Journal of Applied Logic 18 (2016) 19–41

Contents lists available at ScienceDirect

Journal of Applied Logic

www.elsevier.com/locate/jal

Formalization of Reliability Block Diagrams in Higher-order Logic

Waqar Ahmed [a,*], Osman Hasan [a], Sofiène Tahar [b]

[a] *School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad, Pakistan*
[b] *Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada*

ARTICLE INFO

Article history:
Received 26 May 2015
Received in revised form 15 February 2016
Accepted 25 May 2016
Available online 15 June 2016

Keywords:
Reliability Block Diagrams (RBDs)
Higher-order logic
Probability theory
Virtualization configuration
Virtual Data Centers

ABSTRACT

Reliability Block Diagrams (RBDs) allow us to model the failure relationships of complex systems and their sub-components and are extensively used for system reliability, availability and maintainability analyses. Traditionally, these RBD-based analyses are done using paper-and-pencil proofs or computer simulations, which cannot ascertain absolute correctness due to their inaccuracy limitations. As a complementary approach, we propose to use the higher-order logic theorem prover HOL to conduct RBD-based analysis. For this purpose, we present a higher-order logic formalization of commonly used RBD configurations, such as series, parallel, parallel-series and series-parallel, and the formal verification of their equivalent mathematical expressions. A distinguishing feature of the proposed RBD formalization is the ability to model nested RBD configurations, which are RBDs having blocks that also represent RBD configurations. This generality allows us to formally analyze the reliability of many real-world systems. For illustration purposes, we formally analyze the reliability of a generic Virtual Data Center (VDC) in a cloud computing infrastructure exhibiting the nested series-parallel RBD configuration.

1. Introduction

Reliability Block Diagrams (RBDs) [6] are used to assess various failure-related characteristics, such as reliability [18], availability [13] and maintainability [7], of a wide range of engineering systems. An RBD is primarily a graphical structure that consists of blocks and connectors (lines) representing the functional behavior of the system components and their interconnectivity with each other, respectively. For example, while assessing the reliability of a computational software, the blocks may represent the computational elements, with some given failure rate, and the connectors between them may be used to describe various alternative paths required for a successful computation using the given software [1]. Now, based on this

* Corresponding author.
E-mail addresses: waqar.ahmad@seecs.nust.edu.pk (W. Ahmed), osman.hasan@seecs.nust.edu.pk (O. Hasan), tahar@ece.concordia.ca (S. Tahar).

http://dx.doi.org/10.1016/j.jal.2016.05.007
1570-8683/

RBD, the failure characteristics of the overall system can be judged based on the failure rates of individual components, whereas the overall system failure happens if all paths for successful execution fail. The RBD analysis enables us to evaluate the impact of component failures on the overall system reliability and thus is widely used for assessing the trade-offs of various possible system configurations, such as series, parallel or a combination of both, at the system design stage.

Traditionally, RBD-based analysis is carried out using paper-and-pencil proof methods and computer simulations. The first step in the paper-and-pencil proof methods is to express the reliability of each component of the system in terms of its failure rate λ and a random variable, like exponential [31] or Weibull [16], which models the failure time. This information, along with the RBD of the system, is then used to analytically derive mathematical expressions for the system-level failure characteristics. Due to the involvement of manual manipulation and simplification, this kind of analysis is prone to human errors and the problem becomes more sever when analyzing large systems. Moreover, it is possible, and in fact a common occurrence, that many key assumptions required for the analytical proofs are in the mind of the specialist assisting the system engineers in the analysis of the system and they are hence not documented. These missing assumptions are thus not communicated to the design engineers and are ignored in system implementations, which may also lead to unreliable designs. On the other hand, computer simulators, such as ReliaSoft [24] and ASENT reliability analysis tool [4], have been extensively used for the RBD analysis of the various real-world systems. However, they cannot ensure absolute correctness as well due to the involvement of pseudo-random numbers and numerical methods.

To overcome the above-mentioned inaccuracy problems, formal methods have also been proposed for the RBD-based analysis using both state-based [21,25] and theorem proving techniques [3]. However, state-based approaches can neither be used to reason about continuous elements and nor for verifying generic reliability expressions. These limitations can be overcome by using theorem proving, given the high expressiveness of higher-order logic and inherent soundness of the provers, and thus generic mathematical expressions involving continuous elements can be verified.

In [3], we presented a formalization of the series RBD using the HOL4 theorem prover [27]. This formalization was then successfully used to verify the reliability of an oil and gas pipeline [3]. However, most of the RBDs for real-world systems involve a combination of series and parallel configurations. Moreover, another limitation of the series RBD formalization of [3] is that the series RBD function takes a single-dimension list of random variables as an argument, where each element of this list models a single component of the structure. This fact limits the usage of this function to model the case when the system as well as its components are also modeled by the RBD configurations, or in other words, this formalization does not cater for nested RBD configurations. The ability to handle such nested RBD configurations requires assigning the random variables to each block or sub-stage of the system-level RBD.

To overcome the above-mentioned limitations, we propose a deep embedding approach to formalize the commonly used RBD configurations, such as series, parallel, parallel-series and series-parallel. In particular, we introduce a recursive datatype *rbd* to formalize RBD configurations consisting of type constructors, such as `series`, `parallel` and `atomic`. Then, a semantic function over the *rbd* datatype is defined with the ability to decode the RBD configuration encoded by these type-constructors to yield the corresponding reliability event, which corresponds to the scenario when the given system or component does not fail before a certain time. This proposed formalization approach is compositional in nature and can be easily extended to cater for any combination of series and parallel RBD configurations. Also, it allows us to verify the generic reliability expressions for RBDs on any reliability event list of arbitrary length and thus overcomes the above-mentioned limitations of series RBD formalization [3]. To elaborate the compositional ability of the proposed RBD formalization, we also present a higher-order logic formalization of a nested series-parallel RBD, which is a series-parallel RBD having each block itself modeled by a series-parallel RBD configuration.

To illustrate the practical effectiveness of our work, we utilize our proposed RBD formalization to conduct the formal reliability analysis of a generic Virtual Data Center (VDC) system in a cloud computing

infrastructure [30]. A VDC can be viewed as series connection of network modules, where clusters, which in turn contain a bunch of physical (or cloud) servers that are connected in parallel [29]. In order to make an efficient use of hardware resources, hardware virtualization [29] is utilized within each cloud server. This virtualization configuration ascertains the reliability of a cloud server and can be modeled as a series-parallel RBD [23]. Therefore, the reliability of the complete VDC can be analyzed by utilizing the nested series-parallel RBD configuration, where the outer RBD models the connection of clusters, and the inner RBD corresponds to the virtualization configuration in a cloud server [30] and thus the goal is to attain the most reliable configuration. Due to the large number of VDC components and the continuous nature of failure rates, and the associated random variables, traditional techniques, like simulation or model checking, cannot ascertain accurate results for this analysis. Whereas, the proposed RBD-based analysis approach allowed us to analyze a generic n-cluster model of the VDC.

2. Related Work

RBD-based computer simulators, such as ReliaSoft [24] and the ASENT reliability analysis tool [4], generate samples from the exponential and Weibull random variables to model the reliabilities for the components of the system. This data is then manipulated using computer arithmetic and numerical techniques to compute the reliability of the complete system. These software are more scalable than paper-and-pencil proof methods. However, they cannot ensure absolute correctness either due to the involvement of pseudo-random numbers or numerical methods.

Colored Petri Nets (CPN) have also been used to model dynamic RBDs (DRBDs) [25], which are used to describe dynamic reliability behavior of systems. CPN verification tools, based on model checking principles, are then used to verify behavioral properties of the DRBDs models to identify design flaws [25]. Similarly, the probabilistic model checker, PRISM [22], has been used for the quantitative verification of various safety and mission-critical systems, such as failure analysis for an airbag system and the reliability analysis of an industrial process control system and the Herschel–Planck satellite system [21]. However, due to the state-based models, only state-related property verification, like deadlock checks, reachability and safety properties, is supported by these approaches, that is, we cannot verify mathematical reliability relationships for the given systems using the approaches, presented in [21,25]. For example, a state-based model of a specific scenario, with predefined transition probabilities, can only be analyzed using model checking tools. On the other hand, higher-order logic theorem proving has the ability to inductively verify the reliability relationships for an arbitrary number of system components and failure rates and thus can be used to carry out the reliability analysis of a wide variety of real-world systems.

The higher-order logic theorem prover HOL4 has been recently used for RBD analysis and some preliminary results related to the reliability analysis of oil and gas pipelines, composed of serially connected components, are reported in [3]. In particular, this work utilizes the probability theory developed in [19], which is available in the HOL4 theorem prover, to formalize reliability, exponential random variables and series RBD. These foundations are then used to formally verify a generic expression of a simple pipeline structure that has been modeled as a series RBD with an exponential failure time for individual segments. The main limitation of this work is its focused nature since very few real-world systems can be modeled as a series RBD. Moreover, the formalization, presented in [3], does not allow the nesting of RBD structures.

In the current paper, we extend the work in [3] to formally reason about series, parallel, parallel-series and series-parallel RBDs and cater for nested RBDs as well. We believe that the results of this paper widen the scope of formal RBD analysis as most of the real-world systems require parallel or a combination of series and parallel RBDs and nested RBDs for modeling their respective behaviors.

3. Preliminaries

In this section, we give a brief introduction to theorem proving and the HOL4 theorem prover to facilitate the understanding of the rest of the paper.

3.1. Theorem Proving

Theorem proving [11] is a widely used formal verification technique. The system that needs to be analyzed is mathematically modeled in an appropriate logic and the properties of interest are verified using computer-based formal tools. The use of formal logics as a modeling medium makes theorem proving a very flexible verification technique as it is possible to formally verify any system that can be described mathematically. The core of theorem provers usually consists of some well-known axioms and primitive inference rules. Soundness is assured as every new theorem must be created from these basic or already proved theorems and primitive inference rules.

The verification effort of a theorem in a theorem prover varies from trivial to complex depending on the underlying logic [12]. For instance, first-order logic [10] utilizes the propositional calculus and terms (constants, function names and free variables) and is semi-decidable. A number of sound and complete first-order logic automated reasoners are available that can enable automated proofs for large set of problems. More expressive logics, such as higher-order logic [8], can be used to model a wider range of problems than first-order logic, but theorem proving for these logics cannot be fully automated and thus involves user interaction to guide the proof tools. For the formalization of RBDs, we need to formalize random variables as functions, and their distribution properties are verified by quantifying over random variable functions. Henceforth, first-order logic does not support such formalization and we need to use higher-order logic to formalize the foundations of RBDs that are then in turn used to formally analyze the reliability of various real-world systems, such as VDCs.

3.2. HOL4 Theorem Prover

HOL4 is an interactive theorem prover developed at the University of Cambridge, UK, for conducting proofs in higher-order logic. It utilizes the simple type theory of Church [9] along with Hindley–Milner polymorphism [20] to implement higher-order logic. HOL4 has been successfully used as a verification framework for both software and hardware as well as a platform for the formalization of pure mathematics.

The HOL4 core consists of only 5 basic axioms and 8 primitive inference rules, which are implemented as ML functions. The ML's type system ensures that only valid theorems can be constructed. Soundness is assured as every new theorem must be verified by applying these basic axioms and primitive inference rules or any other previously verified theorems/inference rules.

In the work presented in this paper, we utilize the HOL4 theories of Booleans, lists, sets, positive integers, *real* numbers, measure and probability. In fact, one of the primary motivations of selecting the HOL4 theorem prover for our work was to benefit from these built-in mathematical theories. Table 1 provides the mathematical interpretations of some frequently used HOL4 symbols and functions, which are inherited from existing HOL4 theories.

4. Probability and Reliability in HOL

Mathematically, a measure space is defined as a triple (Ω, Σ, μ), where Ω is a set, called the sample space, Σ represents a σ-algebra of subsets of Ω, where the subsets are usually referred to as measurable sets, and μ is a measure with domain Σ. A probability space is a measure space (Ω, Σ, Pr), such that the measure, referred to as the probability and denoted by Pr, of the sample space is 1. In the HOL4 formalization

Table 1
HOL symbols and functions.

HOL4 symbol	Standard symbol	Meaning
$\wedge$	*and*	Logical *and*
$\vee$	*or*	Logical *or*
$\neg$	*not*	Logical *negation*
::	*cons*	Adds a new element to a list
++	*append*	Joins two lists together
`HD L`	*head*	Head element of list L
`TL L`	*tail*	Tail of list L
`EL n L`	*element*	nth element of list L
`MEM a L`	*member*	True if a is a member of list L
λ`x.t`	$\lambda x.t$	Function that maps x to $t(x)$
`SUC n`	$n+1$	Successor of a *num*
`lim(`λ`n.f(n))`	$\lim_{n\to\infty} f(n)$	Limit of a *real* sequence f

of probability theory [19], given a probability space p, the functions `space`, `subsets` and `prob` return the corresponding Ω, Σ and Pr, respectively. This formalization also includes the formal verification of some of the most widely used probability axioms, which play a pivotal role in formal reasoning about reliability properties.

A random variable is a measurable function between a probability space and a measurable space. The measurable functions belong to a special class of functions, which preserves the property that the inverse image of each measurable set is also measurable. A measurable space refers to a pair $(S, \mathcal{A})$, where S denotes a set and $\mathcal{A}$ represents a nonempty collection of sub-sets of S. Now, if S is a set with finite number of elements, then the corresponding random variable is termed as a discrete otherwise it is known as continuous random variable.

The probability that a random variable X is less than or equal to some value t, $Pr(X \leq t)$ is called the Cumulative Distribution Function (CDF) and it characterizes the distribution of both discrete and continuous random variables. The CDF has been formalized in HOL4 as follows [3]:

```
⊢ ∀ p X t. CDF p X t = distribution p X {y | y ≤ Normal t}
```

where the variables $p : (\alpha \to bool)\#((\alpha \to bool) \to bool)\#((\alpha \to bool) \to real)$, $X : (\alpha \to extreal)$ and $t : real$ represent a probability space, a random variable and a *real* number respectively. The function `Normal` takes a *real* number as its inputs and converts it to its corresponding value in the *extended-real* data-type, i.e., it is the *real* data-type with the inclusion of positive and negative infinity. The function `distribution` takes three parameters: a probability space p, a random variable X and a set of *extended-real* numbers and outputs the probability of a random variable X that acquires all the values of the given set in probability space p.

Now, reliability $R(t)$ is stated as the probability of a system or component performing its desired task over certain interval of time t.

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

where $F_X(t)$ is the CDF. The random variable X, in the above definition, models the time to failure of the system and is usually modeled by the exponential random variable with parameter λ, which corresponds to the failure rate of the system. Based on the HOL4 formalization of probability theory [19], Equation (1) has been formalized as follows [3]:

```
⊢ ∀ p X t. Reliability p X t = 1 - CDF p X t
```

The series RBD, presented in [3], is based on the notion of mutual independence of random variables, which is one of the most essential prerequisites for reasoning about the mathematical expressions for all RBDs. If N reliability events are mutually independent then

$$Pr(\bigcap_{i=1}^{N} A_i) = \prod_{i=1}^{N} Pr(A_i) \tag{2}$$

This concept has been formalized as follows [3]:

```
⊢ ∀ p L. mutual_indep p L = ∀ L1 n. PERM L L1 ∧
     1 ≤ n ∧ n ≤ LENGTH L ⇒
      prob p (inter_list p (TAKE n L1)) =
      list_prod (list_prob p (TAKE n L1))
```

The function `mutual_indep` accepts a list of events L and probability space p and returns $True$ if the events in the given list are mutually independent in the probability space p. The predicate `PERM` ensures that its two lists as its arguments form a permutation of one another. The function `LENGTH` returns the length of the given list. The function `TAKE` returns the first n elements of its argument list as a list. The function `inter_list` performs the intersection of all the sets in its argument list of sets and returns the probability space if the given list of sets is empty. The function `list_prob` takes a list of events and returns a list of probabilities associated with the events in the given list of events in the given probability space. Finally, the function `list_prod` recursively multiplies all the elements in the given list of real numbers. Using these functions, the function `mutual_indep` models the mutual independence condition such that for any 1 or more events n taken from any permutation of the given list L, the property $Pr(\bigcap_{i=1}^{N} A_i) = \prod_{i=1}^{N} Pr(A_i)$ holds.

5. Reliability Block Diagrams

Reliability Block Diagrams (RBDs) [6] are graphical structures consisting of blocks and connector lines. The blocks usually represent the system components and the connection of these components is described by the connector lines. The system is functional, if at least one path of properly functional components from input to output exists, otherwise it fails.

An RBD construction can follow any of these three basic patterns of component connections: (i) series (ii) active redundancy or (iii) standby redundancy. In the series connection, shown in Fig. 1(a), all components have to be functional for the system to remain functional. Whereas, in an active redundancy all components in at least one of the redundant stages must be functioning in fully operational mode. The components in an active redundancy might be connected in a parallel structure (Fig. 1(b)) or a combination of series and parallel structures as shown in Figs. 1(c) and 1(d). In a standby redundancy, all components are not required to be active. Three types of information are necessary to build the RBD of a given system: (i) functional interaction of the system components, (ii) reliability of each component, and (iii) mission times at which the reliability is desired. This information is then utilized by the design engineers to identify the appropriate RBD configuration (series, parallel or series-parallel) in order to determine the overall reliability of the given system. A detailed account of the commonly used RBD configurations and their corresponding mathematical expressions is given as follows:

Series Reliability Block Diagram. The reliability of a system with components connected in series is considered to be reliable at time t only if all of its components are functioning reliably at time t, as depicted in Fig. 1(a). If $A_i(t)$ is a mutually-independent event that represents the reliable functioning of the ith component of a serially connected system with N components at time t, then the overall reliability of the complete system can be expressed as [6]:

$$R_{series}(t) = Pr(\bigcap_{i=1}^{N} A_i(t)) = \prod_{i=1}^{N} R_i(t) \tag{3}$$

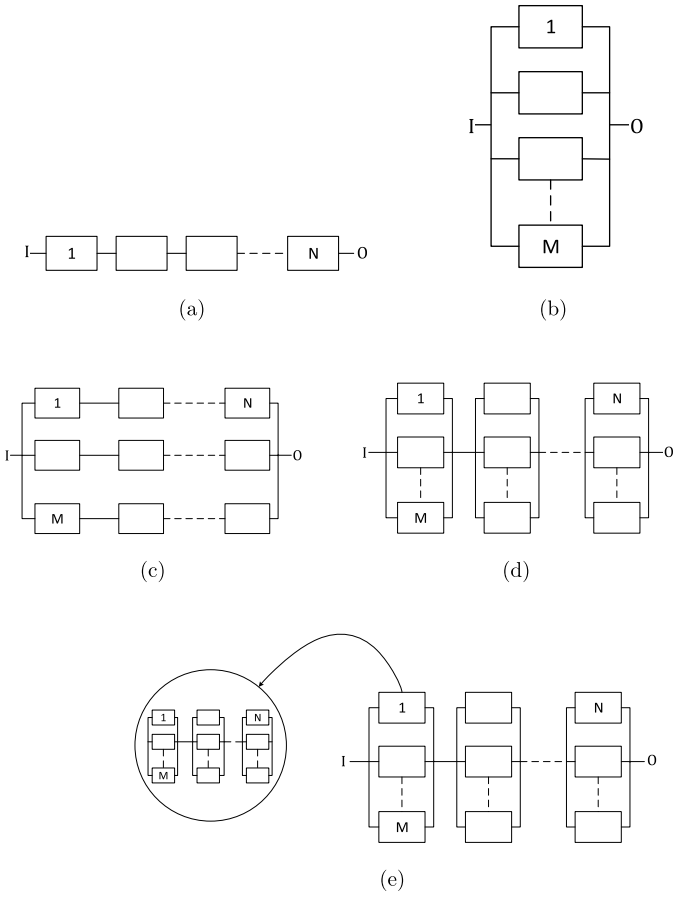

Fig. 1. RBDs (a) Series. (b) Parallel. (c) Parallel-Series. (d) Series-Parallel. (e) Nested Series-Parallel.

Parallel Reliability Block Diagram. The reliability of a system with parallel connected sub-modules, depicted in Fig. 1(b), essentially depends on the component with the maximum reliability. In other words, the system will continue functioning as long as at least one of its components remains functional. If the event $A_i(t)$ represents the reliable functioning of the ith component of a system with M parallel components at time t, then the overall reliability of the system can be mathematically expressed as [6]:

$$R_{parallel}(t) = Pr(\bigcup_{i=1}^{M} A_i) = 1 - \prod_{i=1}^{M}(1 - R_i(t)) \tag{4}$$

Nested Reliability Block Diagrams. Most safety-critical systems in the real-world contain many reserved stages for backup in order to ensure reliable operation [17,26]. If the components in these reserved *subsystems* are connected serially then the structure is called a parallel-series structure, as depicted in Fig. 1(c). The parallel-series RBD is a nested form of series RBD in a parallel RBD configuration. If $A_{ij}(t)$ is the event corresponding to the reliability of the jth component connected in a ith subsystem at time t, then the reliability of the complete system can be expressed as follows:

$$R_{parallel\text{-}series}(t) = Pr(\bigcup_{i=1}^{M}\bigcap_{j=1}^{N} A_{ij}(t)) = 1 - \prod_{i=1}^{M}(1 - \prod_{j=1}^{N}(R_{ij}(t))) \tag{5}$$

Similarly, if in each serial stage the components are connected in parallel, then the configuration is termed as a *series-parallel* structure, shown in Fig. 1(d). If $A_{ij}(t)$ is the event corresponding to the proper

functioning of the jth component connected in an ith subsystem at time index t, then the reliability of the complete system can be expressed mathematically as:

$$R_{series\text{-}parallel}(t) = Pr(\bigcap_{i=1}^{N}\bigcup_{j=1}^{M} A_{ij}(t)) = \prod_{i=1}^{N}(1 - \prod_{j=1}^{M}(1 - R_{ij}(t))) \tag{6}$$

In many cases, real-world systems involve sub-components, which themselves form a nested RBD configuration, as shown in Fig. 1(e). Such systems can be modeled by nested RBD configurations. For instance, if a system and its components both are modeled by the series-parallel RBDs, then the complete system can be modeled by using a nested series-parallel RBD configuration. The reliability of this kind of nested series-parallel RBD can be expressed mathematically as follows:

$$\begin{aligned} R_{nested\text{-}series\text{-}parallel}(t) =& Pr(\bigcap_{i=1}^{N}\bigcup_{j=1}^{M}(\bigcap_{k=1}^{N}\bigcup_{l=1}^{M} A_{ijkl}(t))) \\ =& \prod_{i=1}^{N}(1 - \prod_{j=1}^{M}(1 - (\prod_{k=1}^{N}(1 - \prod_{l=1}^{M}(1 - (R_{ijkl}(t))))))) \end{aligned} \tag{7}$$

where, i and j are the indices of the outer series-parallel RBD and the indices k and l refer to the reliability events corresponding to the inner sub-components of the system.

By utilizing the above-mentioned RBD configurations, we can easily construct the reliability models of many real-world systems. For instance, consider a typical radar system [14] consisting of an antenna, receiver/transmitter system, tracking computers and radar controller, as shown in Fig. 2(a). The information from the antenna is received at the receiver/transmitter system and then sent to the tracking computers for processing. The processed information is then given to the radar controller for an accurate representation of the originally received information. If any of these radar system components malfunction or a break occurs in the flow of information, then the system no longer remains functional and is considered to be in a failed state. Since each component is essential for a radar system to be functional, the system can be modeled from series RBD, which can be seen in Fig. 2(b). However, if we are interested in a detailed reliability analysis of the radar system then we can include the sub-components of the receiver/transmitter system and all tracking computers that are connected in parallel for redundancy. This detailed radar system can be easily modeled by using series-parallel RBD configuration, as shown in Fig. 2(c).

From the RBDs presented in this section, we can notice that only the series and parallel RBDs are the fundamental configurations while other configurations can be easily constructed by nesting the different combinations of these RBDs. In the next section, we present the formalization of series, parallel and nested RBD configurations. These formalized configurations can then be used in turn to formally model systems behaviors in HOL4 and reason about their reliability, availability and maintainability characteristics.

6. Formalization of the Reliability Block Diagrams

The proposed formalization is primarily based on defining a new polymorphic datatype *rbd* that encodes the notion of series and parallel configurations. Then a semantic function is defined on that *rbd* datatype yielding an event for the corresponding RBD configuration. This semantic function allows us to verify the generic reliability expressions of the RBD configurations, that are described in the previous section, by utilizing the underlying probability theory within the sound core of the HOL4 theorem prover. Such a deep embedding considerably simplifies the RBD modeling approach, compared to our previous work [3] (shallow embedding), and also enables us to develop a framework that can deal with arbitrary levels of nested RBD configurations, which can be used to cater for a wide variety of real-world reliability analysis problems.

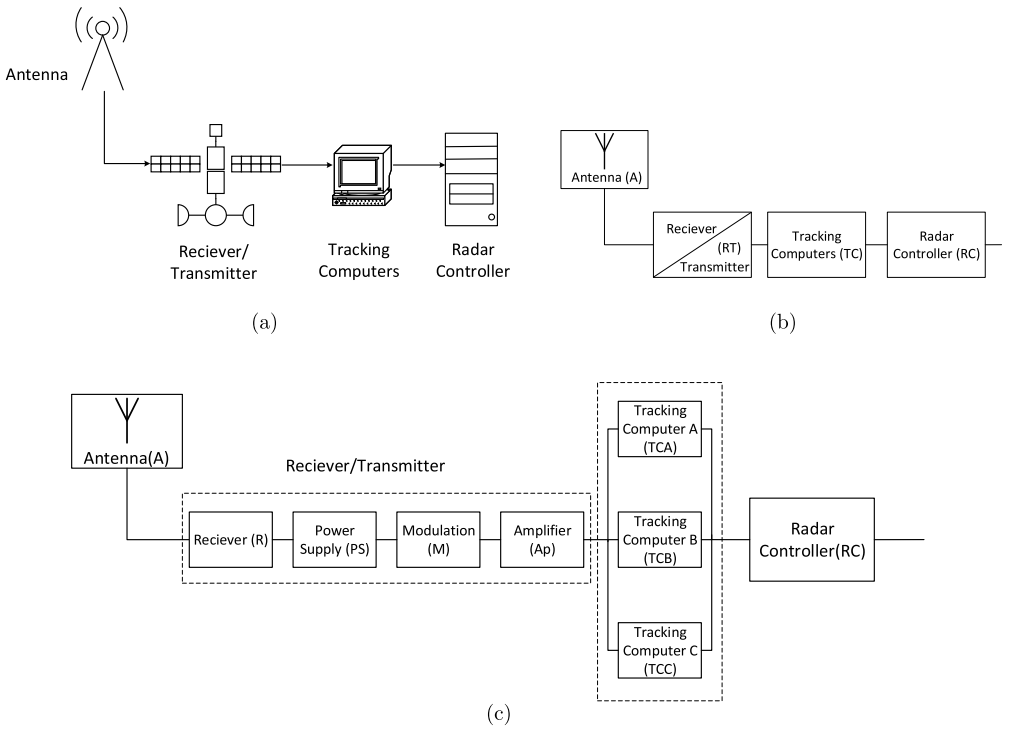

Fig. 2. (a) A typical radar system. (b) RBD of a simplified radar system. (c) RBD of a detailed radar system.

We verify the reliability expressions for the commonly used RBD configurations, as presented in Section 5, on reliability event lists, where a single event represents the scenario when the given system or component does not fail before a certain time:

Definition 1. ⊢ ∀ p X t.

```
rel_event p X t = PREIMAGE X {y | Normal t < y} ∩ p_space p
```

The function `PREIMAGE` takes two arguments, a function f and a set s, and returns a set, which is the domain of the function f operating on a given range set s. The function `rel_event` accepts a probability space p, a random variable X, representing the failure time of a system or a component, and a real number t, which represents the time index at which the reliability is desired. It returns an event representing the reliable functioning of the system or component at time t.

Similarly, a list of reliability events is derived by mapping the function `rel_event` on each element of the given random variable list as follows:

Definition 2. ⊢ ∀ p L t.

```
rel_event_list p L t = MAP (λa. rel_event p a t) L
```

In [3], the series RBD function operates on a single-dimension list of random variables, where each random variable in a list is associated with a block of the series RBD configuration. A major limitation of this modeling approach arises when dealing with nested RBD configurations, such as parallel-series and series-parallel, where the blocks themselves are modeled by RBD configurations. To cater for these RBD configurations, we are required to model a random variable that can incorporate the notion of multiple random variables. For instance, to formalize the parallel-series RBD, we need to assign a random variable to each one of the serial stage such that the random variables associated to each parallel stage model all the

random variables that are assigned to the corresponding components connected in a serial stage and thus making the RBD formalization of [3] challenging. In order to simplify the formalization of nested RBDs, we propose to distinguish the notion of random variable from the reliability event. We thus propose to formally verify generic RBD reliability relationships on reliability event lists. These formally verified expressions can then be used with the random variables corresponding to each component of the system for analyzing the reliability of systems that can be represented as nested RBDs.

We start the formalization process by type abbreviating the notion of event, which is essentially a set of observations with type `'a->bool` as follows:

```
type_abbrev ("event" , ":'a ->bool")
```

We then define a recursive datatype *rbd* in the HOL4 system as follows:

```
Hol_datatype ‘rbd = series of rbd list |
                    parallel of rbd list |
                    atomic of 'a event‘
```

An RBD can either be a series configuration, a parallel configuration or an atomic event. The type constructors `series` and `parallel` recursively function on *rbd*-typed lists and thus enable us to deal with nested RBD configurations. The type constructor `atomic` is basically a typecasting operator between *event* and *rbd*-typed variables. Typically, a new datatype is defined in HOL4 as $(\alpha_1, \alpha_2, ..., \alpha_n)op$, where $(\alpha_1, \alpha_2, ..., \alpha_n)$ represent the arguments taken by the HOL4 datatype *op* [28]. For instance, the `atomic` type constructor is defined with the arbitrary type α, which is taken by the already defined type `events`. On the other hand, the type constructors `series` and `parallel` are defined without any arguments because the datatype `rbd` is not defined at this point.

We define a semantic function `rbd_struct`: (α *event* # α *event event* # (α *event* → *real*) → α *rbd* → α *event*) inductively over the *rbd* datatype. It can yield the corresponding event from the given RBD configuration as follows:

Definition 3.
```
⊢ (∀ p. rbd_struct p (series []) = p_space p) ∧
(∀ xs x p.
  rbd_struct p (series (x::xs)) =
  rbd_struct p x ∩ rbd_struct p (series xs)) ∧
(∀ p. rbd_struct p (parallel []) = {}) ∧
(∀ xs x p.
  rbd_struct p (parallel (x::xs)) =
  rbd_struct p x ∪ rbd_struct p (parallel xs) ∧)
(∀ p a. rbd_struct p (atomic a) = a)
```

The above function decodes the semantic embedding of an arbitrary RBD configuration by yielding a corresponding reliability event, which can then be used to determine the reliability of a given RBD configuration. The function `rbd_struct` takes an *rbd*-typed list identified by a type constructor `series` and returns the whole probability space if the given list is empty and otherwise returns the intersection of the events that are obtained after applying the function `rbd_struct` on each element of the given list in order to model the series RBD configuration behavior. Similarly, to model the behavior of a parallel RBD configuration, the function `rbd_struct` operates on an *rbd*-typed list encoded by a type constructor `parallel`. It then returns the union of the events after applying the function `rbd_struct` on each element of the given list or an empty set if the given list is empty. The function `rbd_struct` returns the reliability event using the type constructor `atomic`.

Now using Definition 3, we can formally verify the reliability expression, given in Equation (3), for a series RBD configuration in HOL4 as follows:

Table 2
Mutual independence lemmas.

Theorem	HOL formalization
mutual_indep_cons	∀ h p L. mutual_indep p (h::L) ⇒ mutual_indep p L
mutual_indep_append_sym	∀ p L1 L. mutual_indep p (L1 ++ L) ⇒ mutual_indep p (L ++ L1)
mutual_indep_front_append	∀ p L1 L. mutual_indep p (L1 ++ L) ⇒ mutual_indep p L
mutual_indep_append_swap	∀ p L1 L2 L. mutual_indep p (L1 ++ L2 ++ L) ⇒ mutual_indep p (L2 ++ L1 ++ L)
mutual_indep_cons_append	∀ p h L1 L. mutual_indep p (h::L1 ++ L2) ⇒ mutual_indep p (L1 ++ h::L2)
mutual_indep_cons_append1	∀ p Q h L1 L. mutual_indep p (h::L1 ++ Q::L) ⇒ mutual_indep p (L1 ++ Q::h::L)
mutual_indep_cons_append2	∀ h p L1 L. mutual_indep p (L1 ++ h::L) ⇒ mutual_indep p (L1 ++ L)
mutual_indep_cons_append3	∀ h p L1 L2 L. mutual_indep p (L1 ++ h::(L2 ++ L)) ⇒ mutual_indep p (h::(L1 ++ L))
mutual_indep_cons_append4	∀ h p L1 L. mutual_indep p (L1 ++ h::L) ⇒ mutual_indep p (L1 ++ L)
mutual_indep_cons_flat	∀ h p L. mutual_indep p (FLAT (h::L)) ⇒ mutual_indep p (FLAT L)
mutual_indep_flat_append	∀ h p L1 L2 L. mutual_indep p (FLAT (L1::L2::L)) ⇒ mutual_indep p (L1 ++ L2)
mutual_indep_flat_append1	∀ h p L1 L. mutual_indep p (FLAT (L1::h::L)) ⇒ mutual_indep p (FLAT ((h ++ L1)::L))
mutual_indep_flat_append2	∀ p L1 L2 L. mutual_indep p (FLAT (L1::L2::L)) ⇒ mutual_indep p (L1 ++ L2)
mutual_indep_cons_flat2	∀ h p L1 L2 L. mutual_indep p (FLAT (L1::(h::L2)::L)) ⇒ mutual_indep p (FLAT ([h]::L))

Theorem 1. ⊢ ∀ p L. prob_space p ∧
¬NULL L ∧ (∀x'. MEM x' L ⇒ x' ∈ events p) ∧
mutual_indep p L ⇒
(prob p (rbd_struct p (series (rbd_list L))) =
 list_prod (list_prob p L))

The first assumption, in Theorem 1, ensures that p is a valid probability space based on the probability theory in HOL4 [19]. The next two assumptions guarantee that the list of events L, representing the reliability of individual components, must have at least one event and the reliability events are mutually independent. The conclusion of the theorem represents Equation (3). The function `rbd_list` generates a list of type *rbd* by mapping the function `atomic` to each element of the given event list L to make it consistent with the assumptions of Theorem 1. It can be formalized in HOL4 as:

∀ L. rbd_list L = MAP (λa. atomic a) L

The proof of Theorem 1 is primarily based on a mutual independence lemma `mutual_indep_cons`, given in Table 2, and some fundamental axioms of probability theory.

The above-mentioned series RBD can be easily utilized to model the series RBD configuration of the radar system, which is described in Section 5. To do this, we are only required to replace the arbitrary list L with the `rel_event_list p [A;RT;TC;RC]`, where A, RT, TC and RC are the associated random variables of the antenna, receiver/transmitter system, tracking computers and the radar controller, respectively.

Similarly by following the above-mentioned formalization approach of series RBD, we can formally verify the reliability expression for the parallel RBD configuration, given in Equation (4), in HOL4 as follows:

Theorem 2. ⊢ ∀ p L.
```
prob_space p ∧ (∀x'. MEM x' L ⇒ x' ∈ events p) ∧
¬NULL L ∧ mutual_indep p L ⇒
 (prob p (rbd_struct p (parallel (rbd_list L))) =
  1 - list_prod (one_minus_list (list_prob p L)))
```

The above theorem is verified under the same assumptions as Theorem 1. The conclusion of the theorem represents Equation (4) where, the function `one_minus_list` accepts a list of *real* numbers $[x1, x2, x3, \cdots, xn]$ and returns the list of *real* numbers such that each element of this list is 1 minus the corresponding element of the given list, i.e., $[1 - x1, 1 - x2, 1 - x3, \cdots, 1 - xn]$.

To verify Theorem 2, we need to verify a lemma that provides an alternate expression for the parallel RBD in terms of the series RBD configuration. As the series and parallel RBD configurations are represented mathematically from the intersection and union of events, respectively. So, this lemma can be expressed mathematically as follows:

$$P(\bigcup_{i=1}^{N} A_i) = 1 - P(\bigcap_{i=1}^{N} A_i) \tag{8}$$

The HOL4 formalization of Equation (8) is as follows:

Lemma 1. ⊢ ∀ p L. prob_space p ∧ ¬NULL L ∧
```
(∀x'. MEM x' L ⇒ x' ∈ events p) ∧
 mutual_indep p L ⇒
 (prob p (rbd_struct p (parallel (rbd_list L))) =
  1 - prob p (rbd_struct p (series (rbd_list (compl_list p L)))))
```

The proof of Theorem 2 is primarily based on Lemma 1 and Theorem 1 along with the fact that given the list of n mutually independent events, the complement of these n events are also mutually independent:

```
⊢ ∀ p L. prob_space p ∧ mutual_indep p L ∧
¬NULL L ∧ (∀x'. MEM x' L ⇒ x' ∈ events p) ⇒
    mutual_indep p (compl_list p L1)
```

The function `compl_list` returns a list of events such that each element of this list is the difference between the probability space p and the corresponding element of the given list. The proof process of the above lemma utilizes mutual independence properties of Table 2 as well as various other probability independence lemmas that can be found in [2].

The above formalization described for series and parallel RBD configurations builds the foundation to formalize the combination of series and parallel RBD configurations. The type constructors `series` and `parallel` can take the argument list containing other *rbd* type constructors, such as `series`, `parallel` or `atomic`, allowing the function `rbd_struct` to yield the corresponding event for an RBD configuration that is composed of a combination of series and parallel RBD configurations.

By extending the RBD formalization approach, presented in Theorems 1 and 2, we formally verified the generic reliability expression for parallel-series RBD configuration, given in Equation (5), in HOL4 as follows:

Theorem 3.
```
⊢ ∀ p L. prob_space p ∧
(∀z. MEM z L ⇒ ∽NULL z) ∧
(∀x'. MEM x' (FLAT L) ⇒ x' ∈ events p) ∧
 mutual_indep p (FLAT L) ⇒
 (prob p
   (rbd_struct p ((parallel of (λa. series (rbd_list a))) L)) =
  (1 - list_prod (one_minus_list) of
    (λa. list_prod (list_prob p a))) L)
```

The first assumption in Theorem 3 is similar to the one used in Theorem 2. The next three assumptions ensure that the sub-lists corresponding to the serial sub-stages are not empty and the reliability events corresponding to the sub-components of the parallel-series configuration are valid events of the given probability space p and are also mutually independent. The HOL4 function `FLAT` is used to flatten the two-dimensional list, i.e., to transform a list of lists, into a single list. The conclusion models the right-hand side of Equation (5). The infixr function `of` connects two *rbd* type-constructors by using the HOL4 `MAP` function and thus facilitates the natural readability of complex RBD configurations. It is formalized in HOL4 as follows:

```
⊢ ∀ g f. f of g = (f o (λa. MAP g a))
```

Similarly, the generic expression of the series-parallel RBD configuration, given in Equation (6), is formalized in HOL4 as follows:

Theorem 4.
```
⊢ ∀ p L. prob_space p ∧
(∀z. MEM z L ⇒ ∽NULL z) ∧
(∀x'. MEM x' (FLAT L) ⇒ x' ∈ events p) ∧
 mutual_indep p (FLAT L) ⇒
  (prob p
     (rbd_struct p ((series of (λa. parallel (rbd_list a))) L)) =
  (list_prod of
     (λa. 1 - list_prod (one_minus_list (list_prob p a)))) L)
```

The assumptions of Theorem 4 are similar to those used in Theorem 3. The conclusion models the right-hand side of Equation (6). To verify Theorems 3 and 4, it is required to formally verify various structural independence lemmas, for instance, given the list of mutually independent reliability events, an event corresponding to the series or parallel RBD structure is independent, in probability, with the corresponding event associated with the parallel-series or series-parallel RBD configurations. Some of the foundational structural independence lemmas are presented in Table 3. These lemmas are verified under the following assumptions: (i) `prob_space p` ensures that p is a valid probability space; (ii) `(∀x. MEM x (L1::L) ⇒ ¬NULL x)` guarantees that the given list must not be empty; (iii) `(∀x. MEM x (FLAT (L1::L)) ⇒ x ∈ events p)` makes sure that each event in a given list is a valid event in a probability space p; and (iv) `mutual_indep p (FLAT (L1::L))` ensures that the given list of events are mutually independent in probability. The proof of these lemmas are primarily based on the mutual independence lemmas, given in Table 2, and many fundamental probability theory axioms, for instance, Probabilistic Inclusion–exclusion (PIE) Principle for two events, which can be found in [2].

The modeling of series-parallel RBD, presented in Theorem 4, can be easily extended to model the series-parallel RBD configuration of the detailed radar system, as shown in Fig. 2(c). The HOL4 formalization of the corresponding series-parallel RBD of the detailed radar system is as follows:

Table 3
Independence of RBD configurations lemmas.

Lemma description	HOL formalization
Probability Independence of Series and Parallel-Series RBD Configurations	∀ p L1 L. (prob p (rbd_struct p (series (rbd_list L1)) ∩ rbd_struct p ((parallel of (λa. series (rbd_list a))) L)) = prob p (rbd_struct p (series (rbd_list L1))) * prob p (rbd_struct p ((parallel of (λa. series (rbd_list a))) L)))
Probability Independence of Parallel and Parallel-Series RBD Configurations	∀ p L1 L. (prob p (rbd_struct p (parallel (rbd_list L1)) ∩ rbd_struct p ((parallel of (λa. series (rbd_list a))) L)) = prob p (rbd_struct p (parallel (rbd_list L1))) * prob p (rbd_struct p ((parallel of (λa. series (rbd_list a))) L)))
Probability Independence of Series and Series-Parallel RBD Configurations	∀ p L1 L. (prob p (rbd_struct p (series (rbd_list L1)) ∩ rbd_struct p ((series of (λ a. parallel (rbd_list a))) L)) = prob p (rbd_struct p (series (rbd_list L1))) * prob p (rbd_struct p ((series of (λa. parallel (rbd_list a))) L)))
Probability Independence of Parallel and Series-Parallel RBD Configurations	∀ p L1 L. (prob p (rbd_struct p (parallel (rbd_list L1)) ∩ rbd_struct p ((series of (λa. parallel (rbd_list a))) L)) = prob p (rbd_struct p (parallel (rbd_list L1))) * prob p (rbd_struct p ((series of (λa. parallel (rbd_list a))) L)))

```
⊢ ∀ p A R PS M Ap TCA TCB TCC RC t.
radar_series_parallel_rbd p A R PS M Ap TCA TCB TCC RC t =
rbd_struct p
   ((series of (λa. parallel (rbd_list (rel_event_list p a t))))
        ([[A];[R];[PS];[M];[Ap];[TCA;TCB;TCC];[RC]]))
```

where $A, R, PS, M, Ap, TCA, TCB, TCC$ and RC are the corresponding random variables of antenna, receiver, power supply, modulator, amplifier, tracking computer A, B and C and the radio controller, respectively.

Now, using Theorem 4, we can formally model and verify the reliability relationship of a nested series-parallel RBD configuration as well, given in Equation (7), in HOL4 as follows:

Theorem 5.
```
⊢ ∀ p L.
prob_space p ∧ (∀ z. MEM z (FLAT (FLAT L)) ⇒ ¬NULL z) ∧
(∀ x'. MEM x' (FLAT (FLAT (FLAT L))) ⇒ x' ∈ events p) ∧
mutual_indep p (FLAT (FLAT (FLAT L))) ⇒
(prob p (rbd_struct p
  ((series of parallel of series of
    (λa. parallel (rbd_list a))) L)) =
(list_prod of (λa. 1 - list_prod (one_minus_list a)) of
 (λa. list_prod a) of
  (λa. 1 - list_prod (one_minus_list (list_prob p a)))) L)
```

Most of the assumptions of the above theorem are very similar to those used in Theorem 4 and the remaining ones are used to ensure that the reliability event lists are not empty and their corresponding reliability events are mutually independent with respect to the given probability space. The proof of above theorem uses the results of Theorems 1 and 3 and various lemmas, like the ones presented in Table 3, stating that

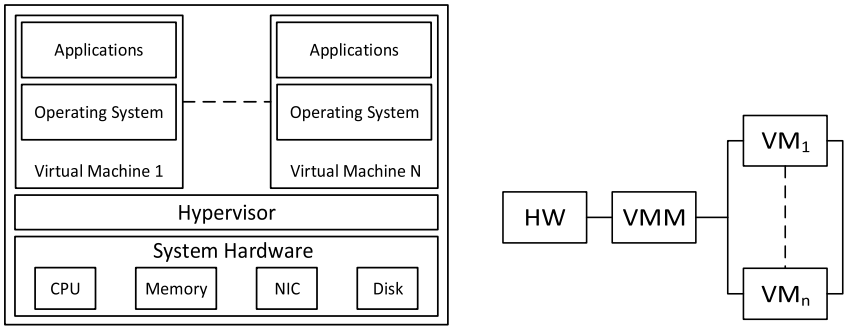

Fig. 3. a) Virtualization configuration in a cloud server. b) Equivalent series-parallel RBD model.

given the mutually independent reliability events list, a reliability event associated with the sub-component of the inner series-parallel RBD configuration is independent of the reliability event associated with the nested series-parallel RBD configuration.

The formalization, reported in this paper, took about 3500 lines of HOL4 proof script and took about 400 man-hours. It is worthwhile to mention that the generic formalization of lemmas, presented in Tables 2 and 3, related to mutual independence and probability independence of RBD configurations significantly reduced the HOL4 proof script for RBD formalization, which is available for download at [2]. The formal reasoning was primarily based on probabilistic, set-theoretic and arithmetic simplification and some parts of the proofs were also handled automatically using the various built-in automatic provers and advanced simplifiers in HOL4. The main benefit of the formalization, presented in this section, is the ability to formally analyze the reliability aspects of safety-critical systems within the sound environment of a theorem prover, as will be demonstrated using a Virtual Data Center example in the next section.

7. Application: Virtual Data Center

Virtual Data Centers (VDCs) are mainly utilized as an infrastructure of cloud computing and are heavily populated by virtualized resource pools of storage and computing. A typical VMware [29] data center consists of basic physical computing blocks such as x86 servers, storage networks and arrays, IP networks, a management server and desktop clients. A number of x86 servers can be grouped together, with shared storage and network subsystems, to form a Cluster.

One of the widely used methods to achieve virtualization of these physical servers is based on utilizing a layer of software, known as a Virtual Machine Monitor (VMM) or hypervisor [5], which links the physical hardware with the operating system and allows hardware resource multiplexing between multiple virtual machines (VMs), as shown in Fig. 3(a). Each VM has its own operating system, generally known as guest OS, and virtual hardware resources, such as a virtual CPU, virtual network card, virtual RAM, and virtual disks. A VMM, can be directly hosted on a physical computer system, such as Xen [5], or within a host operating system, such as VMware [29].

The dependability of the VDCs is primarily based on the reliability of the virtualization configuration in a cloud server, which in turn affects the reliability of the cloud computing infrastructure. A study for cloud computing vulnerabilities shows that there were about 172 unique cloud computing outage incidents between 2008 and 2012 [15]. The major causes of these incidents include (i) insecure interfaces and APIs, (ii) data loss and leakages and (iii) hardware failures [15]. The main victims of these vulnerabilities include Google, Amazon, Microsoft and Apple, and the vulnerabilities resulted in heavy financial losses [15]. Due to the increasing usage of cloud computing in meeting computing needs of all kinds of application domains, including online shopping, financial services, medicine and transportation, their reliability has become of

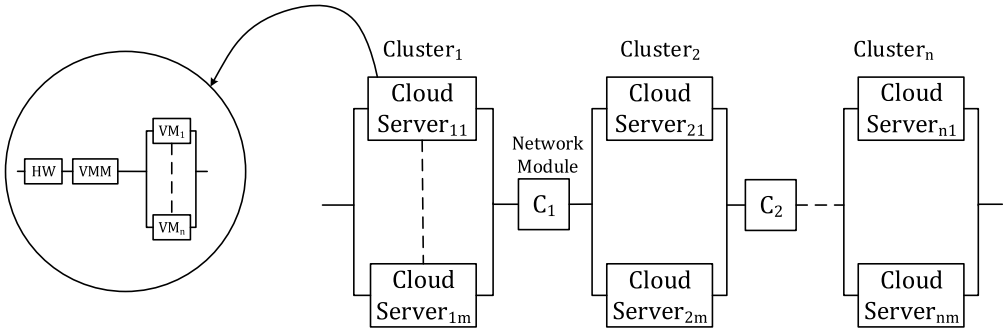

Fig. 4. VDC cloud RBD model.

utmost importance and thus a significant time and effort is spent on their reliability analysis prior to their deployment.

A series-parallel RBD of a typical VDC, shown in Fig. 4, is composed of n network modules, and n clusters, whereas each cluster is in turn composed of m cloud servers. The virtualization configuration in these cloud servers, presented in Fig. 3(a), can also be modeled by a series-parallel RBD as shown in Fig. 3(b). So, a complete VDC can be modeled by using nested series-parallel RBD configuration, as shown in Fig. 1(e), where the outer series-parallel RBD models the cloud servers, clusters and the network modules and the inner series-parallel RBD models the virtualization configuration in a cloud server. The reliability expression of this nested series-parallel RBD, as depicted in Fig. 4, can be expressed mathematically as [30]:

$$R_{VDC} = \prod_{i=1}^{n}[(1 - \prod_{j=1}^{m}(1 - R_{Server_{ij}}))R_{C_i}] \tag{9}$$

where, $R_{Server_{ij}}$ and R_{C_i} represent the reliabilities of cloud servers and network modules.

The reliability of a cloud server can be expressed mathematically as:

$$R_{Server} = R_{HW}R_{VMM}[1 - \prod_{i=1}^{n}(1 - R_{VM_i})] \tag{10}$$

where, $R_{Server}, R_{HW}, R_{VMM}$ and R_{VM_i} represent the reliability of the overall configuration in a cloud server having n VMs, physical hardware, a hypervisor and the ith virtual machine, respectively.

The inequality for the lower bound that determines the number of VMs, which are essentially required to bring the reliability of the virtualized physical cloud server above that of an unvirtualized server [23], can be expressed mathematically as:

$$n > \frac{\log(1 - \frac{R_{Server}}{R\_VM})}{\log(1 - R\_VMM)} \tag{11}$$

We will now first present the formal reliability analysis of the virtualization configuration in a cloud server and then use this formally verified relation to conduct the formal reliability analysis of a VDC.

7.1. Reliability of the virtualization configuration

Consider a virtualization configuration that consists of a physical hardware HW, a hypervisor VMM, and n VMs, such that $(n > 1)$. Each VM is concurrently performing identical tasks and working independently. For example, a VM can be considered as a virtual server, which responds to client requests for static web

content. The random variables X_VM, X_VMM and X_HW associated with the virtualization configuration components is assumed to be exponentially distributed with the failure rate C_VM, C_VMM and C_HW [30]. The reliability of the virtualization configuration modeled by series-parallel RBD configuration with exponential distribution can be expressed mathematically as:

$$R_{Server} = (\exp^{-(\lambda_{VMM}+\lambda_{HW})t})[1 - \prod_{i=1}^{n}(1 - \exp^{-\lambda_{VM_i}t})] \tag{12}$$

In order to formally verify the above equation, we first formally defined the notion of exponential distribution in HOL4 as follows:

Definition 4. ⊢ ∀ p X c. exp_dist p X c =

∀ t. (CDF p X t = if 0 ≤ t then 1 - exp (-c * t) else 0)

The function `exp_dist` guarantees that the CDF of the random variable X is that of an exponential random variable with a failure rate c in a probability space p. We classify a list of exponentially-distributed random variables based on this definition as follows:

Definition 5. ⊢ (∀ p L. exp_dist_list p L [] = T) ∧
∀ p h t L. exp_dist_list p L (h::t) =
exp_dist p (HD L) h ∧ exp_dist_list p (TL L) t

where the symbol T stands for logical *True*. The function `exp_dist_list` accepts a list of random variables L, a list of failure rates and a probability space p. It guarantees that all elements of the random variable list L are exponentially distributed with the corresponding failure rates, given in the other list, within the probability space p. For this purpose, it utilizes the list functions `HD` and `TL`, which return the *head* and *tail* of a list, respectively.

Using the RBD formalization approach, described in Section 6, we can construct a RBD for virtualization configuration in HOL4 as follows:

Definition 6. ⊢ ∀ p X_VMM X_HW X_VM t.
rbd_virt_cloud_server p X_VMM X_HW X_VM t =
rbd_struct p
((series of (λa. parallel (rbd_list (rel_event_list p a t))))
[[X_VMM];[X_HW];X_VM]))

where X_VMM, X_HW and X_VM are the random variables associated with the virtualization configuration components. We have utilized the series-parallel RBD formalization, presented in Theorem 4, to model the virtualization configuration RBD by simply using its associated reliability event.

Now, using the above definitions, we formally verified the reliability of virtualization configuration, given in Equation (12), in HOL4 as follows:

Theorem 6. ⊢ ∀ X_VM X_VMM X_HW C_VM C_VMM C_HW p t.
[A1]: ¬NULL X_VM ∧ [A2:] 0 ≤ t ∧ [A3]: prob_space p ∧
[A4]: in_events p (rel_event_list p ([[X_VMM];[X_HW];X_VM]) t) ∧
[A5]: mutual_indep p (rel_event_list p ([[X_VMM];[X_HW];X_VM]) t) ∧
[A6]: LENGTH C_VM = LENGTH X_VM ∧
[A7]: exp_dist_list p [[X_VMM];[X_HW];X_VM] [[C_VMM];[C_HW];C_VM] ⇒
(prob p (rbd_virt_cloud_server p X_VMM X_HW X_VM t =
exp (-(C_VMM + C_HW) * t) *
(1 - list_prod (one_minus_list (exp_func_list C_VM t)))

where `X_VMM, X_HW` are the random variables corresponding to the failure distributions of a VMM and physical hardware HW, respectively. The list `X_VM` corresponds to the list of random variables that in turn represent the failure distribution of virtual machines (VMs). Similarly, `C_VMM` and `C_HW` are the failure rates for the hypervisor VMM and physical hardware HW, respectively, and the list `C_VM` represents the list of failure rates for the virtual machines VMs. The first two assumptions (A1–A2) of the above theorem guarantee that the list of random variables associated with the virtual machines must not be empty and the time index must be positive. The next two assumptions (A3–A4) ensure that p is a valid probability space and the reliability events associated with the virtualization configuration must be in the events space. The predicate function `in_events` takes a probability space p and an events list L and makes sure that each element must be in events space p. It can be formalized in HOL4 as `(∀x'. MEM x' L ⇒ x' ∈ events p)`. The assumption (A5) guarantees that the reliability events associated with the virtualization configuration components are mutually independent. The last two assumptions (A6–A7) ensure that the length of the list of random variables and the corresponding list of failure rates for virtual machines is the same, and that the exponential distributions of the virtualization configuration components, connected in the series-parallel structure, are associated with their respective failure rates, respectively. The conclusion of the above theorem models Equation (12) such that the left-hand side of the conclusion is the series connection of the VMM, the physical hardware HW and the parallel connection of VMs, shown in Fig. 3(b), and the right-hand side provides the simplified expression for the reliability evaluation of the virtualization configuration in a physical cloud server. The functions `list_prod`, `one_minus_list` and `exp_func_list` accept a two-dimensional list of failure rates and return a list with products of one minus exponentials of every sublist. For example, `list_prod (one_minus_list (exp_func_list a t))` $[[c1;c2;c3];[c4;c5];[c6;c7;c8]]x = [(1-e^{-(c1)x})*(1-e^{-(c2)x})*(1-e^{-(c3)x}); (1-e^{-(c4)x})*(1-e^{-(c5)x}); (1-e^{-(c6)x})*(1-e^{-(c7)x})*(1-e^{-(c8)x})]$.

The proof of Theorem 6 involves Theorem 4, some probability theory axioms and several properties of the exponential function.

7.1.1. Bounds for the number of VMs

It is important to find out the number of VMs required to bring the reliability of the virtualized physical cloud server above that of an unvirtualized server [23]. The HOL formalization of Equation (11) is as follows:

Theorem 7. ⊢ ∀ X_VM X_VMM X_HW p n t.
[A1]: prob_space p ∧ [A2]: 0 ≤ t ∧
[A3]: ¬NULL (gen_rv_list X_VM n) ∧
[A4]: rel_event p X_VMM t ∈ events p ∧
rel_event p X_VM t ∈ events p ∧
rel_event p X_HW t ∈ events p ∧
[A5]: in_events p (rel_event_list p (gen_rv_list X_VM n) t) ∧
[A6]: rel_virt_cloud_server p [[X_VMM];[X_HW];gen_rv_list X_VM n] t <
Reliability p X_VMM t ∧
[A7]: (Reliability p X_HW t < 1) ∧ (0 < Reliability p X_VMM t) ∧
(0 < Reliability p X_VM t ∧ Reliability p X_VM t < 1) ∧
[A8]: mutual_indep p
(rel_event_list p (X_VMM::X_HW::gen_rv_list X_VM n) t) ⇒

$$\&n > \frac{\log10\ (1 - \frac{\text{rel\_cloud\_server p [[X\_VMM];[X\_HW];gen\_rv\_list X\_VM n] t}}{\text{Reliability p X\_VMM t}})}{\log10\ (1 - \text{Reliability p X\_VM t})}$$

where `X_VM, X_VMM and X_HW` are the random variables associated with virtual machine VM_i, virtual machine Monitor VMM and a physical hardware HW, as depicted in Fig. 3(b). The function `gen_rv_list` accepts a number n and a random variable and generates a list of identical random variables of length n by using the given random variable. The function `rel_cloud_server` returns the overall reliability of

the virtualization configuration, which is exhibiting a series-parallel structure, by utilizing Theorem 4. The assumptions of Theorem 7 are similar to the ones used in Theorem 6, with the addition of some essential bounds on the reliability of the virtualization configuration components. The conclusion of the above theorem models Equation (11). The proof of the above theorem involves Theorem 4 and some formally-verified properties from real and probability theory in HOL4.

7.2. Reliability of a Virtual Data Center

The reliability of the VDC with virtualization server exhibiting exponential distribution can be expressed mathematically as follows:

$$R_{VDC_{nm}} = \prod_{i=1}^{n}[1 - \prod_{j=1}^{m}(1 - R_{Server_{ij}}) * \exp^{-\lambda_{C_i} t}] \tag{13}$$

where λ_{C_i} is the failure rate of the ith network module connected between the clusters, as shown in Fig. 4.

Now, using the formalization of the nested series-parallel RBD configuration, presented in Section 6, we formalized the RBD configuration of VDC in HOL4 as follows:

Definition 7. ⊢ ∀ p X_C X_VMM X_HW X_VM m n t.
rbd_VDC_cloud p X_C X_VMM X_HW X_VM m n t =
rbd_struct p (series (rbd_list (rel_event_list p X_C t))) ∩
rbd_struct p ((series of parallel of series of
(λa. parallel (rbd_list (rel_event_list p a t))))
(cloud_server_rv_list [[X_VMM];[X_HW];X_VM] m n)))

where X_C is the random variable associated with the VDC network module. The function `cloud_server_rv_list` takes a two dimensional list of random variables L and two numbers m and n and returns four-dimensional list of n random variables by utilizing a function `gen_list` which accepts a list L and returns a list which contains m copies of the same list L. The function `cloud_server_rv_list` can be defined in HOL4 as follows:

⊢ ∀ L m n. cloud_server_rv_list L m n = gen_list (gen_list L m) n

where the function `gen_list` takes an arbitrary list and a number, say n, and generates the n copies of the given list.

Now, by using Definition 7 and the formalization presented in the Section 6, we can formally analyze the reliability of the complete VDC, given in Equation (13), in HOL4 as follows:

Theorem 8. ⊢ ∀ X_VM X_VMM X_HW X_C C_VM C_VMM C_HW C m n p t.
[A1]: 0 ≤ t ∧ prob_space p ∧
[A2]: ¬NULL (cloud_server_rv_list [X_VM] m n) ∧ ¬NULL X_VM ∧
¬NULL (cloud_server_fail_rate_list [C_VM] m n) ∧ ¬NULL C_VM ∧
[A3]: not_null_list
(FLAT (FLAT (cloud_server_rv_list [X_VM] m n))) ∧
¬NULL (rel_event_list p X_C t) ∧
[A4]: (LENGTH C = LENGTH X_C) ∧ (LENGTH X_VM = LENGTH C_VM) ∧

```
[A5]: in_events p (FLAT (FLAT (FLAT (four_dim_rel_event_list p
        (cloud_server_rv_list [X_VM] m n) t)))) ∧
[A6]: rel_event p X_VMM t ∈ events p ∧
      rel_event p X_VM t ∈ events p ∧
      rel_event p X_HW t ∈ events p ∧
      in_events p (rel_event_list p X_C t) ∧
[A7]: exp_dist_list p X_C C ∧
      four_dim_exp_dist_list p
        (cloud_server_rv_list [[X_VMM];[X_HW];X_VM] m n)
        (cloud_server_fail_rate_list [[C_VMM];[C_HW];C_VM] m n) ∧
[A8]: mutual_indep p (rel_event_list p X_C t ++
       FLAT (FLAT (FLAT (four_dim_rel_event_list p
        (cloud_server_rv_list [[X_VMM];[X_HW];X_VM] m n) t)))) ⇒
  (prob p (rbd_VDC_cloud p X_C X_VMM X_HW X_VM m n t) =
    list_prod (exp_func_list C t) *
     (list_prod of (λa. 1 - list_prod (one_minus_list a)) of
      (λa. list_prod a) of
       (λa. 1 - list_prod (one_minus_list (exp_func_list a t))))
         (cloud_server_fail_rate_list [[C_VMM];[C_HW];C_VM] m n))
```

The assumptions are quite similar to the ones that are used in Theorem 6. The predicate function `not_null_list` in assumption (A3) makes sure that each element of the given list must not be empty and it can be formalized in HOL4 as `(∀z. MEM z L ⇒ ¬NULL z)`. The function `four_dim_exp_dist_list` accepts the probability space p, a four dimensional lists of random variables and failure rates and imitates the same behavior as that of the function `exp_dist_list`, which is already described in the explanation of Theorem 6. The function `four_dim_rel_event_list` takes a probability space p, a four dimensional list of random variables, and a real number t, which represents the time index at which the reliability is desired. It returns a corresponding four dimensional list of reliability events by applying the function `rel_event_list`, described in Definition 2, on each element of the random variables list. The function `cloud_server_fail_rate_list` accepts a two dimensional list of failure rates L and two numbers m and n and returns an n-length four dimensional list of failure rates where each sub-lists of `cloud_server_fail_rate_list` contains the m length list of failure rates corresponding to the random variables associated with the virtualization configuration in a cloud server. The conclusion of the above theorem models the Equation (9). The proof of the above Theorem utilizes the results of Theorems 4, 5 and 6, and properties of probability theory and the exponential function.

Theorem 8 is verified for n-clusters that are connected through network modules and each of these clusters contains m cloud servers. The universal quantification on all variables of Theorem 8 allows us to specialize the analysis of this theorem for any number of clusters or cloud servers. For example, the reliability of a VDC for the case of 3 clusters, 2 network modules, 2 cloud servers and 3 virtual machines in each cloud server can be verified as follows:

$$\begin{aligned} R_{VDC_{32}} = e^{-\lambda_{C_1+C_2}t} * (1 - (1 - e^{-\lambda_{VMM+HW}t} * \\ (1 - (1 - e^{-\lambda_{VM1}t}) * ((1 - e^{-\lambda_{VM2}t}) * (1 - e^{-\lambda_{VM3}t}))))^2)^3) \end{aligned} \tag{14}$$

Theorem 9. `⊢ ∀ X_VM1 X_VM2 X_VM3 X_VMM X_HW X_C1 X_C2 C_VM1 C_VM2 C_VM3 C_VMM C_HW C1 C2 p t.`

```
[A1]: 0 ≤ t ∧ prob_space p ∧
[A2]: in_events p (rel_event_list p
        [X_C1;X_C2;X_VMM;X_HW;X_VM1;X_VM2;X_VM3] t) ∧
```

```
[A3]: exp_dist_list p [X_C1;X_C2;X_VMM;X_HW;X_VM1;X_VM2;X_VM3]
                      [C1;C2;C_VMM;C_HW;C_VM1;C_VM2;C_VM3] ∧
[A4]: mutual_indep p (rel_event_list p [X_C1;X_C2] t ++
       FLAT(FLAT(FLAT (four_dim_rel_event_list p
        (cloud_server_rv_list
          [[X_VMM];[X_HW];[X_VM1;X_VM2;X_VM3]] 2 3) t)))) ⇒
 (prob p (rbd_struct p
     (series (rbd_list (rel_event_list p [X_C1;X_C2] t))) ∩
  rbd_struct p ((series of parallel of series of
   (λa. parallel (rbd_list (rel_event_list p a t))))
    cloud_server_rv_list [[X_VMM];[X_HW];[X_VM1;X_VM2;X_VM3]] 2 3)) =
  exp (-((C1 + C2) * t)) *
  (1 - (1 - exp (-((C_VMM + C_HW) * t)) *
    (1 - (1 - exp (-(C_VM1 * t))) * ((1 - exp (-(C_VM2 * t))) *
      (1 - exp (-(C_VM3 * t)))))) pow 2) pow 3)
```

The assumptions in the above theorem are significantly simplified and the conclusion represents Equation (14). The proof of Theorem 9 is primarily based on Theorem 8 and only required some real-theoretic reasoning.

7.3. Discussion

The reasoning process for the formal verification of Theorems 6, 7 and 8 took about 1500 lines of HOL4 script [2] and was very straightforward compared to the reasoning for the verification of Theorems 4 and 5, which involved probability-theoretic guidance. This fact illustrates the usefulness of our core formalization for conducting the RBD analysis of virtualization configurations using a theorem prover. Table 4 reports on the effort involved in the verification of Theorems 1–8 in terms of HOL4 proof script lines and man-hours consumed.

The distinguishing feature of the formally verified Theorems 6, 7 and 8, compared to the traditional reliability analysis of the VDC [23,30], include its generic nature and guaranteed correctness. The reliability analysis is conducted for an n-stage VDC and all variables are universally quantified and thus can be specialized to obtain the reliability of any VDC and virtualization configuration in a VDC for any given failure rates. This fact is quite evident in Theorem 9, which utilizes Theorem 8, to provide a reliability analysis of a VDC consisting of only 3 clusters, 2 network modules, 2 cloud servers and 3 virtual machines. The correctness of our results is guaranteed thanks to the involvement of a sound theorem prover in their verification, which ensures that all required assumptions for the validity of the results are accompanying the theorem. Unlike the work presented in [23,30], the formally verified reliability results in Theorems 6–9 are sound and obtained through rigorous reasoning process during the mechanization of their proofs. To the best of our knowledge, the above-mentioned benefits are not shared by any other computer-based VDC reliability analysis approach.

Table 4
Verification detail for each theorem.

Formalized theorems	Proof lines	Man-hours
Theorem 1 (Series RBD)	90	10
Theorem 2 (Parallel RBD)	775	88
Theorem 3 (Parallel-Series RBD)	766	90
Theorem 4 (Series-Parallel RBD)	460	55
Theorem 5 (Nested Series-Parallel RBD)	1150	131
Theorem 6 (Virtualization Configuration)	297	20
Theorem 7 (Inequality Bounds)	253	13
Theorems 8 and 9 (Virtual Data Centers)	992	50

8. Conclusions

The accuracy of reliability analysis of engineering systems has become a dire need these days due to their extensive use in safety-critical applications, where an incorrect reliability estimate may lead to disastrous situations including the loss of lives [7,13]. In this paper, we presented a higher-order logic formalization of commonly used RBD configurations, namely, series, parallel, parallel-series and series-parallel, and also a nested series-parallel RBD to facilitate the formal reliability analysis of safety-critical systems within the sound environment of a theorem prover. In order to illustrate the effectiveness of the proposed method, we analyzed the reliability of a Virtual Data Center. The generic nature and soundness of our analysis were found to be the distinguishing features. Building upon the results presented in this paper, the formalization of other commonly used RBDs, including series-parallel and K-out-of-N, and the Weibull random variable is underway. Besides these foundational developments, we also plan to conduct some extensive case studies involving some mechanical systems, such as oil and gas pipelines and automobiles.

To facilitate the utilization of our proposed approach, we plan to build a GUI that can be used to capture any RBD model, like the Virtual Data Center RBD, from the user and return the formally-verified reliability expression, by using the HOL4 theorem prover that is running seamlessly beneath this GUI, of the given system. This would bring great benefits to non-HOL users, like industrial reliability engineers, in many respects. For instance, it can be used to certify the results estimated by the design engineers or provide essential feedback at the design stage to correct this estimated result, which are traditionally either obtained through manual manipulation or computer simulation.

Acknowledgements

This publication was made possible by NPRP Grant # [5-813-1-134] from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the author[s].

References

[1] A. Abd-Allah, Extending reliability block diagrams to software architectures, Technical report USC-CSE-97-501, Dept. of Computer Science, Univ. Southern California, USA, 1997.
[2] W. Ahmad, Formalization of reliability block diagrams, proof script, http://save.seecs.nust.edu.pk/projects/rbd/rbd, 2016.
[3] W. Ahmed, O. Hasan, S. Tahar, M.S. Hamdi, Towards the formal reliability analysis of oil and gas pipelines, in: Intelligent Computer Mathematics, in: Lect. Notes Comput. Sci., vol. 8543, Springer, 2014, pp. 30–44.
[4] ASENT, https://www.raytheoneagle.com/asent/rbd.htm, 2016.
[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, ACM SIGOPS Oper. Syst. Rev. 37 (5) (2003) 164–177.
[6] R. Bilintion, R. Allan, Reliability Evaluation of Engineering System, Springer, 1992.
[7] H.D. Boyd, C.A. Locurto, Reliability and maintainability for fire protection systems, in: Fire Safety Science, IAFSS, 1986, pp. 963–970.
[8] C. Brown, Automated reasoning in Higher-Order Logic, College Publications (2007).
[9] A. Church, A formulation of the simple theory of types, J. Symb. Log. 5 (1940) 56–68.
[10] M. Fitting, First-Order Logic and Automated Theorem Proving, Springer, 1996.
[11] M. Gordon, Mechanizing programming logics in higher-order logic, in: Current Trends in Hardware Verification and Automated Theorem Proving, Springer, 1989, pp. 387–439.
[12] J. Harrison, Formalized mathematics, Technical report 36, Turku Centre for Computer Science, 1996.
[13] D.L. Huffman, F. Antelme, Availability analysis of a solar power system with graceful degradation, in: Reliability and Maintainability Symposium, IEEE, 2009, pp. 348–352.
[14] J. Klion, Practical Electronic Reliability Engineering: Getting the Job Done from Requirement Through Acceptance, Springer Science & Business Media, 2012.
[15] S.R. Ko, S. Lee, Cloud computing vulnerability incidents: a statistical overview, https://cloudsecurityalliance.org/download/cloudcomputingvulnerabilityincidentsastatisticaloverview/, 2013.
[16] K. Kolowrocki, Reliability and risk analysis of multi-state systems with degrading components, Electr. J. Int. Group Reliab. 2 (1) (2009) 86–104.
[17] Y. Li, H. Yi, Research on the inherent reliability and the operational reliability of the supply chain, u- and e-service, Sci. Technol. 7 (1) (2014) 104–112.

[18] C.-M. Lin, H.-K. Teng, C.-C. Yang, H.-L. Weng, M.-C. Chung, C.-C. Chung, A mesh network reliability analysis using reliability block diagram, in: Industrial Informatics, IEEE, 2010, pp. 975–979.
[19] T. Mhamdi, O. Hasan, S. Tahar, On the formalization of the lebesgue integration theory in HOL, in: Interactive Theorem Proving, in: Lect. Notes Comput. Sci., vol. 6172, Springer, 2011, pp. 387–402.
[20] R. Milner, A theory of type polymorphism in programming, J. Comput. Syst. Sci. 17 (1977) 348–375.
[21] G. Norman, D. Parker, Quantitative verification: formal guarantees for timeliness, reliability and performance, 2014.
[22] PRISM, www.cs.bham.ac.uk/~dxp/prism, 2016.
[23] H.V. Ramasamy, M. Schunter, Architecting dependable systems using virtualization, in: Workshop on Architecting Dependable Systems, IEEE, 2007, pp. 1–6.
[24] ReliaSoft, http://www.reliasoft.com/, 2016.
[25] R. Robidoux, H. Xu, L. Xing, M. Zhou, Automated modeling of dynamic Reliability Block Diagrams using Colored Petri Nets, IEEE Trans. Syst. Man Cybern., Part A, Syst. Hum. 40 (2) (2010) 337–351.
[26] F.K. Shaikh, A. Khelil, N. Suri, On modeling the reliability of data transport in wireless sensor networks, in: Parallel, Distributed and Network-Based Processing, 2007, IEEE, 2007, pp. 395–402.
[27] K. Slind, M. Norrish, A brief overview of HOL4, in: Theorem Proving in Higher-Order Logics, in: Lect. Notes Comput. Sci., vol. 5170, Springer, 2008, pp. 28–32.
[28] The HOL System Logic, Kananaskis-10, https://sourceforge.net/projects/hol/files/hol/kananaskis-10/ (November 6, 2014).
[29] VMware infrastructure architecture overview, http://www.vmware.com/pdf/vi_architecture_wp.pdf, 2016.
[30] B. Wei, C. Lin, X. Kong, Dependability modeling and analysis for the virtual data center of cloud computing, in: High Performance Computing and Communications, IEEE, 2011, pp. 784–789.
[31] Z. Zhang, B. Shao, Reliability evaluation of different pipe section in different period, in: Service Operations and Logistics, and Informatics, IEEE, 2008, pp. 1779–1782.