Formal Probabilistic Analysis: A Higher-Order Logic Based Approach

Osman Hasan and Sofiène Tahar

Dept. of Electrical & Computer Engineering, Concordia University
1455 de Maisonneuve W., Montreal, Quebec, H3G 1M8, Canada
{o_hasan,tahar}@ece.concordia.ca

Abstract. Traditionally, simulation is used to perform probabilistic analysis. However, it provides less accurate results and cannot handle large-scale problems due to the enormous CPU time requirements. Recently, a significant amount of formalization has been done in higher-order logic that allows us to conduct precise probabilistic analysis using theorem proving and thus overcome the limitations of the simulation. Some major contributions include the formalization of both discrete and continuous random variables and the verification of some of their corresponding probabilistic and statistical properties. This paper describes the infrastructures behind these capabilities and their utilization to conduct the probabilistic analysis of real-world systems.

1 Introduction

"*In short, we can only pretend to achieve a relative faultless construction, not an absolute one, which is clearly impossible. A problem solution for which is still in its infancy is finding the right methodology to perform an environmental model that is a "good" approximation of the real environment. It is clear that a probabilistic approach would certainly be very useful for doing this.*"

J. Abrial, Faultless Systems: Yes We Can!, IEEE Computer Magazine, 42(9):30-36, 2009.

Probabilistic analysis is a tool of fundamental importance for the analysis of hardware and software systems. These systems usually exhibit some random or unpredictable elements. Examples include, failures due to environmental conditions or aging phenomena in hardware components and the execution of certain actions based on a probabilistic choice in randomized algorithms. Moreover, these systems act upon and within complex environments that themselves have certain elements of unpredictability, such as noise effects in hardware components and the unpredictable traffic pattern in the case of telecommunication protocols. Due to these random components, establishing the correctness of a system under all circumstances usually becomes impractically expensive. The engineering approach to analyze a system with these kind of unpredictable elements is to use probabilistic analysis. The main idea is to mathematically model the unpredictable elements of the given system and its environment by appropriate random variables. The probabilistic properties of these random variables are then used to judge system's behaviors regarding parameters of interest, such as downtime, availability, number of failures, capacity, and cost. Thus, instead of guaranteeing that the system meets some given specification under all circumstances, the probability that the system meets this specification is reported.

Simulation is the most commonly used computer based probabilistic analysis technique. Most simulation softwares provide a programming environment for defining functions that approximate random variables for probability distributions. The random elements in a given system are modeled by these functions and the system is analyzed using computer simulation techniques, such as the Monte Carlo Method [31], where the main idea is to approximately answer a query on a probability distribution by analyzing a large number of samples. Statistical quantities, such as average and variance, may then be calculated, based on the data collected during the sampling process, using their mathematical relations in a computer. Due to the inherent nature of simulation, the probabilistic analysis results attained by this technique can never be termed as 100% accurate. The accuracy of the hardware and software system analysis results has become imperative these days because of the extensive usage of these systems in safety-critical areas, such as, medicine and transportation. Therefore, simulation cannot be relied upon for the analysis of such systems.

In order to overcome the above mentioned limitations, we propose to use higher-order-logic theorem proving for probabilistic analysis. Higher-order logic [11] is a system of deduction with a precise semantics and is expressive enough to be used for the specification of almost all classical mathematics theories. Due to its high expressive nature, higher-order-logic can be utilized to precisely model the behavior of any system, while expressing its random or unpredictable elements in terms of formalized random variables, and any kind of system property, including the probabilistic and statistical ones, as long as they can be expressed in a closed mathematical form. Interactive theorem proving [16] is the field of computer science and mathematical logic concerned with precise computer based formal proof tools that require some sort of human assistance. Due to its interactive nature, interactive theorem proving can be utilized to reason about the correctness of probabilistic or statistical properties of systems, which are usually undecidable.

In this paper, we present a higher-order-logic theorem proving based framework that can be utilized to conduct formal probabilistic analysis of systems. We provide a brief overview of higher-order-logic formalizations that facilitate the formal modeling of random systems [18,19,26] and formal reasoning about their probabilistic and statistical properties [17,20,21]. We show how these capabilities fit into the overall formal probabilistic analysis framework and also point out some of the missing links that need further investigations. For illustration purposes, we discuss the formal probabilistic analysis of some real-world systems from the areas of telecommunications, nanoelectronics and computational algorithms.

The rest of the paper is organized as follows: Section 2 describes the proposed probabilistic analysis framework and how the already formalized mathematical concepts of probability theory fit into it. The case studies are presented in Section 3. Section 4 summarizes the state-of-the-art in the formal probabilistic analysis domain and compares these approaches with higher-order-logic theorem proving based analysis. Finally, Section 5 concludes the paper.

2 Formal Probabilistic Analysis Framework

A hypothetical model of a higher-order-logic theorem proving based probabilistic analysis framework is given in Fig. 1, with some of its most fundamental components depicted with shaded boxes. The starting point of probabilistic analysis is a system description and some intended system properties and the goal is to check if the given system satisfies these given properties. Due to the differences in the underlying mathematical foundations of discrete and continuous random variables [42], we have divided system properties into two categories, i.e., system properties related to discrete random variables and system properties related to continuous random variables.

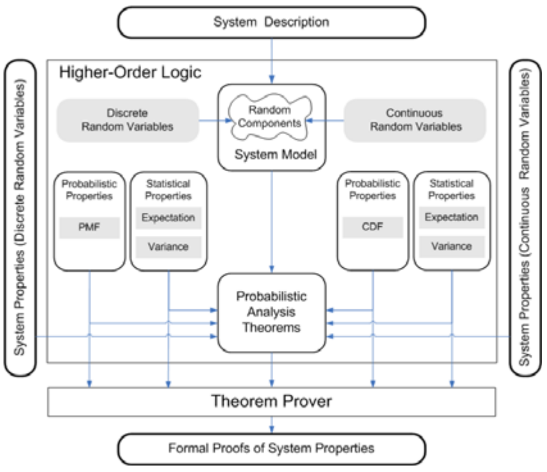

Fig. 1. Higher-order Logic based Probabilistic Analysis Framework

The first step in the proposed approach is to construct a model of the given system in higher-order-logic. For this purpose, the foremost requirement is the availability of infrastructures that allow us to formalize all kinds of discrete and continuous random variables as higher-order-logic functions, which in turn can be used to represent the random components of the given system in its higher-order-logic model. The second step is to utilize the formal model of the system to express system properties as higher-order-logic theorems. The prerequisite for this step is the ability to express probabilistic and statistical properties related to both discrete and continuous random variables in higher-order-logic. All probabilistic properties of discrete and continuous random variables can be expressed in terms of their *Probability Mass Function* (PMF) and *Cumulative Distribution Function* (CDF), respectively. Similarly, most of the commonly used statistical properties can be expressed in terms of the expectation and variance characteristics of the corresponding random variable. Thus, we require the formalization of mathematical definitions of PMF, CDF, expectation and variance for both discrete and continuous random variables in order to be able to express the given system's reliability characteristics as higher-order-logic theorems. The third and the final step for conducting probabilistic analysis in a theorem prover is to formally verify the higher-order-logic theorems developed in the previous step using a theorem prover. For this verification, it would be quite handy to have access to a library of some pre-verified theorems corresponding to some commonly used properties regarding probability distribution functions, expectation and variance. Since, we can build upon such a library of theorems and thus speed up the verification process. The formalization details regarding the above mentioned steps are briefly described now.

2.1 Discrete Random Variables and the PMF

A random variable is called discrete if its range, i.e., the set of values that it can attain, is finite or at most countably infinite [42]. Discrete random variables can be completely characterized by their PMFs that return the probability that a random variable X is equal to some value x, i.e., $Pr(X = x)$. Discrete random variables are quite frequently used to model randomness in probabilistic analysis. For example, the Bernoulli random variable is widely used to model the fault occurrence in a component and the Binomial random variable may be used to represent the number of faulty components in a lot.

Discrete random variables can be formalized in higher-order-logic as deterministic functions with access to an infinite Boolean sequence B^∞; an infinite source of random bits with data type ($natural \rightarrow bool$) [26]. These deterministic functions make random choices based on the result of popping bits in the infinite Boolean sequence and may pop as many random bits as they need for their computation. When the functions terminate, they return the result along with the remaining portion of the infinite Boolean sequence to be used by other functions. Thus, a random variable that takes a parameter of type α and ranges over values of type β can be represented by the function

$$\mathcal{F} : \alpha \rightarrow B^\infty \rightarrow (\beta \times B^\infty)$$

For example, a $Bernoulli(\frac{1}{2})$ random variable that returns 1 or 0 with probability $\frac{1}{2}$ can be modeled as

```
⊢ bit = λs. (if shd s then 1 else 0, stl s)
```

where the variable s represents the infinite Boolean sequence and the functions `shd` and `stl` are the sequence equivalents of the list operations *'head'* and *'tail'*. A function of the form λ`x.t` represents a lambda abstraction function that maps x to $t(x)$. The function `bit` accepts the infinite Boolean sequence and returns a pair with the first element equal to either 0 or 1 and the second element equal to the unused portion of the infinite Boolean sequence.

The higher-order-logic formalization of probability theory [26] also consists of a probability function $\mathbb{P}$ from sets of infinite Boolean sequences to *real* numbers between 0 and 1. The domain of $\mathbb{P}$ is the set $\mathcal{E}$ of events of the probability. Both $\mathbb{P}$ and $\mathcal{E}$ are defined using the Carathéodory's Extension theorem, which ensures that $\mathcal{E}$ is a σ-algebra: closed under complements and countable unions. The formalized $\mathbb{P}$ and $\mathcal{E}$ can be used to formally verify all basic axioms of probability. Similarly, they can also be used to prove probabilistic properties for random variables. For example, we can formally verify the following probabilistic property for the function `bit`, defined above,

$$\vdash \mathbb{P}\ \{\texttt{s} \mid \texttt{fst (bit s) = 1}\} = \tfrac{1}{2}$$

where the function `fst` selects the first component of a pair and $\{x|C(x)\}$ represents a set of all elements x that satisfy the condition C.

The above mentioned infrastructure can be utilized to formalize most of the commonly used discrete random variables and verify their corresponding PMF relations [26]. For example, the formalization and verification of Bernoulli and Uniform random variables can be found in [26] and of Binomial and Geometric random variables can be found in [21].

2.2 Continuous Random Variables and the CDF

A random variable is called continuous if it ranges over a continuous set of numbers that contains all real numbers between two limits [42]. Continuous random variables can be completely characterized by their CDFs that return the probability that a random variable X is exactly less than or equal to some value x, i.e., $Pr(X \leq x)$. Examples of continuous random variables include measuring the arrival time T of a data packet at a web server ($S_T = \{t|0 \leq t < \infty\}$) and measuring the voltage V across a resistor ($S_V = \{v| - \infty < v < \infty\}$).

The sampling algorithms for continuous random variables are non-terminating and hence require a different formalization approach than discrete random variables, for which the sampling algorithms are either guaranteed to terminate or satisfy probabilistic termination, meaning that the probability that the algorithm terminates is 1. One approach to address this issue is to utilize the concept of the nonuniform random number generation [9], which is the process of obtaining arbitrary continuous random numbers using a Standard Uniform random number generator. The main advantage of this approach is that we only need to formalize the Standard Uniform random variable from scratch and use it to model other continuous random variables by formalizing the corresponding nonuniform random number generation method.

Based on the above approach, a methodology for the formalization of all continuous random variables for which the inverse of the CDF can be represented in a closed mathematical form is presented in [18]. The first step in this methodology is the formalization of the Standard Uniform random variable, which can be done by using the formalization approach for discrete random variables and the formalization of the mathematical concept of limit of a *real* sequence [15]:

$$\lim_{n \to \infty} (\lambda n. \sum_{k=0}^{n-1} (\frac{1}{2})^{k+1} X_k) \tag{1}$$

where X_k denotes the outcome of the k^{th} random bit; $True$ or $False$ represented as 1 or 0, respectively. The formalization details are outlined in [19].

The second step in the methodology for the formalization of continuous probability distributions is the formalization of the CDF and the verification of its classical properties. This is followed by the formal specification of the mathematical concept of the inverse function of a CDF. This definition along with the formalization of the Standard Uniform random variable and the CDF properties, can be used to formally verify the correctness of the Inverse Transform Method (ITM) [9]. The ITM is a well known nonuniform random generation technique for generating nonuniform random variables for continuous probability distributions for which the inverse of the CDF can be represented in a closed mathematical form. Formally, it can be verified for a random variable X with CDF F using the Standard Uniform random variable U as follows [18].

$$Pr(F^{-1}(U) \leq x) = F(x) \tag{2}$$

The formalized Standard Uniform random variable can now be used to formally specify any continuous random variable for which the inverse of the CDF can be expressed in a closed mathematical form as $X = F^{-1}(U)$. Whereas, the formally verified ITM, given in Equation (2), can be used to prove the CDF for such a formally specified random variable. This approach has been successfully utilized to formalize and verify Exponential, Uniform, Rayleigh and Triangular random variables [18].

2.3 Statistical Properties for Discrete Random Variables

In probabilistic analysis, statistical characteristics play a major role in decision making as they tend to summarize the probability distribution characteristics of a random variable in a single number. Due to their widespread interest, the computation of statistical characteristics has now become one of the core components of every contemporary probabilistic analysis framework.

The expectation for a function of a discrete random variable, which attains values in the positive integers only, is defined as follows [30]

$$Ex\_fn[f(X)] = \sum_{n=0}^{\infty} f(n) Pr(X = n) \tag{3}$$

where X is the discrete random variable and f represents a function of X. The above definition only holds if the associated summation is convergent, i.e., $\sum_{n=0}^{\infty} f(n)Pr(X = n) < \infty$. The expression of expectation, given in Equation (3), has been formalized in [20] as a higher-order-logic function using the probability function $\mathbb{P}$. The expected value of a discrete random variable that attains values in positive integers can now be defined as a special case of Equation (3)

$$Ex[X] = Ex\_fn[(\lambda n.n)(X)] \tag{4}$$

when f is an identity function. In order to verify the correctness of the above definitions of expectation, they are utilized in [20,21] to formally verify the following classical expectation properties.

$$Ex[\sum_{i=1}^{n} R_i] = \sum_{i=1}^{n} Ex[R_i] \tag{5}$$

$$Ex[a + bR] = a + bEx[R] \tag{6}$$

$$Pr(X \geq a) \leq \frac{Ex[X]}{a} \tag{7}$$

These properties not only verify the correctness of the above definitions but also play a vital role in verifying the expectation characteristics of discrete random components of probabilistic systems, as will be seen in Section 3 of this paper.

Variance of a random variable X describes the difference between X and its expected value and thus is a measure of its dispersion.

$$Var[X] = Ex[(X - Ex[X])^2] \tag{8}$$

The above definition of variance has been formalized in higher-order-logic in [20] by utilizing the formal definitions of expectation, given in Equations (3) and (4). This definition is then formally verified to be correct by proving the following classical variance properties for it [20,21].

$$Var[R] = Ex[R^2] - (Ex[R])^2 \tag{9}$$

$$Var[\sum_{i=1}^{n} R_i] = \sum_{i=1}^{n} Var[R_i] \tag{10}$$

$$Pr(|X - Ex[X]| \geq a) \leq \frac{Var[X]}{a^2} \tag{11}$$

These results allow us to reason about expectation, variance and tail distribution properties of any formalized discrete random variable that attains values in positive integers, e.g., the formal verification for Bernoulli, Uniform, Binomial and Geometric random variables is presented in [21].

2.4 Statistical Properties for Continuous Random Variables

The most commonly used definition of expectation, for a continuous random variable X, is the probability density-weighted integral over the real line [34].

$$E[X] = \int_{-\infty}^{+\infty} xf(x)dx \tag{12}$$

The function f in the above equation represents the *Probability Density Function* (PDF) of X and the integral is the well-known Reimann integral. The above definition is limited to continuous random variables that have a well-defined PDF. A more general, but not so commonly used, definition of expectation for a random variable X, defined on a probability space (Ω, Σ, P) [10], is as follows.

$$E[X] = \int_{\Omega} XdP \tag{13}$$

This definition utilizes the Lebesgue integral and is general enough to cater for both discrete and continuous random variables. The reason behind its limited usage in the probabilistic analysis domain is the complexity of solving the Lebesgue integral, which takes its foundations from the measure theory that most engineers and computer scientists are not familiar with.

The obvious advantage of using Equation (12) for formalizing expectation of a continuous random variable is the user familiarity with Reimann integral that usually facilitates the reasoning process regarding the expectation properties in the theorem proving based probabilistic analysis approach. On the other hand, it requires extended real numbers, $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$, whereas all the foundational work regarding theorem proving based probabilistic analysis, outlined above, has been built upon the standard real numbers $\mathbb{R}$, formalized by Harrison [15]. The expectation definition given in Equation (13) does not involve extended real numbers, as it accommodates infinite limits without any ad-hoc devices due to the inherent nature of the Lebesgue integral. It also offers a more general solution. The limitation, however, is the compromise on the interactive reasoning effort, as it is not a straightforward task for a user to build on this definition to formally verify the expectation of a random variable.

We have formalized the expectation of a continuous random variable as in Equation (13) by building on top of a higher-order-logic formalization of Lebesgue integration theory [6]. Starting from this definition, two simplified expressions for the expectation are verified that allow us to reason about expectation of a continuous random variable in terms of simple arithmetic operations [17]. The first expression is for the case when the given continuous random variable X is bounded in the positive interval $[a, b]$.

$$E[X] = \lim_{n\to\infty} \left[\sum_{i=0}^{2^n-1} (a + \frac{i}{2^n}(b-a)) P \left\{ a + \frac{i}{2^n}(b-a) \le X < a + \frac{i+1}{2^n}(b-a) \right\} \right] \tag{14}$$

The second expression is for an unbounded positive random variable [10].

$$E[X] = \lim_{n\to\infty} \left[\sum_{i=0}^{n2^n-1} \frac{i}{2^n} P\left\{ \frac{i}{2^n} \leq X < \frac{i+1}{2^n} \right\} + nP(X \geq n) \right] \tag{15}$$

Both of the above expressions do not involve any concepts from Lebesgue integration theory and are based on the well-known arithmetic operations like summation, limit of a real sequence, etc. Thus, users can simply utilize them, instead of Equation (13), to reason about the expectation properties of their random variables and gain the benefits of the original Lebesgue based definition. The formal verification details for these expressions are given in [17]. These expressions are further utilized to verify the expected values of Uniform, Triangular and Exponential random variables [17]. The above mentioned definition and simplified expressions will also facilitate the formalization of variance and the verification of its corresponding properties.

3 Applications

We now illustrate the usage of the above mentioned formalization, for conducting probabilistic analysis of some real-world systems.

3.1 Probabilistic Analysis of the Coupon Collector's Problem

The Coupon Collector's problem [34] refers to the problem of probabilistically evaluating the number of trials required to acquire all unique, say n, coupons from a collection of multiple copies of these coupons that are independently and uniformly distributed. The problem is similar to the example when each box of cereal contains one of n different coupons and once you obtain one of every type of coupon, you win a prize. The Coupon Collector's problem is a commercially used computational problem and is commonly used for the identification of routers that are encountered in packet communication between two hosts [34].

Based on the probabilistic analysis framework, presented in Section 2, particularly the capabilities to formally specify discrete random variables and formally reason about the statistical properties of systems, a formal probabilistic analysis of the Coupon Collector's problem is presented in [21]. The first goal is to verify that the expected value of acquiring all n coupons is $nH(n)$, where $H(n)$ is the *harmonic number* ($\sum_{i=1}^{n} 1/i$). Based on this expectation value, the next step is to reason about the tail distribution properties of the Coupon Collector's problem using the formally verified Markov's and Chebyshev's inequalities.

The first step in the proposed approach is to model the behavior of the given system as a higher-order-logic function, while representing its random component using the formalized random variables. The Coupon Collector's problem can be formalized by modeling the total number of trials required to obtain all n unique coupons, say T, as a sum of the number of trials required to obtain each distinct coupon, i.e., $T = \sum_{i=1}^{n} T_i$, where T_i represents the number of trials to obtain the i^{th} coupon, while $i-1$ distinct coupons have already been acquired. The advantage of breaking the random variable T into the sum of n random variables $T_1, T_2 \cdots, T_n$ is that each T_i can be modeled by the Geometric random variable function. It is important to note here that the probability of success for these Geometric random variables would be different from one another and would be equal to the probability of finding a new coupon while conducting uniform selection trials on the available n coupons. Thus, the success probability depends on the number of already acquired coupons and can be modeled using the higher-order-logic function for the discrete Uniform random variable. Using this approach the Coupon Collector's problem has been modeled in [21] as a higher-order-logic function, `coupon_collector`, that accepts a positive integer greater than 0, $n+1$, which represents the total number of distinct coupons that are required to be collected. The function returns the number of trials for acquiring these $n+1$ distinct coupons. Now, using this function along with the formal definitions of expectation and variance and their formally verified corresponding properties, given in Section 2.3, the following statistical characteristics can be verified [21].

$$\vdash \forall \texttt{ n. expec (coupon\_collector (n + 1)) = (n + 1) } (\textstyle\sum_{i=0}^{n+1} \frac{1}{i+1})$$

$$\vdash \forall \texttt{ n a. 0 < a} \Rightarrow \mathbb{P}\ \{\texttt{s | (fst(coupon\_collector (n + 1) s))} \geq \texttt{a}\} \leq (\frac{(n+1)}{a}\ (\textstyle\sum_{i=0}^{n+1} \frac{1}{(i+1)}))$$

$$\vdash \forall \texttt{ n a. 0 < a} \Rightarrow \mathbb{P}\ \{\texttt{s | abs((fst(coupon\_collector (n + 1) s)) - expec (coupon\_collector (n + 1)))} \geq \texttt{a}\} \leq (\frac{(n+1)^2}{a^2}\ (\textstyle\sum_{i=0}^{n+1} \frac{1}{(i+1)^2}))$$

where `expec` and `abs` represent the higher-order-logic functions for expectation and absolute functions, respectively.

The first theorem gives the expectation of the Coupon Collector's problem, while the next two correspond to the tail distribution bounds of the Coupon Collector's problem using Markov and Chebyshev's inequalities, respectively. The above results exactly match the results of the analysis based on paper-and-pencil proof techniques [34] and are thus 100 % precise, which is a novelty that cannot be achieved, to the best of our knowledge, by any existing computer based probabilistic analysis tool. The results were obtained by building on top of the formally verified linearity of expectation and variance properties and the Markov and Chebyshev's inequalities and thus the proof script corresponding to the formalization and verification of the Coupon Collector's problem translated to approximately 1000 lines of code and the analysis took around 100 man-hours.

3.2 Performance Analysis of the Stop-and-Wait Protocol

The Stop-and-Wait protocol [29] utilizes the principles of error detection and retransmission to ensure reliable communication between computers. The main idea is that the transmitter keeps on transmitting a data packet unless and until it receives a valid acknowledgement (ACK) of its reception from the receiver.

The message delay of a communication protocol is the most widely used performance metric. In the case of the Stop-and-Wait protocol, the message delay is an unpredictable quantity since it depends on the random behavior of channel noise and thus probabilistic techniques are utilized for its assessment.

The Stop-and-Wait protocol is a classical example of a real-time system and thus involves a subtle interaction of a number of distributed processes. The behavior of these processes over time may be specified by higher-order-logic predicates on positive integers [5] that represent the ticks of a clock counting physical time in any appropriate units, e.g., nanoseconds. The granularity of the clock's tick is believed to be chosen in such a way that it is sufficiently fine to detect properties of interest. Using this approach, the Stop-and-Wait protocol can be formalized in higher-order logic as a logical conjunction of six processes (Data Transmission, Data Channel, Data Reception, ACK Transmission, ACK Channel, ACK Reception) and some initial conditions [22]. The random component in the Stop-and-Wait protocol is channel noise, which can be expressed using the formal Bernoulli random variable function.

The next step is to utilize the formal model of the Stop-and-Wait protocol to formally verify the average message delay relation of the Stop-and-Wait protocol, for the case when the processing time of a message is equal to 1, as the following theorem [22].

```
⊢ ∀ source sink rem s i r ws sn ackty maxP abort dataS dataR
  ackS ackR d tprop dtout dtf dta tf ack_msg ta tout rec_flag
  bseqt bseq p.
    STOP_WAIT_NOISY source sink rem s i r ws sn ackty maxP abort
    dataS dataR ackS ackR d tprop dtout dtf dta tf
    ack_msg ta tout rec_flag bseqt bseq ∧
    LIVE_ASSUMPTION abort ∧ 0 ≤ p ∧ p < 1 ∧ ¬NULL source ∧
     tprop + 1 + ta + tprop + 1 ≤ tout ⇒
     (expec (DELAY_STOP_WAIT_NOISY rem source bseqt) =
       ((tf + tout)p/(1-p) + (tf + tprop + 1 + ta + tprop + 1)))
```

The antecedent of the above theorem contains the formal definition of the Stop-and-Wait protocol under noisy channel conditions (`STOP_WAIT_NOISY`), liveness constraints and the fact that the probability of channel error p is bounded in the real interval $[0, 1)$. The function `DELAY_STOP_WAIT_NOISY` formally represents the delay of the Stop-and-Wait protocol and thus the left-hand-side of the conclusion of the above theorem represents the average delay of the Stop-and-Wait protocol. On the right-hand-side of the conclusion of the above theorem, the variables tf, ta, $tprop$ and $tout$ denote the time delays associated with data transmission, ACK transmission, message propagation, message processing and time-out delays, respectively. More details on the variables used above and the proof sketch of this theorem can be found in [22].

It is important to note here that the relation for the average delay of a Stop-and-Wait protocol is not new. In fact its existence dates back to the early days of introduction of the Stop-and-Wait protocol. However, it has always been verified using theoretical paper-and-pencil proof techniques, e.g. [29]. Whereas, the analysis described above is based on mechanical verification using a theorem prover, which is a superior approach to both paper-and-pencil proofs and simulation based analysis techniques. To the best of our knowledge, it is the first time that a statistical property for a real-time system has been been formally verified.

3.3 Reliability Analysis of Reconfigurable Memory Arrays

Reconfigurable memory arrays with spare rows and columns are quite frequently used as reliable data storage components in present age System-on-Chips. The spare memory rows and columns can be utilized to automatically replace rows or columns that are found to contain a cell fault, such as stuck-at or coupling fault [33]. One of the biggest design challenges is to estimate, prior to the actual fabrication process, the right number of these spare rows and spare columns for meeting the reliability specifications. Since the fault occurrence in a memory cell is an unpredictable event, probabilistic techniques are utilized to estimate the number of spare rows and columns [39].

The analysis for this example is done by formally expressing a fault model for reconfigurable memory arrays in higher-order logic [23]. The formalization utilizes the precise Binomial random variable function to express the random components in the model. This model is then utilized to express and verify statistical properties, such as expectation and variance of the number of faults in terms of memory array and spare rows and columns sizes, as higher-order logic theorems. Finally, this formal statistical information is built upon to formally verify repairability and irrepairability conditions for a square memory array with stuck-at and coupling faults that are independent and identically distributed. For example, the repairability condition for a square nxn memory array, with axn spare rows and bxn spare columns, has been verified as the following higher-order-logic theorem.

$$\begin{array}{l}\vdash \forall\ \texttt{a b w.}\ (0 \leq \texttt{a}) \wedge (\texttt{a} \leq 1) \wedge (0 \leq \texttt{b}) \wedge (\texttt{b} \leq 1) \wedge \\ \quad (\texttt{c1 + c2 = a + b}) \wedge (1 < \texttt{n}) \wedge (\forall\ \texttt{n.}(0 < \texttt{w(n)}) \wedge \\ \quad\quad (\texttt{w(n)} < (\texttt{min c1}\sqrt{\texttt{n}}\ \texttt{c2}\sqrt{\texttt{n}}))) \wedge (\texttt{lim}\ (\lambda \texttt{n.}\ \frac{1}{\texttt{w(n)}}) = 0) \Rightarrow \\ (\texttt{lim}\ (\lambda \texttt{n.}\mathbb{P}\{\texttt{s}\ |(\texttt{fst(num\_of\_faults n c1 c2 w s)}) \leq \texttt{(a+b)n}\}) = 1))\end{array}$$

where `lim M` represents the higher-order logic formalization of the limit of a real sequence M (i.e., $lim\ M = \lim_{n \to \infty} M(n)$) [15]. The first four assumptions in the above theorem ensure that the fractions a and b are bounded by the interval $[0, 1]$ as the number of spares can never exceed the number of original rows. The relationship between a and b with two arbitrary real numbers $c1$ and $c2$ is given in the fifth assumption. The precondition $1 < n$ has been used in order to ensure that the given memory array has more than one cell. The next two assumptions are about the real sequence w and basically provides its upper and lower bounds. These bounds have been used in order to prevent the stuck-at and coupling fault occurrence probabilities p_s and p_c from falling outside their allowed interval $[0, 1]$ [23]. The last assumption $(\texttt{lim}(\lambda \texttt{n.}\frac{1}{\texttt{w(n)}}) = 0)$ has been added to formally represent the intrinsic characteristic of *real* sequence w that it tends to infinity as its *natural* argument becomes very very large. The theorem proves that under these assumptions a very large square memory array is almost always repairable (with probability 1) since the probability that the number of faults is less than the number of spare rows and columns is 1.

The above theorem leads to the accurate estimation of the number of spare rows and columns required for reliable operation against stuck-at and coupling faults of any reconfigurable memory array without any CPU time constraints. The distinguishing feature of this analysis is its generic nature as our theorems are verified for all sizes of memories nxn with any number of spare rows (axn) or columns (bxn).

This case study clearly demonstrate the effectiveness of theorem proving based probabilistic analysis. Due to the formal nature of the models, the high expressiveness of higher-order logic, and the inherent soundness of theorem proving, we have been able to verify generic properties of interest that are valid for any given memory array with 100% precision; a novelty which is not available in simulation. Similarly, we have been able to formally analyze properties that cannot be handled by model checking. The proposed approach is also superior to the paper-and-pencil proof methods [39] in a way as the chances of making human errors, missing critical assumptions and proving wrongful statements are almost nil since all proof steps are applied within the sound core of a higher-order-logic theorem prover. These additional benefits come at the cost of the time and effort spent, while formalizing the memory array and formally reasoning about its properties. But, the fact that we were building on top of already verified probability theory foundations, described in Section 2, helped significantly in this regard as the memory analysis only consumed approximately 250 man-hours and 3500 lines of proof code.

3.4 Round-Off Error Analysis in Floating-Point Representation

Algorithms involving floating-point numbers are extensively used these days in almost all digital equipment ranging from computer and digital processing to telecommunication systems. Due to their complexity and wide spread usage in safety critical domains, formal methods are generally preferred over traditional testing to ensure correctness of floating-point algorithms. A classical work in this regard is Harrison's error analysis of floating-point arithmetic in higher-order logic [14]. Harrison presents a formalization of floating point numbers, verification of upper bounds on the error in representing a real number in floating-point and the error in floating-point arithmetic operations. Even though this analysis is very useful in identifying the worst case conditions, it doest not reflect upon typical or average errors. In fact, the assumed worst case conditions rarely occur in practice. So the error analysis, based under these worst-case conditions can improperly suggest that the performance of the algorithm is poor.

In paper-and-pencil analyses, probabilistic techniques are thus utilized in the error analysis of floating-point algorithms [41]. The main idea behind this probabilistic approach is to model the error in a single floating-point number by an appropriate random variable and utilize this information to judge the expected value of error while representing a real number in floating-point system. This expected value of error can then be used to find the expected value of error in different floating-point arithmetic operations.

The above mentioned probabilistic analysis involves reasoning about the expectation value of a continuous random variable, since the error between a real number and its corresponding floating-point representation is continuous in nature. Thus, our proposed infrastructure can be directly utilized to conduct such analysis, something that to the best of our knowledge was not possible before.

We built upon Harrison's error bounds for floating-point representations of *big* ($|x| \in [2^k, 2^{k+1})$, *small* ($|x| \in [\frac{1}{2^{k+1}}, \frac{1}{2^k}] : k < 126$), and *tiny* ($|x| \in [0, \frac{1}{2^{126}}]$) real numbers [14]. The error is defined as the difference between the real value of the floating-point representation and the actual value of the corresponding real number ($\mathtt{error(x)} = \mathtt{float(x)} - \mathtt{x}$), with round-to-nearest rounding mode. Based on this definition, upper bounds on the absolute value of error are verified to be equal to $\frac{2^k}{2^{24}}$, $\frac{1}{2^{k+1}2^{24}}$ and $\frac{1}{2^{150}}$, for the three cases above, respectively.

Assuming any value of error to be equally likely [41], we constructed formal probabilistic models for representing the above mentioned rounding errors using Uniform random variables defined in the intervals $[0, \frac{2^k}{2^{24}}]$, $[0, \frac{1}{2^{k+1}2^{24}}]$ and $[0, \frac{1}{2^{150}}]$, respectively. The formally verified expectation of the Uniform random variable [17] was then used to verify the expectation values of these floating-point errors using a theorem prover.

$$\begin{aligned}\vdash \forall\ \mathtt{k\ x.}\ &\Big(\mathtt{expec(uniform\_rv\ 0\ }\tfrac{2^k}{2^{24}}\mathtt{)\ =\ }\tfrac{2^{k-1}}{2^{24}}\Big) \wedge\\ &\Big(\mathtt{expec(uniform\_rv\ 0\ }\tfrac{1}{2^{k+1}2^{24}}\mathtt{)\ =\ }\tfrac{1}{2^{k+1}2^{25}}\Big) \wedge\\ &\Big(\mathtt{expec(uniform\_rv\ 0\ }\tfrac{1}{2^{150}}\mathtt{)\ =\ }\tfrac{1}{2^{151}}\Big)\end{aligned}$$

This theorem plays a vital role in the statistical error analysis of floating-point arithmetic. Based on these averages of error in a single floating-point number, the average errors in floating point operations, like addition and multiplication, that involve multiple floating-point numbers, can be evaluated. Similarly, this information can be utilized in conducting the statistical error analysis of digital signal processing (DSP) systems by building on top of the DSP verification framework in higher-order logic [1], which does not include any probabilistic considerations.

The verification of the above result was automatic as the verified theorem is a direct consequence of the expectation property of the continuous Uniform random variable, which is available in the proposed framework. This fact clearly demonstrates the usefulness of the proposed infrastructure that calls for formalizing and verifying the fundamental concepts of probability theory in order to facilitate the formal probabilistic analysis of real-world systems.

4 Related Work

Due to the vast application domain of probability in safety-critical applications, many researchers around the world are trying to improve the quality of computer based probabilistic analysis. The ultimate goal is to come up with a formal probabilistic analysis framework that includes robust and accurate analysis methods, has the ability to perform analysis for large-scale problems and is easy to use. In this section, we provide a brief account of the state-of-the-art in this field.

Probabilistic model checking [3,37] is one of the commonly used formal probabilistic analysis technique. It involves the construction of a precise state-based mathematical model of the given probabilistic system, which is then subjected to exhaustive analysis to formally verify if it satisfies a set of formally represented probabilistic properties. Numerous probabilistic model checking algorithms and methodologies have been proposed in the open literature, e.g., [8,35], and based on these algorithms, a number of tools have been developed, e.g., PRISM [36,28], $E \vdash MC^2$ [24], Rapture [27] and VESTA [38]. Besides the accuracy of the results, the most promising feature of probabilistic model checking is the ability to perform the analysis automatically. On the other hand, it is limited to systems that can only be expressed as probabilistic finite state machines. Another major limitation of the probabilistic model checking approach is state space explosion [4]. The state space of a probabilistic system can be very large, or sometimes even infinite. Thus, at the outset, it is impossible to explore the entire state space with limited resources of time and memory. Similarly, we cannot reason about mathematical expressions in probabilistic model checking. This is a big limitation as far as reasoning about PMF, CDF or expectation or variance of a random behavior is concerned, which are basically functions of the range of a random variable. Thus, the probabilistic model checking approach, even though is capable of providing exact solutions, is quite limited in terms of handling a variety of probabilistic analysis problems. Whereas higher-order-logic theorem proving is capable of overcoming all the above mentioned problems but at a significant cost of user interaction.

Besides higher-order-logic theorem proving and model checking, another formal approach that is capable of providing exact solutions to probabilistic properties is proof based languages that have been extended with probabilistic choice. The main idea behind this approach is to use refinement or utilize the expectations (or probabilistic invariants) to reason about probabilistic properties. Many formalisms have been extended with probabilistic choice, e.g., B (pB) [25], Hoare logic (pL) [7], Z [40] and Event-B [13]. Besides their precision, another major benefit of these approaches is their automatic or semi-automatic nature. Out of these formalisms, Probabilistic B (pB) is one of the more commonly used mainly because of its ability to obtain algebraic relationships between the different parameters of the model and of the design requirements. On the other hand, even though some efforts have been reported, e.g. [2], it is not mature enough to model and reason about random components of the system that involve all different kinds of continuous probability distributions. Similarly, all of the above mentioned formalisms cannot be used to reason about generic mathematical expressions for probabilistic or statistical properties, such as PMF, CDF, expectation or variance, due to their limited expressiveness, which is not an issue with the proposed higher-order-logic theorem proving based approach. For example, the Chaums Dining Cryptographers (DC) problem, a well-known security problem, has been recently analyzed formally using the pB approach and the mean and variance of utterances have been computed for only a finite number of DC nets and specific set of fixed values for coin fairness [32]. By contrast, higher-order-logic theorem proving can be utilized to prove generic mathematical expressions, for the mean and variance characteristics of interest, that are quantified over n cryptographers and all values of coin fairness.

5 Conclusions

This paper provides a brief overview of the existing capabilities of higher-order-logic theorem proving based probabilistic analysis approach. The main idea behind this emerging trend is to use random variables formalized in higher-order logic to model systems, which contain some sort of randomness, and to verify the corresponding probabilistic and statistical properties in a theorem prover. Because of the formal nature of the models, the analysis is 100% accurate and due to the high expressive nature of higher-order logic a wider range of systems can be analyzed. Thus, the theorem proving based probabilistic analysis approach can prove to be very useful for the performance and reliability optimization of safety critical and highly sensitive engineering and scientific applications.

The proposed approach has been illustrated by providing the formal probabilistic analysis of four real-world systems. The analysis results exactly matched the results obtained by paper-and-pencil proof techniques and are thus 100 % precise. The successful handling of these diverse problems by the proposed approach clearly demonstrates its feasibility for real-world probabilistic analysis issues. In all these applications, we have been able to formally reason about real valued expressions of probabilistic or statistical properties of systems, something that cannot be achieved by probabilistic model checking or probabilistic language based approaches.

All higher-order-logic formalizations, presented in this paper, have been done using the HOL theorem prover [12]. The main reason being that the foundational measure and probability theories were formalized in HOL first [26] and then the rest of the infrastructure kept building upon that. Though, it is important to note that the presented methodologies and framework are not specific to the HOL theorem prover and can be adapted to any other higher-order-logic theorem prover, such as Isabelle, Coq or PVS, as well.

The theorem proving based probabilistic analysis framework can no way be considered to be mature enough to be able to handle all kind of problems. There are many open research issues that need to resolved in order to achieve this goal. To name a few, first of all the capability to reason about multiple continuous random variables is not available. Secondly, some of the most commonly used random variables, like the Normal random variable, have not been formalized so far. Thirdly, no formalization related to stochastic processes and Markov chains is available, which are widely used concepts in probabilistic analysis.

References

1. Akbarpour, B., Tahar, S.: An Approach for the Formal Verification of DSP Designs using Theorem Proving. IEEE Transactions on CAD of Integrated Circuits and Systems 25(8), 1141–1457 (2006)
2. Andrews, Z.: Towards a Stochastic Event B for Designing Dependable Systems. In: Proc. Workshop on Quantitative Formal Methods: Theory and Applications, Eindhoven, The Netherlands (November 2009)
3. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model Checking Algorithms for Continuous time Markov Chains. IEEE Transactions on Software Engineering 29(4), 524–541 (2003)
4. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press, Cambridge (2008)
5. Cardell-Oliver, R.: The Formal Verification of Hard Real-time Systems. PhD Thesis, University of Cambridge, UK (1992)
6. Coble, A.: Anonymity, Information, and Machine-Assisted Proof. Ph.D Thesis, University of Cambridge, UK (2009)
7. Corin, R.J., Den Hartog, J.I.: A Probabilistic Hoare-style Logic for Game-based Cryptographic Proofs. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 252–263. Springer, Heidelberg (2006)
8. de Alfaro, L.: Formal Verification of Probabilistic Systems. PhD Thesis, Stanford University, Stanford, USA (1997)
9. Devroye, L.: Non-Uniform Random Variate Generation. Springer, Heidelberg (1986)
10. Galambos, J.: Advanced Probability Theory. Marcel Dekker Inc., New York (1995)
11. Gordon, M.J.C.: Mechanizing Programming Logics in Higher-Order Logic. In: Current Trends in Hardware Verification and Automated Theorem Proving, pp. 387–439. Springer, Heidelberg (1989)
12. Gordon, M.J.C., Melham, T.F.: Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic. Cambridge University Press, Cambridge (1993)
13. Hallerstede, S., Hoang, T.S.: Qualitative Probabilistic Modelling in Event-B. In: Davies, J., Gibbons, J. (eds.) IFM 2007. LNCS, vol. 4591, pp. 293–312. Springer, Heidelberg (2007)
14. Harrison, J.: Floating Point Verification in HOL Light: The Exponential Function. Technical Report 428, Computing Laboratory, University of Cambridge, UK (1997)
15. Harrison, J.: Theorem Proving with the Real Numbers. Springer, Heidelberg (1998)
16. Harrison, J.: Handbook of Practical Logic and Automated Reasoning. Cambridge University Press, Cambridge (2009)
17. Hasan, O., Abbasi, N., Akbarpour, B., Tahar, S., Akbarpour, R.: Formal reasoning about expectation properties for continuous random variables. In: Cavalcanti, A., Dams, D.R. (eds.) FM 2009: Formal Methods. LNCS, vol. 5850, pp. 435–450. Springer, Heidelberg (2009)
18. Hasan, O., Tahar, S.: Formalization of the Continuous Probability Distributions. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 3–18. Springer, Heidelberg (2007)
19. Hasan, O., Tahar, S.: Formalization of the Standard Uniform Random Variable. Theoretical Computer Science 382(1), 71–83 (2007)
20. Hasan, O., Tahar, S.: Using Theorem Proving to Verify Expectation and Variance for Discrete Random Variables. Journal of Automated Reasoning 41(3-4), 295–323 (2008)

21. Hasan, O., Tahar, S.: Formal Verification of Tail Distribution Bounds in the HOL Theorem Prover. Mathematical Methods in the Applied Sciences 32(4), 480–504 (2009)
22. Hasan, O., Tahar, S.: Performance Analysis and Functional Verification of the Stop-and-Wait Protocol in HOL. Journal of Automated Reasoning 42(1), 1–33 (2009)
23. Hasan, O., Tahar, S., Abbasi, N.: Formal Reliability Analysis using Theorem Proving. IEEE Transactions on Computers (2009), doi:10.1109/TC.2009.165
24. Hermanns, H., Katoen, J.P., Meyer-Kayser, J., Siegle, M.: A Markov Chain Model Checker. In: Schwartzbach, M.I., Graf, S. (eds.) TACAS 2000. LNCS, vol. 1785, pp. 347–362. Springer, Heidelberg (2000)
25. Hoang, T.S.: The Development of a Probabilistic B Method and a Supporting Toolkit. PhD Thesis, The University of New South Wales, UK (2005)
26. Hurd, J.: Formal Verification of Probabilistic Algorithms. PhD Thesis, University of Cambridge, UK (2002)
27. Jeannet, B., Argenio, P.D., Larsen, K.: Rapture: A Tool for Verifying Markov Decision Processes. In: Tools Day, 13^{th} Int. Conf. Concurrency Theory, Brno, Czech Republic (2002)
28. Kwiatkowska, M., Norman, G., Parker, D.: Quantitative Analysis with the Probabilistic Model Checker PRISM. Electronic Notes in Theoretical Computer Science 153(2), 5–31 (2005)
29. Leon-Garcia, A., Widjaja, I.: Communication Networks: Fundamental Concepts and Key Architectures. McGraw-Hill, New York (2004)
30. Levine, A.: Theory of Probability. Addison-Wesley series in Behavioral Science, Quantitative Methods (1971)
31. MacKay, D.J.C.: Introduction to Monte Carlo Methods. In: Learning in Graphical Models, NATO Science Series, pp. 175–204. Kluwer Academic Press, Dordrecht (1998)
32. McIver, A., Meinicke, L., Morgan, C.: Security, Probability and Nearly Fair Coins in the Cryptographers' Café. In: Cavalcanti, A., Dams, D.R. (eds.) FM 2009: Formal Methods. LNCS, vol. 5850, pp. 41–71. Springer, Heidelberg (2009)
33. Miczo, A.: Digital Logic Testing and Simulation. Wiley Interscience, Hoboken (2003)
34. Mitzenmacher, M., Upfal, E.: Probability and Computing. Cambridge University Press, Cambridge (2005)
35. Parker, D.: Implementation of Symbolic Model Checking for Probabilistic System. PhD Thesis, University of Birmingham, UK (2001)
36. PRISM (2008), `http://www.cs.bham.ac.uk/~dxp/prism`
37. Rutten, J., Kwaiatkowska, M., Normal, G., Parker, D.: Mathematical Techniques for Analyzing Concurrent and Probabilisitc Systems. CRM Monograph Series, vol. 23. American Mathematical Society (2004)
38. Sen, K., Viswanathan, M., Agha, G.: VESTA: A Statistical Model-Checker and Analyzer for Probabilistic Systems. In: Proc. IEEE International Conference on the Quantitative Evaluation of Systems, pp. 251–252 (2005)
39. Shi, W., Fuchs, W.K.: Probabilistic Analysis and Algorithms for Reconfiguration of Memory Arrays. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 11(9), 1153–1160 (1992)
40. White, N.: Probabilistic Specification and Refinement. Masters Thesis, Oxford University, UK (1996)
41. Widrow, B.: Statistical Analysis of Amplitude-quantized Sampled Data Systems. AIEE Transactions on Applications and Industry 81, 555–568 (1961)
42. Yates, R.D., Goodman, D.J.: Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers. Wiley, Chichester (2005)