the *bottom-up* algorithm only once, and then we evaluate the symbolic expression for each time point. One can notice that this approach is based on the application of formula (19), which defines system reliability as a time-dependent function with a priori given probability distributions specifying reliabilities (13) of the system components over time.

Algorithm 3 presents a pseudocode that shows the *symbolic* approach in the evaluation of system reliability at multiple time points. This can be compared with Algorithm 2 to see the difference between the two approaches. An interesting question is which approach is better if we need to evaluate system reliability at multiple time points. Therefore, in Section VI we provide an experimental comparison of the two approaches in multiple scenarios.

Algorithm 3 Symbolic approach to the evaluation of system reliability in multiple time points.

```
1: function EvaluateSymbolic(bdd, timePoints, p)
2:     exprTree ← CreateTree(bdd, p)
3:     for ∀ t ∈ timePoints do
4:         R ←EvaluateTree(exprTree, t)
5:     end for
6: end function
```

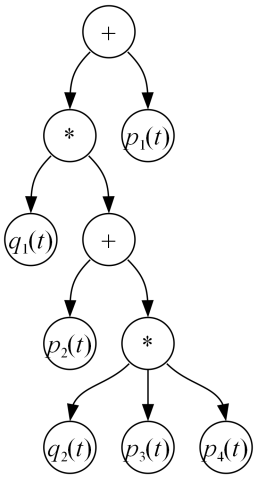

Figure 6. Expression tree representing expression (32).

V. TEDDY

The implementation of probabilistic calculations with BDDs requires support from the software library for the manipulation of BDDs. Several high-quality BDD packages exist including CUDD [8], BuDDy [9], and Sylvan [33]. However, they do not support probabilistic calculations or they support it in a very limited way. Since the probabilistic calculations are one of the cornerstones of reliability evaluation, we have decided to implement our own decision diagram package named TeDDy (**T**emplated **D**ecision **D**iagram librar**y**) [10]. Furthermore, our motivation was also to make the library available under an open-source license with support for MDDs (see. section III).

A. LIBRARY STRUCTURE

Figure 7 shows a UML package diagram describing the logical structure of the library, which consists of four modules. Library implementation uses well-known techniques such as unique tables, apply cache, or node pooling [34] utilized by other decision diagram packages.

The **core** module implements typical functionalities of the decision diagram package. It contains all the algorithms described by Bryant for BDDs [2]. Its biggest asset is that it supports MDDs by providing the MDD structure as well as generalized versions of the aforementioned algorithms.

The **core-io** module is built on top of the core module. It implements input from and output of the diagrams to different formats such as truth table, expression tree, or PLA file [35].

The **reliability** module is the biggest asset of TeDDy. It is dedicated specifically to reliability analysis by BDDs and MDDs. It uses the core module in the implementation of various reliability analysis algorithms proposed by our research group [10]. Using it, we can evaluate importance of system components [21], [24], [36] or find minimal path sets and minimal cut sets [37]–[39]. This module implements the probabilistic algorithms described in section IV. The extension of this module to support time-dependent probabilistic analysis is the central topic of this paper.

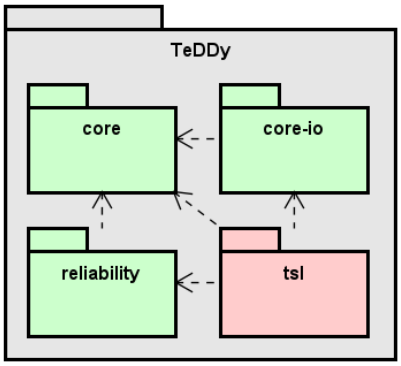

Figure 7. UML package diagram describing the logical structure of the TeDDy library.

Finally, the **tsl** – **t**est **s**upport **l**ibrary – module contains tools used for testing of other modules. In the future, we plan on expanding it to a standalone module providing reliability analysis tools beyond decision diagrams.

B. PROBABILISTIC ANALYSIS

An implementation of the *symbolic* approach requires a modification of the *bottom-up* algorithm in a way that it can add and multiply symbolic expressions instead of numbers. This in turn requires a suitable representation of the symbolic expression. The first option is to manually implement the expression tree. The second option is to use an existing library. In TeDDy, we went with the second option picking the mature library GiNaC [16] – an open-source C++ library for (besides other use cases) the creation, manipulation, and evaluation of symbolic expressions. This made the modification of the *bottom-up* algorithm quite straightforward, since the GiNaC expression overloads arithmetic operators (a feature of the C++ language) and, thus, can "act" as floating-point values. A clever use of C++ templates can allow us to use the same code for time-independent and time-dependent versions of the bottom-up algorithm.

In the time-dependent case, the input of the *bottom-up* algorithm is a vector of GiNaC expressions `GiNaC::ex` representing component state probabilities containing a single variable – symbolic expression representing time `GiNaC::realsymbol("t")`. The output of the algorithm is a single `GiNaC::ex` expression representing the NTP of terminal node 1, which can be evaluated for any value of t. The caching mechanism described in section IV works in the same way – the *memo* table just needs to map pointers to GiNaC expression. It is interesting to note that TeDDy provides an alternative option. It can store the cached data directly in the nodes of the diagram. Thus, the mapping is performed just by reading a property of the node. Comparison of the two approaches in terms of speed will

Table 2. Comparison of the *basic* and *symbolic* approach in the computation of system reliability of the four-component storage system from Figure 3.

Time points	Basic [ns]	Symbolic init [ns]	Symbolic [ns]
10	956	9,252	267,071
100	7,258	9,454	2,607,722
1,000	69,447	9,949	25,945,398
10,000	692,683	13,456	257,718,581

be the subject of our future research since right now it is not clear which of the options is more performant.

Finally, one feature of the *symbolic* approach worth mentioning is that it allows for easier interaction with computer algebra systems – we can serialize the expression and import it into some computer algebra systems for further analysis. For example, we obtained the expression (32) by running the *bottom-up* algorithm with a vector containing symbols such as `GiNaC::realsymbol("p_1")` as input we obtained the chart in Figure 5 by exporting the resulting GiNaC expression (which GiNaC allows in multiple formats) and importing it into R system and using the *ggplot* library to create the chart.

VI. EXPERIMENTAL EVALUATION

As we mentioned in section IV, the comparison of the *basic* and *symbolic* approaches in terms of speed is an interesting question. Since we have both approaches implemented in TeDDy, we performed an experimental comparison in the evaluation of time-dependent system reliability at multiple time points. We performed three experiments on a PC with an Intel i9-10900KF processor with 128GiB of DDR4 RAM running the Void Linux operating system. The code was compiled with the g++-12.2.0 compiler with the -O3 optimization flag.

A. STORAGE SYSTEM EXAMPLE

The first comparison we performed was on the storage system depicted in Figure 3. In the experiment, we evaluated system reliability using component reliabilities presented in Table 1 at 10; 100; 1,000; and 10,000 selected time points. Table 2 shows the result of the comparison. The durations in the table were obtained as average from 100 replications of the computation. Column *Basic* contains the total time in nanoseconds required to compute system reliability at the given number of time points. Column *Symbolic init* contains the time needed to create the expression tree and column *Symbolic* contains the total time in nanoseconds required to compute system reliability at the given number of time points. The results clearly show that the *basic* approach is in the order of magnitude faster than the symbolic approach even when we need to evaluate a higher number of time points.

Table 3. Comparison of the *basic* and *symbolic* approach in the computation of system reliability of randomly generated series-parallel systems.

n	|BDD|	|Tree|	Basic [ns]	Symbolic init [ns]	Symbolic [ns]
10	12	599	1,739	26,187	3,823,367
20	22	15,218	3,606	51,791	101,280,004
30	32	546,208	6,020	82,222	3,595,178,608
40	42	11,494,828	7,401	103,151	72,100,562,769

B. RANDOM SERIES-PARALLEL SYSTEMS

The second comparison aims to compare the two approaches in the analysis of series-parallel systems with different topologies. For this purpose, we generated random series-parallel systems with 10, 20, 30, and 40 components using the approach described in [40]. For each such system, we computed system reliability in 10 time points. Since the systems are randomly generated, we assume exponential distributions of component reliabilities with randomly generated rate parameters. Table 3 contains the results of the comparison. The durations in the table were obtained for each variable count n as average from 10 randomly generated system topologies and 10 replications for each topology. In addition to the previously described columns the table also contains *|BDD|* and *|Tree|* columns, which contain the average number of nodes in BDD and the expression trees respectively.

The results confirm the results of the first experiment that the *basic* approach is significantly faster. Moreover, the results also indicate that the complexity of the expression tree increases dramatically with increasing number of variables. This suggests that the *symbolic* approach is not suitable for a system with a higher number of components while the *basic* approach seems to scale very well if the size of the BDD stays reasonable.

C. PLA BENCHMARK CIRCUITS

The two experiments that we described so far used series-parallel systems. Therefore, in the last experiment, we decided to analyze systems of different nature – PLA circuits from the IWLS'93 benchmark [41]. Reliability analysis of logic circuits is specific since the structure function contains variables representing inputs of the circuits as well as variables representing unreliable logic gates [42]. The analysis aims only at the variables representing the logic gates fixing the input variables for all possible inputs. Hence, the size of the BDD is relatively small despite a higher number of variables n. In this experiment, we also assumed exponential distributions of component reliabilities. Just like in the first experiment, we evaluated system reliability at 10; 100; 1,000; and 10,000 time points.

Table 4 presents the results of the experiment. In addition to the previously described columns, the *PLA file* columns contain the name of the benchmark, and the n column contains the number of variables representing the logic gates – the number of variables in the analyzed BDD. The

Table 4. Comparison of the *basic* and *symbolic* approach in the computation of system reliability of PLA circuits.

PLA file	n	Time points	Basic [ns]	Symbolic init [ns]	Symbolic [ns]
con1	11	10	1,905	986	36,352
con1	11	100	18,002	1,078	357,491
con1	11	1,000	178,788	1,275	3,566,846
con1	11	10,000	1,791,139	2,017	35,643,194
xor5	17	10	2,210	698	23,954
xor5	17	100	21,317	763	237,646
xor5	17	1,000	212,002	874	2,384,453
xor5	17	10,000	2,120,591	1,391	23,797,647
rd53	35	10	3,860	2,158	74,064
rd53	35	100	37,498	2,338	731,114
rd53	35	1,000	374,347	2,709	7,277,917
rd53	35	10,000	3,736,264	4,274	72,870,141
squar5	40	10	4,469	8,019	220,688
squar5	40	100	43,178	8,306	2,189,516
squar5	40	1,000	430,617	9,086	21,934,328
squar5	40	10,000	4,297,874	14,722	218,934,519
sqrt8	44	10	4,864	2,000	67,313
sqrt8	44	100	47,538	2,220	661,964
sqrt8	44	1,000	473,333	2,555	6,612,181
sqrt8	44	10,000	4,760,873	4,303	66,284,840

results show that, again, the *basic* approach performed better than the *symbolic* approach. However, the relative difference between the two approaches is much smaller than in the previous experiments.

Each of the above-described experiments showed that the *basic* approach performs much better than the *symbolic* approach if we consider the speed of evaluating system reliability in multiple time points. Although the results are specific for our implementation – our library TeDDy and GiNaC library for the manipulation of expressions – the relative difference between the two approaches is considerable and therefore is unlikely to change *significantly* for other implementations. However, the *symbolic* approach that we presented is still a valid and useful tool for time-dependent reliability analysis because of the mentioned possibilities to further manipulate and analyze the expression.

VII. CONCLUSION

Reliability belongs to the key characteristics of almost every system. Its evaluation can be complicated and time-consuming in the case of complex systems, which typically are composed of many heterogeneous components. Therefore, computer-aided reliability analysis of such systems requires the application of data structures able to represent their structure and the development of methods that can work with these data structures in an efficient way. One of the most popular data structures used for this purpose is a BDD.

As we presented in this paper, system reliability can be viewed as a function depending on three parameters, which are structure function $\phi(\boldsymbol{x})$, component state probabilities $\boldsymbol{p}(t)$ and time t. Depending on parameters that are a priori known, we distinguish topological, time-dependent probabilistic, and time-independent probabilistic reliability analysis.

In previous works, BDDs had been applied successfully in topological and time-independent probabilistic reliability analysis. In this paper, we showed a simple extension of the basic algorithm for probabilistic evaluation by BDD. The extensions allow the calculation of time-dependent system reliability (and other probabilistic system characteristics) using the basic time-independent algorithm with little to no modifications. Furthermore, we introduced symbolic expressions as another way of time-dependent analysis by BDD. Symbolic expressions allow easier interaction with other computer algebra systems as well as further examination of the system's reliability function. However, our experimental evaluation shows that if we are only interested in the evaluation of system reliability in numerous time points, then the simpler modification of the basic approach is more efficient.

In future research, we would like to focus on the development of methods for time-dependent probabilistic reliability analysis of multi-state systems using TeDDy. In this case, more than two states are used to model the behavior of the system and its components, therefore, MDDs are used to model the structure function of such systems [43].

References

[1] Y. Crama and P. Hammer, Boolean Functions: Theory, Algorithms, and Applications. Cambridge University Press, 2011.

[2] R. Bryant, "Graph-based algorithms for boolean function manipulation," IEEE Transactions on Computers, vol. C-35, no. 8, pp. 677–691, 1986.

[3] C. Meinel and T. Theobald, Algorithms and Data Structures in VLSI Design. Berlin, Heidelberg, DE: Springer Berlin Heidelberg, 1998.

[4] A. Deb, R. Wille, O. Keszöcze, S. Shirinzadeh, and R. Drechsler, "Synthesis of optical circuits using binary decision diagrams," Integration, vol. 59, pp. 42–51, 2017.

[5] T. Hadžić and J. N. Hooker, "Cost-bounded binary decision diagrams for 0-1 programming," in Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, P. Van Hentenryck and L. Wolsey, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 84–98.

[6] A. Rauzy, "Binary decision diagrams for reliability studies," in Handbook of Performability Engineering, K. B. Misra, Ed. London, UK: Springer London, 2008, pp. 381–396.

[7] E. Zaitseva, V. Levashenko, and J. Kostolny, "Importance analysis based on logical differential calculus and binary decision diagram," Reliability Engineering & System Safety, vol. 138, pp. 135–144, 2015.

[8] F. Somenzi, "Cudd: Cu decision diagram package," https://github.com/vscosta/cudd, 2005, accessed: 2023-07-20.

[9] J. Lind-Nielsen, "Buddy – a binary decision diagram package," https://github.com/jgcoded/BuDDy, 2014, accessed: 2023-07-20.

[10] M. Mrena, M. Kvassay, and E. Zaitseva, "Teddy: Templated decision diagram library," SoftwareX, vol. 26, p. 101715, 2024.

[11] A. K. Das and T. K. Roy, "Fractional order EOQ model with linear trend of time-dependent demand," International Journal of Intelligent Systems and Applications, vol. 7, no. 3, pp. 44–53, 2015.

[12] V. Kharchenko, Y. Ponochovnyi, A.-S. M. Q. Abdulmunem, and A. Boyarchuk, "Security and availability models for smart building automation systems," International Journal of Computing, vol. 16, no. 4, pp. 194–202, 2017.

[13] Rakhi and G. L.Pahuja, "Component Importance Measures based Risk and Reliability Analysis of Vehicular Ad Hoc Networks," International Journal of Computer Network and Information Security, vol. 10, no. 10, pp. 38–45, 2018.

[14] L. Ozirkovskyy, B. Volochiy, O. Shkiliuk, M. Zmysnyi, and P. Kazan, "Functional safety analysis of safety-critical system using state transition diagram," Radioelectronic and Computer Systems, no. 2, pp. 145–158, 2022.

[15] T. Nakagawa, Stochastic Processes, ser. Springer Series in Reliability Engineering. London, UK: Springer London, 2011.
[16] C. Bauer, A. Frink, and R. Kreckel, "Introduction to the ginac framework for symbolic computation within the c++ programming language," Journal of Symbolic Computation, vol. 33, no. 1, pp. 1–12, 2002.
[17] E. Babeshko, V. Kharchenko, K. Leontiiev, and E. Ruchkov, "Practical aspects of operating and analytical reliability assessment of FPGA-based I&C systems," Radioelectronic and Computer Systems, no. 3, pp. 75–83, 2020.
[18] H. Fesenko, V. Kharchenko, A. Sachenko, R. Hiromoto, and V. Kochan, "An internet of drone-based multi-version post-severe accident monitoring system: Structures and reliability," in Dependable IoT for Human and Industry, V. Kharchenko, A. L. Kor, and A. Rucinski, Eds. New York: River Publishers, 2022, pp. 197–217.
[19] Y. Sun, H. Fesenko, V. Kharchenko, L. Zhong, I. Kliushnikov, O. Illiashenko, O. Morozova, and A. Sachenko, "UAV and IoT-based systems for the monitoring of industrial facilities using digital twins: Methodology, reliability models, and application," Sensors, vol. 22, no. 17, p. 6444, 2022.
[20] E. Zio, "Reliability engineering: Old problems and new challenges," Reliability Engineering & System Safety, vol. 94, no. 2, pp. 125–141, 2009.
[21] M. Rausand and A. Høyland, System Reliability Theory, 2nd ed. John Wiley & Sons, Inc., 2004.
[22] A. Lisnianski, I. Frenkel, and Y. Ding, Multi-state System Reliability Analysis and Optimization for Engineers and Industrial Managers. London, UK: Springer-Verlag London Ltd., 2010.
[23] B. Natvig, Multistate Systems Reliability Theory with Applications, ser. Wiley Series in Probability and Statistics. Chichester, UK: John Wiley & Sons, Ltd, 2011.
[24] W. Kuo and X. Zhu, Importance Measures in Reliability, Risk, and Optimization: Principles and Applications, 1st ed. Wiley Publishing, 2012.
[25] M. Kvassay and E. Zaitseva, "Topological analysis of multi-state systems based on direct partial logic derivatives," in Recent Advances in Multi-state Systems Reliability, ser. Springer Series in Reliability Engineering, A. Lisnianski, I. Frenkel, and A. Karagrigoriou, Eds. Cham, CH: Springer International Publishing, 2018, pp. 265–281.
[26] J. Newton and D. Verna, "A theoretical and numerical analysis of the worst-case size of reduced ordered binary decision diagrams," ACM Trans. Comput. Logic, vol. 20, no. 1, 2019.
[27] A. Srinivasan, T. Ham, S. Malik, and R. Brayton, "Algorithms for discrete function manipulation," in 1990 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers, 1990, pp. 92–95.
[28] S. Yanushkevich, D. Miller, V. Shmerko, and R. Stankovic, Decision Diagram Techniques for Micro- and Nanoelectronic Design Handbook. CRC Press, 2006.
[29] S. Nagayama, A. Mishchenko, T. Sasao, and J. T. Butler, "Exact and heuristic minimization of the average path length in decision diagrams," J. Multiple Valued Log. Soft Comput., vol. 11, pp. 437–465, 2005.
[30] Y. Mo, L. Xing, and S. V. Amari, "A multiple-valued decision diagram based method for efficient reliability analysis of non-repairable phased-mission systems," IEEE Transactions on Reliability, vol. 63, no. 1, pp. 320–330, 2014.
[31] P. Rusnak, J. Rabcan, M. Kvassay, and V. G. Levashenko, "Time-dependent reliability analysis based on structure function and logic differential calculus," in International Conference on Dependability of Computer Systems, 2018.
[32] S. Du, R. Kang, Z. Zeng, and E. Zio, "Time-dependent reliability assessment of a distributed generation system based on multi-valued decision diagrams and markov processes," in Safety and Reliability – Theory and Applications. CRC Press, 2017.
[33] T. van Dijk, Sylvan: multi-core decision diagrams. University of Twente, 2016.
[34] K. Brace, R. Rudell, and R. Bryant, "Efficient implementation of a bdd package," in 27th ACM/IEEE Design Automation Conference, 1990, pp. 40–45.
[35] K. McElvain, "Iwls'93 benchmark set: Version 4.0," Tech. Rep., 1993.
[36] E. Zaitseva and V. Levashenko, "Importance analysis by logical differential calculus," Automation and Remote Control, vol. 74, 2013.
[37] W.-C. Yeh, "A new algorithm for generating minimal cut sets in k-out-of-n networks," Reliability Engineering & System Safety, vol. 91, no. 1, pp. 36–43, 2006.
[38] ——, "An improved algorithm for searching all minimal cuts in modified networks," Reliability Engineering & System Safety, vol. 93, no. 7, pp. 1018–1024, 2008.
[39] M. Forghani-elahabad and N. Kagan, "An approximate approach for reliability evaluation of a multistate flow network in terms of minimal cuts," Journal of Computational Science, vol. 33, pp. 61–67, 2019.
[40] M. Mrena, M. Kvassay, and S. Czapp, "Single and series of multi-valued decision diagrams in representation of structure function," in Lecture Notes in Networks and Systems, vol. 484 LNNS, 2022, pp. 176–185.
[41] P. Fišer, "Collection of digital design benchmarks," https://ddd.fit.cvut.cz/www/prj/Benchmarks/, 1991, accessed: 2023-10-31.
[42] M. Kvassay, E. Zaitseva, V. Levashenko, and J. Kostolny, "Reliability analysis of multiple-outputs logic circuits based on structure function approach," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 3, pp. 398–411, 2017.
[43] A. Shrestha, L. Xing, and Y. Dai, "Decision diagram based methods and complexity analysis for multi-state systems," IEEE Transactions on Reliability, vol. 59, no. 1, pp. 145–161, 2010.

MICHAL MRENA received the bachelor's degree, master's degree, and PhD degree in informatics from the University of Zilina, Slovakia in 2019, 2021, and 2024 respectively. He is currently an assistant professor with the Department of Informatics at the Faculty of Management Science and Informatics, University of Zilina, Slovakia. His research focuses on data structures, optimization of program code, and decision diagrams in the reliability analysis of complex systems.

MIROSLAV KVASSAY received the bachelor's degree, master's degree, and PhD degree in informatics from the University of Zilina, Slovakia in 2010, 2012, and 2015 respectively. He is currently an associate professor with the Department of Informatics at the Faculty of Management Science and Informatics, University of Zilina, Slovakia. He deals with the theory of reliability engineering, Boolean and multiple-valued logic, and their applications in reliability analysis.