Formal Probabilistic Analysis using Theorem Proving

Osman Hasan

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montréal, Québec, Canada

April 2008

Your file Votre référence
ISBN: 978-0-494-37753-6
Our file Notre référence
ISBN: 978-0-494-37753-6

ABSTRACT

Formal Probabilistic Analysis using Theorem Proving

Osman Hasan, Ph.D.

Concordia University, 2008

Probabilistic analysis is a tool of fundamental importance to virtually all scientists and engineers as they often have to deal with systems that exhibit random or unpredictable elements. Traditionally, computer simulation techniques are used to perform probabilistic analysis. However, they provide less accurate results and cannot handle large-scale problems due to their enormous computer processing time requirements. To overcome these limitations, this thesis proposes to perform probabilistic analysis by formally specifying the behavior of random systems in higher-order logic and use these models for verifying the intended probabilistic and statistical properties in a computer based theorem prover. The analysis carried out in this way is free from any approximation or precision issues due to the mathematical nature of the models and the inherent soundness of the theorem proving approach.

The thesis mainly targets the two most essential components for this task, i.e., the higher-order-logic formalization of random variables and the ability to formally verify the probabilistic and statistical properties of these random variables within a theorem prover. We present a framework that can be used to formalize and verify any continuous random variable for which the inverse of the cumulative distribution function can be expressed in a closed mathematical form. Similarly, we provide a formalization infrastructure that allows us to formally reason about statistical properties, such as mean, variance and tail distribution bounds, for discrete random variables. In order to

illustrate the practical effectiveness of the proposed approach, we consider the probabilistic analysis of three examples: the Coupon Collector's problem, the roundoff error in a digital processor and the Stop-and-Wait protocol. All the above mentioned work is conducted using the HOL theorem prover.

To My Grand Father:

Mohammed Hasan Khan

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor, Dr. Sofiène Tahar, for his strong support, encouragement and guidance through out my Ph.D studies. He was always approachable and his insights about research and immense knowledge in the field of formal methods have strengthened this work significantly. I would also like to acknowledge his efforts for establishing a very friendly and stimulating atmosphere in the Hardware Verification Group (HVG).

I am indebted to Dr. Joe Hurd from Galois Inc., who not only inspired this thesis, but also followed the progress of my work and provided valuable feedback at various stages. I would also like to thank Prof. Mike Gordon for giving me the opportunity to visit his Automated Reasoning Group at the University of Cambridge. It was a great experience for my research development. Dr. Behzad Akbarpour, a former colleague and Research Associate at the Automated Reasoning Group, also provided great support during my stay at Cambridge. Many thanks to the active members of the HOL mailing list for getting back to me regarding numerous formalization and theorem proving problems.

I would like to express my gratitude to Dr. Konrad Slind for taking time out of his busy schedule to serve as my external examiner. I could not have a better expert than him worldwide. I sincerely thank Dr. Otmane Ait Mohamed, Dr. Ahmed Elhakeem and Dr. Hon Fung Li for serving on my doctoral advisory committee. Their constructive feedback and comments at various stages have been significantly useful in shaping the thesis to completion.

There are a number of other people at Concordia who have enriched my professional life in various ways. I would like to thank Dr. Skander Kort for all his help and support

as my supervisor for the initial period of my doctorol studies. The HVG members have always been very kind and helpful to me. I wish to thank all my present and former colleagues for their support and the nice time we have spent together.

Last but by no means least, it gives me immense pleasure to thank my family for their perpetual love and encouragement. Nothing I say can do justice to how I feel about their support. I feel very lucky to have a family that shares my enthusiasm for academic pursuits. My parents have provided me with countless opportunities for which I am eternally grateful. I must acknowledge my father who has been the prime inspiration for me for pursuing doctoral studies. My mother has always been very closely involved in my education and she is the one who developed my interest in the fields of mathematics and science. No doubt, its due to her efforts that I have reached this point. My wife, who has been with me in every moment of my PhD tenure, is my source of strength and without her support this thesis would never have started much less finished. Her love and encouragement kept me going on a thousand occasions. My sister always showed great interest in my work and provided immense appreciation, which has been very motivating. I would also like to mention my daughters, Minahil and Sireen, for bringing so much joy and fun in my life, which helped me in coming out of many frustrating moments during my PhD research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ABP	Alternating Bit Protocol
ACK	Acknowledgement
ARQ	Automated Repeat Request
CDF	Cumulative Distribution Function
CPU	Central Processing Unit
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
HOL	Higher Order Logic
ITM	Inverse Transform Method
LHS	Left-Hand-Side
ML	Meta Language
PDF	Probability Distribution Function
pGCL	Probabilistic Guarded-Command Language
PMF	Probability Mass Function
PVS	Prototype Verification System
RHS	Right-Hand-Side
RNG	Random Number Generator
TCP	Transmission Control Protocol

Chapter 1

Introduction

1.1 Motivation

> *"It is remarkable that this science, which originated in the consideration of games of chance, should have become the most important object of human knowledge ··· The most important questions of life are, for the most part, really only problems of probability."*
>
> *Pierre-Simon, Marquis de Laplace (1749-1827)*

This quote by the famous French mathematician and astronomer may appear exaggerated, but it is a fact that *probabilistic analysis* has become a tool of fundamental importance in almost every area of science and engineering. Of particular interest are modern 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 unavoidable elements of randomness and uncertainty is to use probabilistic analysis. The main idea behind probabilistic analysis is to mathematically model the random and 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 the system's behavior 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.

Even for hardware and software systems for which correctness may be unconditionally guaranteed, the study of system performance primarily relies on probabilistic analysis. In fact, the term *system performance* commonly refers to the average time required by a system to perform a given task, such as the average runtime of a computational algorithm or the average message delay of a telecommunication protocol. These averages can be computed, based on the probabilistic analysis approach, by using appropriate random variables to model inputs for the system model.

Simulation is the state-of-the-art probabilistic analysis technique. It allows us to conduct probabilistic analysis of analytically complex randomized models but most of the time is found to be quite inefficient. In fact, simulation requires an enormous

amount of numerical computations to generate meaningful results and can never guarantee exact answers. The precision and accuracy of the hardware and software system analysis results has become imperative these days because of the extensive usage of these systems in safety and financial critical areas, such as, medicine, transportation and stock exchange markets. Therefore, simulation cannot be relied upon for the analysis of such systems.

Formal methods [30] are capable of conducting precise system analysis and thus allow us to overcome the above mentioned limitations of the simulation approach. The main principle behind formal analysis of a system is to construct a computer based mathematical model of the given system and formally verify, within a computer, that this model meets rigorous specifications of intended behavior. Two of the most commonly used formal verification methods are model checking [15] and higher-order-logic theorem proving [26]. Model checking is an automatic verification approach for systems that can be expressed as a finite-state machine. Higher-order-logic theorem proving, on the other hand, is an interactive approach but is more flexible in terms of tackling a variety of systems.

Both model checking and theorem proving have been successfully used for the precise functional correctness of a broad range of hardware and software systems. On the other hand, their usage for probabilistic analysis has been somewhat limited. The main limitations being the restricted system expressibility and the inability to precisely reason about statistical properties, such as average values, in the case of model checking and the lack of mathematical foundations to conduct probabilistic analysis related proofs in the case of theorem proving.

This thesis takes steps to fill this gap as it presents some mathematical foundations that facilitate probabilistic analysis using the theorem-proving approach. What

distinguishes this thesis from previous work in the area is the broader range of systems and properties covered. Besides achieving 100% precise results, another major motivation behind using formal methods for probabilistic analysis is the ability to tackle both probabilistic analysis and functional correctness of a system in a single formal framework and thus allowing the usage of similar models for both tasks.

The ability to precisely conduct probabilistic analysis may prove to be a very advantageous feature for the analysis of hardware and software systems that are used in safety critical applications. The consequences of erroneous probabilistic analysis in these critical domains could be very devastating, as is quite evident from the following statement from Yale Patt's invited talk at the ISPASS-2004 conference [64] regarding the accuracy of performance analysis in the analysis of computer systems.

> *"Performance analysis can be one of the most important elements in the design cycle of a computer system. It can also be very important after the fact to influence future designs. But only if it is done right. If done wrong, it can be more harmful than if not done at all."*

1.2 Probabilistic Analysis

In this section, we elaborate more upon probabilistic analysis and present some of the fundamental components of a probabilistic analysis framework. This information will be used in the next section to compare the capabilities of the available computer based probabilistic analysis approaches.

1.2.1 Random Variables and Probabilistic Properties

In probabilistic analysis, random variables are used to describe random or unpredictable phenomenon in mathematical terms. For example, a random variable may be used to mathematically describe the outcome of rolling a die. Formally speaking, a random variable is defined as a measurable function from a probability space to some measurable space [7].

Every random variable gives rise to a probability distribution, which contains most of the important information about this random variable. The probability distribution of a random variable X can be uniquely described by its *cumulative distribution function* (CDF), which is defined as

$$F_X(x) = Pr(X \leq x) \tag{1.1}$$

for any number x, where Pr represents the probability function. There are two types of random variables - discrete and continuous.

Discrete Random Variables

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 [87]. Examples of discrete random variables include the outcome of rolling a dice and the number of children in a family. Discrete random variables can be completely characterized by their *probability mass function* (PMF). In probability theory, the PMF gives the probability that a random variable X is exactly equal to some value x.

$$p_X(x) = Pr(X = x) \tag{1.2}$$

A distinguishing characteristic of every discrete random variable is that its CDFs consists of a sequence of finite jumps. For example, the CDF and PMF for a discrete

Uniform random variable, which models the roll of a fair 6-sided die, are given in Figure 1.1.

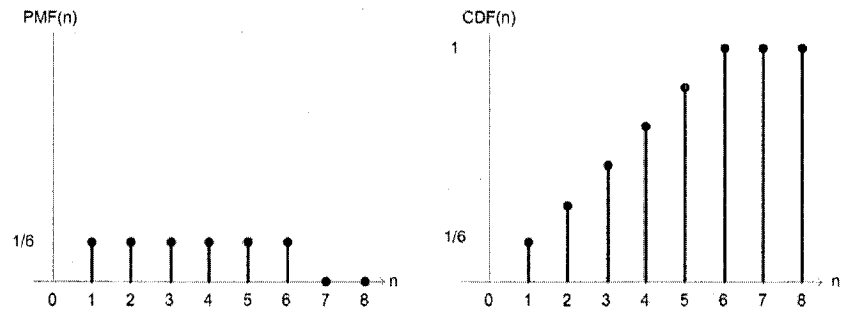

Figure 1.1: PMF and CDF for a Discrete Uniform Random Variable

Discrete random variables are used quite frequently to model random phenomenon while conducting probabilistic analysis of scientific and engineering applications. For example, the Bernoulli random variable is used to model channel noise in analyzing digital communication protocols [49], the Geometric random variable is widely used in the analysis of algorithms, such as QuickSort, median computation or the Coupon Collector's problem [61], and it is quite common to use the Binomial random variable in the probabilistic analysis related to quality control problems [84]. Another major application domain for discrete random variables is the performance analysis of cryptographic protocols [52].

Continuous Random Variables

A random variable is called continuous if it ranges over a continuous set of numbers [87]. A continuous set of numbers, sometimes referred to as an interval, contains all real numbers between two limits. An interval can be open (a,b) corresponding to the set $\{x|a < x < b\}$, closed [a,b] corresponding to the set $\{x|a \leq x \leq b\}$, or half-open (a,b], [a,b). Many experiments lead to random variables with a range that is a

continuous interval. Examples include measuring T, the arrival time of a data packet at a web server ($S_T = \{t|0 \leq t < \infty\}$) and measuring V, the voltage across a resistor ($S_V = \{v| - \infty < v < \infty\}$), where T and V are both continuous random variables.

A distinguishing feature of all continuous random variables is that their PMF is 0. Whereas, the CDF of continuous random variables is always a continuous function. Another useful probability distribution characteristic for continuous random variables is the *probability density function* (PDF), which represents the slope of the CDF.

$$f_X(x) = \frac{dF_X(x)}{dx} \tag{1.3}$$

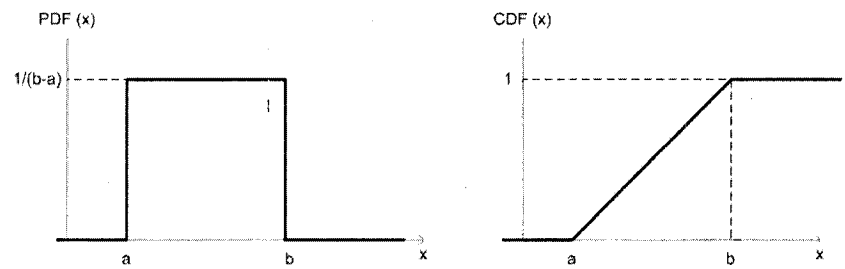

Figure 1.2: PDF and CDF for a Continuous Uniform Random Variable

As an example, consider the CDF and PDF for a continuous uniform random variable on the interval [a,b] given in Figure 1.2.

Like discrete random variables, continuous random variables are also widely used to mathematically describe random phenomenon in engineering and scientific applications. For example, the Continuous Uniform distribution is used to model quantization errors in computer arithmetic applications [86], the Exponential distribution occurs in applications such as queuing theory to model *interarrival* and *service* times and the Normal distribution is extensively used to model signals in data transmission and digital signal processing systems [84].

1.2.2 Statistical Properties: Expectation, Moments and Deviations

The concept of expectation, or the average, of a random variable is about as old as that of probability itself. The first published text on probability by C. Huyghens in 1657 [38] based probability on expectation. Expected values summarize the behavior of a random variable as a single number, rather than a distribution function, and thus prove to be a very convenient decision making factor while choosing the best option among the probabilistic analysis results of several possible candidates. The expectation of a random variable X is defined by

$$Ex[X] = \begin{cases} \sum_i x_i p_X(x_i), & \text{if } X \text{ is discrete;} \\ \int_{-\infty}^{\infty} x f_X(x) dx, & \text{if } X \text{ is continuous.} \end{cases} \tag{1.4}$$

where $\sum_i$ denotes the summation carried over all possible values of the random variable X.

Besides the expectation, there are several other statistical quantities that we can associate with a random variable. For example, we define the n^{th} moment of the random variable X as the expected value of the random variable X^n. With this terminology, the first moment of X is just the expectation. Another useful decision making characteristic of random variables is the measure of their dispersion. In performance and reliability analysis while looking at the failure rates of a system, it is often the case that we are quite interested in questions like, "How typical is the average?" or, "What are the chances of observing an event far from the average?". The most important measures of dispersion are the standard deviation and the variance. The variance of a random variable X describes the difference between X and its expected value.

$$Var[X] = \sigma_X^2 = \begin{cases} \sum_i (x_i - Ex[X])^2 p_X(x_i), & \text{if } X \text{ is discrete;} \\ \int_{-\infty}^{\infty} (x - Ex[X])^2 f_X(x) dx, & \text{if } X \text{ is continuous.} \end{cases} \tag{1.5}$$

It is clear that σ_X^2 is always a nonnegative number. The square root of the variance is known as the standard deviation σ_X; which is a useful characteristic as it has the same units as X and thus can be compared directly with the expected value.

Based on the expectation and variance characteristics of a random variable, we can find bounds for the tail distribution, i.e., the probability that a random variable assumes values that are far from its expectation. These bounds are usually calculated using the Markov's or the Chebyshev's inequalities [7]. The Markov's inequality gives an upper bound for the probability that a non-negative random variable X is greater than or equal to some positive constant

$$Pr(X \geq a) \leq \frac{Ex[X]}{a} \tag{1.6}$$

Markov's inequality gives the best tail bound possible, for a nonnegative random variable, using the expectation for the random variable only [61]. This bound can be improved upon if more information about the distribution of the random variable is taken into account. Chebyshev's inequality is based on this principle and it presents a significantly stronger tail bound in terms of variance of the random variable

$$Pr(|X - Ex[X]| \geq a) \leq \frac{Var[X]}{a^2} \tag{1.7}$$

where Var denotes the variance function. Due to the widespread interest in failure probabilities and the ease of calculation of tail distribution bounds using Equations (1.6) and (1.7), Markov and Chebyshev's inequalities have now become one of the core techniques in modern probabilistic analysis.

1.3 Probabilistic Analysis Approaches

Due to the vast application domain of probability, 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 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 and some related work in this field.

1.3.1 Simulation

Since the development of the probability theory in the last century, paper-and-pencil proof techniques have been used to perform probabilistic analysis for various engineering and scientific applications. These traditional techniques cannot cope with the complex systems and environments that are being worked on in this era. The advent of fast and inexpensive computational power in the last two decades opened up venues for using computers for probabilistic analysis. Today, simulation [72] is the most commonly used computer based probabilistic analysis technique. Most simulation software, e.g., Minitab [60], SAS [74], *mathStatica* [54], SPSS [77], Microsoft Excel [22], MATLAB [78], NESSUS [70], etc. contain a large collection of discrete and absolutely continuous univariate and multivariate distributions which in turn can be used to model systems with random or unpredictable components. These models are then analyzed using computer based techniques, such as the Monte Carlo method [51], 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 expectation, variance and tail distribution bounds, may then be calculated, based on the data collected during the sampling process, using their mathematical relations in a computer.

The probabilistic and statistical analysis conducted using simulation techniques is based on certain approximations and can be quite unreliable at times. It is a common occurrence that different software packages come up with different solutions to the same probabilistic analysis problem. The approximations that lead to this unreliability can be classified into three main categories.

- The source of randomness in the simulation based software packages is usually pseudorandom numbers; the numbers seem random but are actually the output of a *random number generator* (RNG) which is basically a deterministic algorithm [48]. The *period* of an RNG, i.e., the number of calls which can be made to the RNG before it begins to repeat itself, severely affects the accuracy of the result from a probabilistic or statistical software package [44].

- In most simulation based software packages, the CPU time required for generating nonuniform random numbers for arbitrary probability distributions is a major issue. Therefore functions for evaluating probability distributions are approximated using a variety of efficient algorithms [40]; some more accurate than others. For example, several algorithms for the incomplete beta function were assessed in [9] and only one of them was found to be reliable. Details of numerically stable computation of statistical functions are explored in [42] and [43].

- Like all other computer based computations, roundoff and truncation errors also creep into the numerical computations in these simulation based software packages [35]. These errors arise due to the finite precision representation of numbers in the computers.

McCullough [55] proposed a collection of intermediate-level tests for assessing

the numerical reliability of a statistical package in three areas: estimation, random number generation, and calculation of statistical distributions. McCullough applied his methodology to uncover flaws in most of the mainstream statistical packages, e.g., his analysis of the SAS and the SPSS packages are presented in [56], and of Microsoft Excel 2003 in [58]. *mathStatica* is the statistical add-on package for Mathematica [71], which provides an arbitrary-precision numeric engine, whereas most software packages provide only finite-precision numerics. There are two ways to increase the precision of calculation in Mathematica: (1) to increase the number of digits carried through calculations, and (2) *rationalize* the input data by converting them from reals to rationals. It is because of this arbitrary precision calculation of Mathematica that *mathStatica*'s accuracy outperforms all of the competition in each of the three benchmark areas in McCullough's methodology [57]. Though there are exceptions, the general rule of computing is that there is a tradeoff between speed and accuracy. That is, more accurate computation tends to be slower. In his excellent article on numerics in Mathematica, Sofroniou [76] maintains that, *"if you are using machine numbers you are primarily concerned with efficiency"*, as opposed to using extended precision numbers and being particularly concerned with accuracy.

Due to the above mentioned inaccuracies and the inherent nature of simulation, the probabilistic analysis results attained via simulation can never be termed as 100% accurate. Besides the inaccuracy of the results, another major limitation of simulation based probabilistic analysis is the enormous amount of CPU time requirement for attaining meaningful estimates. This approach generally requires hundreds of thousands of simulations to calculate the probabilistic quantities and becomes impractical when each simulation step involves extensive computations.

1.3.2 Model Checking

Probabilistic model checking [4, 73] is a rapidly emerging formal technique that is capable of providing exact solutions to probabilistic properties. Like traditional model checking [30], it involves the construction of a precise state-based mathematical model of the given probabilistic system, which is then subjected to exhaustive analysis to 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., [18, 63], and based on these algorithms, a number of tools have been developed, e.g., PRISM [67, 45], E⊢MC^2 [34], Rapture [39] and VESTA [75].

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 [16]. 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. 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.

Similarly, to the best of our knowledge, it has not been possible to precisely reason about statistical quantities, such as expectation, variance and tail distribution bounds, using probabilistic model checking so far. The most that has been reported in this domain is the approximate evaluation of expectation. Some probabilistic model checkers, such as PRISM and VESTA, offer the capability of verifying expected values in a semi-formal manner. For example, in the PRISM model checker, the basic idea is to augment probabilistic models with cost or rewards: real values associated

with certain states or transitions of the model, in order to analyze the expectation properties related to these rewards. The meaning ascribed to expected properties is, of course, dependent on the definitions of the rewards themselves and thus there is always some risk of verifying false properties. Similarly, the computed expected values are expressed in a computer based notation, such as fixed or floating point numbers, which also introduces some degree of approximation in the results.

1.3.3 Theorem Proving

Theorem proving [26] is another 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 axioms and primitive inference rules or any other already proved theorems or inference rules.

The verification effort of a theorem in a theorem prover varies from trivial to complex depending on the underlying logic [31]. For instance, first-order logic [24] is restricted to 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 enable completely automated proofs. More expressive logics, such as higher-order logic [10], 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 probabilistic

analysis, we need to formalize (mathematically model) random variables as functions and formalize characteristics of random variables, such as CDF and expectation, etc., 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 probabilistic analysis.

Probabilistic analysis can be conducted in the environment of a higher-order-logic theorem prover by first modeling the behavior of the system that needs to be analyzed in higher-order logic, while expressing its random or unpredictable elements in terms of formalized random variables. The second step is to use this formal model to express the probabilistic and statistical properties, regarding the system, in higher-order logic. For this purpose, we need to have access to higher-order-logic definitions of probabilistic and statistical properties of random variables, such as, PMF, CDF, expectation and variance, etc. Finally, theorems corresponding to the probabilistic and statistical properties of the system model can be mechanically checked for correctness in a theorem prover.

The above mentioned theorem proving based probabilistic analysis approach tends to overcome the limitations of the simulation and model checking based probabilistic analysis approaches. Due to the formal nature of the models and properties and the inherent soundness of the theorem proving approach, probabilistic analysis carried out in this way will be free from any approximation and precision issues. Similarly, the high expressibility of higher-order logic allows us to analyze a wider range of systems without any modeling limitations, such as the state-space explosion problem in the case of probabilistic model checking, and formally verify analytically complex properties, such as expectation, variance and tail distribution bounds. Furthermore, the use of mechanical theorem provers will increase the confidence in the proofs over

the traditional paper-and-pencil approach.

Nędzusiak [62] and Bialas [6] were among the first ones to propose a formalization of some probability theory in higher-order logic. Hurd [36] developed a framework for the verification of probabilistic algorithms in the HOL (which stands for Higher-Order-Logic) theorem prover [27]. Random variables are basically probabilistic algorithms and thus can be formalized and verified, based on their probability distribution properties, using the methodology proposed in [36]. In fact, [36] presents the formalization of some discrete random variables along with their verification, based on the corresponding PMF properties. These random variables can thus be used to formally model discrete random components of a system in higher-order logic. This system's probabilistic properties can also be expressed in terms of the PMF relations of the corresponding random variables and thus can be formally reasoned about in the HOL theorem prover. Hurd demonstrated the practical effectiveness of his formal framework by successfully verifying the Miller-Rabin primality test, a well-known and commercially used probabilistic algorithm. This case study utilizes a formal model of the discrete Uniform random variable. Hurd *et. al* [37] also formalized the *probabilistic guarded-command language* (*pGCL*) in HOL. The *pGCL* contains both demonic and probabilistic nondeterminism and thus makes it suitable for reasoning about distributed random algorithms. Celiku [13] built upon the formalization of the *pGCL* to mechanize a quantitative temporal logic and demonstrated the ability to reason about quantized aspects of randomized algorithms in HOL.

An alternative method for probabilistic verification in higher-order logic has been presented by Audebaud *et. al* [3]. Instead of using the measure theoretic concepts of probability space, as is the case in Hurd's approach, Audebaud *et. al* based their methodology on the monadic interpretation of randomized programs as probabilistic

distribution. This approach only uses functional and algebraic properties of the unit interval and has been successfully used to verify a sampling algorithm of the Bernoulli distribution and the termination of various probabilistic programs in the Coq theorem prover [17].

Richter [69] formalized a significant portion of the Lebesgue integration theory in higher-order logic using Isabelle/Isar [65]. He also linked the Lebesgue integration theory to probabilistic algorithms, developing upon Hurd's [36] framework, and presented the formalization of the first moment method. Due to its strong mathematical foundations, the Lebesgue integration theory may be used to formalize the expectation of continuous random variables.

To the best of our knowledge, no higher-order logic formalization of *continuous random variables* exists in the open literature so far. The algorithms for discrete random variables, formalized by the above mentioned researchers, are either guaranteed to terminate or satisfy probabilistic termination, meaning that the probability that the algorithm terminates is 1. On the other hand, the modeling of continuous random variables requires non-terminating programs and hence calls for a different approach. Thus, it is not possible to conduct the probabilistic analysis of systems with continuous random components using the available formalization. Similarly, formally reasoning about *statistical characteristics of random variables* is not possible using the available research. Richter's formalization of the Lebesgue integral may be used for this purpose but it involves dense mathematical concepts and leads to a complex verification task. It is not a straightforward task to formalize a random variable and verify its expectation property using the formalized Lebesgue integration theory. Likewise, the verification of variance and tail distribution properties and the analysis of probabilistic systems that involve multiple random variables is even more

challenging.

1.4 Proposed Methodology

The objective of this thesis is mainly targeted towards the development of a theorem proving based probabilistic analysis framework that can handle the analysis of real-world hardware and software systems. In particular, we have developed a framework characterizing:

1. The ability to formally express both discrete and continuous random variables in higher-order logic. These formalized random variables can be used to develop formal models for systems with either discrete or continuous random components.

2. The ability to formally express the probabilistic and statistical properties of both discrete and continuous random variables in higher-order logic. These formalized properties can be used to develop theorems representing the characteristics of interest for the formal model of the given system.

3. The ability to formally reason about the theorems, formalized in Step 2, using a theorem prover.

4. The ability to utilize the above mentioned capabilities to formally model and reason about real-world probabilistic analysis problems.

Figure 1.3 depicts the above mentioned characteristics and presents a hypothetical model of our intended probabilistic analysis framework. The discrete and continuous random variable boxes denote frameworks for the formalization of the corresponding random variables. Probabilistic and statistical property boxes are used

for both discrete and continuous random variables and they basically denote the infrastructures to express and reason about the corresponding properties in a higher-order-logic theorem prover.

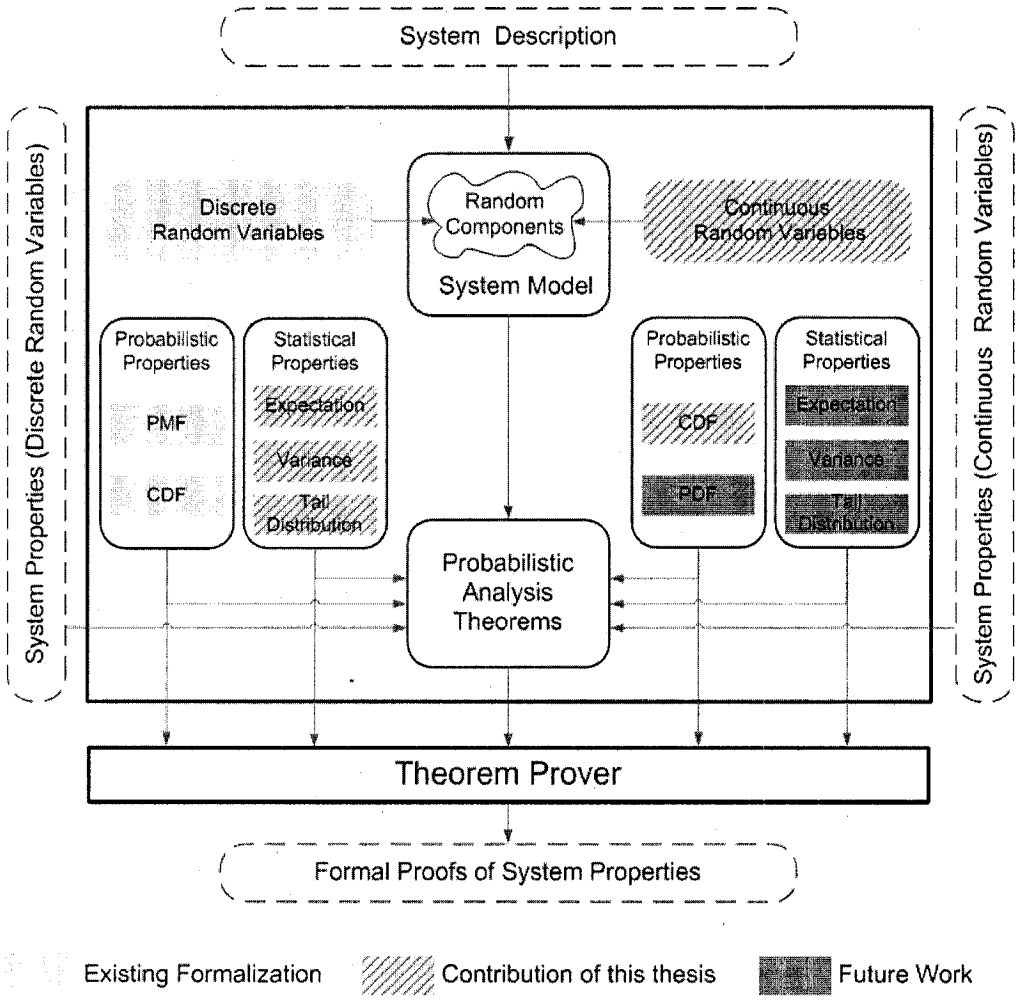

Figure 1.3: Formal Probabilistic Analysis Framework

Hurd's methodology for the verification of probabilistic algorithms [36] can be utilized to formalize discrete random variables and verify their corresponding probabilistic properties in a theorem prover. The boxes associated with these existing capabilities, i.e., a framework for the formalization of discrete random variables and

an infrastructure to formally express and reason about probabilistic properties regarding discrete random variables, are highlighted in a light shade of grey in Figure 1.3. Moving towards a successful theorem proving based probabilistic analysis framework, we tackle three main characteristics. First of all, we propose an infrastructure that allows us to formally express and reason about some statistical properties regarding discrete random variables. Secondly, we propose a framework that facilitates both the formalization of continuous random variables and the formalization and verification of the corresponding CDF properties. These are the main contributions of this thesis and are highlighted with stripes in Figure 1.3. The remaining characteristics of the intended probabilistic analysis framework, i.e., formally expressing and reasoning about PDF and statistical properties of continuous random variables, are still open issues. These future research direction are highlighted in a dark shade of grey in Figure 1.3,

In the following, we briefly present our methodologies for developing the formal probabilistic analysis characteristics that we have tackled in this thesis.

- **Statistical Properties for Discrete Random Variables**: We propose an infrastructure that allows us to formally specify statistical properties like expectation, variance and tail distribution bounds, for discrete random variables and formally verify them in the HOL theorem prover. The infrastructure is primarily based on a formal definition of expectation of a function of a discrete random variable in higher-order logic. Instead of formalizing a generalized definition of expectation based on the mathematical concept of sample space, we consider one of its variants that deals with discrete random variables that take on values in the positive integers, i.e., $\{0, 1, 2, \cdots\}$, only. This simplification leads to a much faster formalization and verification and thus enables

us to demonstrate the effectiveness of the proposed performance analysis approach by targeting real-world applications. Building upon our definition of expectation of a function of a discrete random variable, we formalize the mathematical concept of variance and verify some classical properties of expectation and variance. For illustration purposes, we present the formal verification of expectation and variance relations for some of the widely used discrete random variables, such as Uniform(m), Bernoulli(p), Geometric(p) and Binomial(m, p). The thesis also provides the formal verification of Markov's and Chebyshev's inequalities for discrete random variables, which allows us to formally reason about the respective tail distribution properties in a theorem prover. Finally, in order to demonstrate the usefulness of the above mentioned formalization, we utilize the HOL theorem prover for conducting probabilistic analysis of the Coupon Collector's problem [61], a well-known commercially used probabilistic algorithm.

- **Continuous Random Variables and their CDF Properties**: We present a generic framework for the formalization of continuous random variables, for which the inverse of the CDF can be represented in a closed mathematical form, and the verification of the corresponding CDF properties in the HOL theorem prover. The proposed framework is primarily based on the Inverse Transform Method (ITM) [20]. The ITM is a well known nonuniform random generation technique for generating random variates with arbitrary distributions using a Standard Uniform random number generator. The main advantage of this approach is that we only need to formalize one continuous random variable from scratch, i.e., the Standard Uniform random variable. Other continuous random variables can then be formalized and verified by using the formally verified

ITM. The thesis includes the higher-order-logic formalization and verification details of the Standard Uniform random variable and the ITM. For illustration purposes, we present the formalization and verification of Exponential(λ), Uniform(a, b), Rayleigh(λ) and Triangular($0, a$) random variables in HOL. We also demonstrate the usefulness of formalized continuous random variables in the formal probabilistic analysis approach by presenting the verification of a couple of probabilistic properties regarding the roundoff error in a digital processor.

- **Case Study: Stop-and-Wait Protocol**: In order to demonstrate the feasibility and usefulness of the proposed formalization, we present the functional verification and probabilistic analysis of the Stop-and-Wait protocol. Like other real-time systems, the Stop-and-Wait protocol involves a subtle interaction of a number of distributed components and exhibits a high degree of parallelism, which makes its performance analysis quite complex. Thus, traditional techniques, such as simulation or the state-based formal methods, usually fail to produce reasonable results. On the other hand, we are able to achieve precise results for this problem using the higher-order-logic theorem proving based probabilistic analysis approach. We formalize the protocol as a logical conjunction of higher-order-logic predicates while the channel noise is modeled using the Bernoulli(p) random variable. Based on this model, the average message delay relation, which is the most essential performance related parameter for communication protocols, is formally verified in HOL. Our results exactly match the results obtained by paper-and-pencil proof techniques and are thus 100 % precise. The Stop-and-Wait protocol is a classical example of a real-time system and thus the approach followed in this case study can be essentially utilized for the analysis of any other real-time system as well.

It is important to note that the proposed higher-order-logic theorem proving approach has its own limitations. Even though theorem provers have been successfully used for a variety of tasks, including some that have eluded human mathematicians for a long time, but these successes are sporadic, and work on hard problems usually requires a proficient user and a lot of formalization. On the other hand, simulation based techniques are at least capable of offering approximate solutions to very complex problems and probabilistic model checking can automatically provide precise answers to probabilistic analysis problems that can be expressed by a reasonable sized state-machine. Therefore, we consider higher-order-logic theorem proving as a complementary technique to simulation and probabilistic model checking, i.e., the methods have to play together for a successful probabilistic analysis framework. For example, theorem proving can be used for the safety critical parts of the analysis, which cannot be handled by probabilistic model checking, and simulation and model checking based approaches can handle the rest.

1.5 Thesis Contributions

In summary, the primary focus of this thesis is on the idea of using higher-order-logic theorem proving for conducting probabilistic analysis. This approach allows us to achieve precise probabilistic analysis results and thus proves to be quite useful for the performance and reliability optimization of safety critical hardware and software systems. In moving towards a successful higher-order-logic theorem proving based probabilistic analysis framework, the thesis makes the following contributions.

1. It presents an infrastructure that allows us to formally specify and verify higher-order-logic theorems related to the expectation, variance and tail distribution

properties for discrete random random variables in a theorem prover [Bio-Jr-2,Bio-Jr-3,Bio-Cf-2,Bio-Cf-3,Bio-Tr-1][1].

2. It presents a framework that can be used to formally specify higher-order-logic models of any continuous random variable for which the inverse of the CDF can be expressed in a closed mathematical form. This framework also allows us to specify and verify higher-order-logic theorems related to the CDF properties for these continuous random variables in a theorem prover [Bio-Jr-1,Bio-Cf-5,Bio-Cf-6,Bio-Tr-1,Bio-Tr-2,Bio-Tr-3].

3. It presents the complete functional verification and performance analysis of the Stop-and-Wait protocol in a higher-order-logic theorem prover. We believe that this is the first time that a computer based formal method has been able to tackle these problems in a unified framework with 100% precise results. The analysis approach for this case study is quite general and can be utilized to conduct the analysis of other real-time systems as well [Bio-Jr-4,Bio-Cf-1].

1.6 Organization of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, we provide a brief introduction to the HOL theorem prover and an overview of Hurd's methodology for the verification of probabilistic algorithms in HOL to equip the reader with some notation and concepts that are going to be used in the rest of this thesis. Chapter 3 describes the formalization infrastructure that enables us to formally specify and verify expectation, variance and tail distribution bounds for discrete random variables in HOL. To demonstrate this infrastructure, the chapter also presents the formal verification of

[1]The references with prefix Bio are provided in the Biography Section.

mean and variance characteristics of some commonly used discrete random variables and the probabilistic analysis of Coupon Collector's problem as a case study. Chapter 4 presents the framework for the formalization of continuous random variables and the verification of the corresponding CDF properties in HOL. To demonstrate this framework, the chapter also presents the formalization and verification of some commonly used continuous random variables and the simple probabilistic analysis example of roundoff error in a digital processer. Next, in Chapter 5, we illustrate the practical effectiveness of the proposed approach (and the above mentioned formalization) by successfully applying it for the formal functional verification and performance analysis of the Stop-and-Wait protocol. Finally, Chapter 6 concludes the thesis and outlines some future research directions.

Chapter 2

Preliminaries

In this chapter, we provide a brief introduction to the HOL theorem prover and present an overview of Hurd's methodology [36] for the verification of probabilistic algorithms. The intent is to introduce the basic theories along with some notation that is going to be used in the rest of the thesis.

2.1 HOL Theorem Prover

This thesis uses the HOL theorem prover to conduct all the probabilistic analysis related formalization and verification. HOL is an interactive theorem prover developed by Mike Gordon at the University of Cambridge for conducting proofs in higher-order logic. It utilizes the simple type theory of Church [14] along with Hindley-Milner polymorphism [59] to implement higher-order logic. HOL has been successfully used as a verification framework for both software and hardware as well as a platform for the formalization of pure mathematics.

2.1.1 Secure Theorem Proving

In order to ensure secure theorem proving, the logic in the HOL system is represented in the strongly-typed functional programming language ML [66]. An ML abstract data type is used to represent higher-order-logic theorems and the only way to interact with the theorem prover is by executing ML procedures that operate on values of these data types. The HOL core consists of only 5 basic axioms and 8 primitive inference rules, which are implemented as ML functions. 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.

2.1.2 Terms

There are four types of HOL terms: constants, variables, function applications, and lambda-terms (denoted function abstractions). Polymorphism, types containing type variables, is a special feature of higher-order logic and is thus supported by HOL. Semantically, types denote sets and terms denote members of these sets. Formulas, sequences, axioms, and theorems are represented by using terms of Boolean types.

2.1.3 Theories

A HOL theory is a collection of valid HOL types, constants, axioms and theorems and is usually stored as a file in computers. Users can reload a HOL theory in the HOL system and utilize the corresponding definitions and theorems right away. The concept of HOL theory allows us to build upon existing results in an efficient way without going through the tedious process of regenerating these results using the basic axioms and primitive inference rules.

HOL theories are organized in a hierarchical fashion. Any theory may inherit

types, definitions and theorems from other available HOL theories. The HOL system prevents loops in this hierarchy and no theory is allowed to be an ancestor and descendant of a same theory. Various mathematical concepts have been formalized and saved as HOL theories by the HOL users. These theories are available to a user when he first starts a HOL session. We utilized the HOL theories of Booleans, lists, sets, positive integers, *real* numbers, measure and probability in our work. In fact, one of the primary motivations of selecting the HOL theorem prover for our work was to benefit from these built-in mathematical theories.

2.1.4 Writing Proofs

HOL supports two types of interactive proof methods: forward and backward. In forward proof, the user starts with previously proved theorems and applies inference rules to reach the desired theorem. In most cases, the forward proof method is not the easiest solution as it requires the exact details of a proof in advance. A backward or a goal directed proof method is the reverse of the forward proof method. It is based on the concept of a *tactic*; which is an ML function that breaks goals into simple subgoals. In the backward proof method, the user starts with the desired theorem or the main goal and specifies tactics to reduce it to simpler intermediate subgoals. Some of these intermediate subgoals can be discharged by matching axioms or assumptions or by applying built-in decision procedures. The above steps are repeated for the remaining intermediate goals until we are left with no further subgoals and this concludes the proof for the desired theorem.

The HOL theorem prover includes many proof assistants and automatic proof procedures [31] to assist the user in directing the proof. The user interacts with a proof editor and provides it with the necessary tactics to prove goals while some of

the proof steps are solved automatically by the automatic proof procedures.

2.1.5 HOL Symbols

Table 2.1 provides the mathematical interpretations of some frequently used HOL symbols and functions, which are inherited from existing HOL theories, in this thesis.

HOL 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*	n^{th} element of list L
mem a L	*member*	True if a is a member of list L
length L	*length*	Length of list L
(a, b)	a x b	A pair of two elements
fst	fst (a, b) = a	First component of a pair
snd	snd (a, b) = b	Second component of a pair
λx.t	$\lambda x.t$	Function that maps x to $t(x)$
{x|P(x)}	$\{\lambda x.P(x)\}$	Set of all x such that $P(x)$
num	$\{0, 1, 2, \ldots\}$	Positive Integers data type
real	All Real numbers	Real data type
suc n	$n + 1$	Successor of a *num*
sqrt x	$\sqrt{x}$	Square root function
abs x	$\|x\|$	Absolute function
lim(λn.f(n))	$\lim_{n\to\infty} f(n)$	Limit of a *real* sequence f
convergent(λn.f(n))	$\exists x. \lim_{n\to\infty} f(n) = x$	f is convergent
suminf(λn.f(n))	$\lim_{k\to\infty} \sum_{n=0}^{k} f(n)$	Infinite summation of f
summable(λn.f(n))	$\exists x. \lim_{k\to\infty} \sum_{n=0}^{k} f(n) = x$	Summation of f is convergent

Table 2.1: HOL Symbols and Functions

2.2 Verifying Probabilistic Algorithms in HOL

The proposed formalization and verification methodologies, given in this thesis, build upon Hurd's methodology [36] for the verification of probabilistic algorithms. In this section, we provide a brief description about this work whereas more details can be found in the original dissertation [36].

2.2.1 Formalization of Probabilistic Algorithms

Probabilistic algorithms can be formalized in higher-order logic as deterministic functions with access to an infinite Boolean sequence $\mathbb{B}^\infty$; source of an infinite random bits modeled by $num \rightarrow bool$. These deterministic functions make random choices based on the result of popping the top most bit in the infinite Boolean sequence and may pop as many random bits as they need for their computation. When the algorithms terminate, they return the result along with the remaining portion of the infinite Boolean sequence to be used by other programs. Thus, a probabilistic algorithm which takes a parameter of type α and ranges over values of type β can be represented in HOL 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 equal probability $\frac{1}{2}$ can be modeled as follows

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

where the variable s, in the above definition, represents the infinite Boolean sequence and the functions `shd` and `stl` are the sequence equivalents of the list operation *'head'* and *'tail'*. The function `bit` accepts the infinite Boolean sequence and returns

a random number that is either 0 or 1 together with a sequence of unused Boolean sequence, which in this case is the tail of the sequence.

One of the primary advantages of using an infinite Boolean sequence as the only source of randomness for probabilistic algorithms is the fact that only one probability space needs to be formalized in the logic.

2.2.2 Monadic Notation

Higher-order-logic functions for probabilistic algorithms can also be expressed in the more general state-transforming monad, where the states are the infinite Boolean sequences.

```
⊢ ∀ a s.  unit a s = (a,s)
⊢ ∀ f g s.  bind f g s = g (fst (f s)) (snd (f s))
```

where the HOL functions `fst` and `snd` return the first and second components of a pair, respectively. The `unit` operator is used to lift values to the monad, and the `bind` is the monadic analogue of function application. All monad laws hold for this definition, and the notation allows us to write functions without explicitly mentioning the sequence that is passed around, e.g., function `bit` can be defined as

```
⊢ bit_monad = bind sdest (λb.  if b then unit 1 else unit 0)
```

where the function `sdest` returns the head and tail of its argument sequence as a pair `sdest s = (shd s, stl s)`.

2.2.3 Formalized Probability Theory

Hurd [36] also formalized some mathematical measure theory in HOL in order to define 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 verify the basic laws of probability in the HOL theorem prover. For example, the additive law, which represents the probability of two disjoint events as the sum of their probabilities, can be formally verified as follows:

$$\vdash \forall\ \mathtt{A}\ \mathtt{B}.\ \mathtt{A} \in \mathcal{E} \wedge \mathtt{B} \in \mathcal{E} \wedge \mathtt{A} \cap \mathtt{B} = \emptyset \Rightarrow \mathbb{P}(\mathtt{A} \cup \mathtt{B}) = \mathbb{P}(\mathtt{A}) + \mathbb{P}(\mathtt{B})$$

The formalized $\mathbb{P}$ and $\mathcal{E}$ can also be used to prove probabilistic properties for probabilistic programs such as

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

where the function `fst` returns the first component of a pair.

2.2.4 Measurability and Independence

The *measurability* and *independence* of a probabilistic function are important concepts in probability theory. A property `indep`, called *strong function independence*, is introduced in [36] such that if $f \in$ `indep`, then all sets involving the function f will be both measurable and independent. In this approach, a set of infinite Boolean sequences, S, is said to be measurable if and only if it is in $\mathcal{E}$, i.e., S $\in \mathcal{E}$. Since the probability measure $\mathbb{P}$ is only defined on sets in $\mathcal{E}$, it is very important to prove that sets that arise in verification are measurable. It has been shown in [36] that a function is guaranteed to preserve *strong function independence*, if it accesses the infinite Boolean sequence using only the `unit`, `bind` and `sdest` primitives. All reasonable probabilistic programs preserve *strong function independence*, and these extra properties are a great aid to verification.

2.2.5 Probabilistic While Loop

The features of Hurd's methodology described so far, allow us to formalize and verify all probabilistic algorithms that compute a finite number of values equal to 2^n and the occurrence of each one of these values is equiprobable with a probability $\frac{m}{2^n}$: where m is a positive integer and is always less than 2^n. These kind of random variables can be modeled, using Hurd's framework, by well-founded recursive functions. There are many probabilistic algorithms that do not satisfy the above conditions. For example, the Geometric(p) returns the index of the first success in an infinite sequence of Bernoulli(p) trials [19] and thus ranges over a countably infinite number of positive integers.

Hurd defined a probabilistic version of the while loop, i.e., the *probabilistic while loop*, which allows us to formalize the probabilistic algorithms that do not satisfy the above conditions but are sure to terminate. It has been verified in HOL that the *probabilistic while loop* preserves useful program properties of measurability and independence, provided the condition that "*from every starting state, the while loop will terminate with probability 1*" is satisfied. For illustration purposes, [36] presents the formalization and verification of two probabilistic algorithms using the *probabilistic while loop*.

Chapter 3

Statistical Properties for Discrete Random Variables

Statistical properties play a vital role in the present age probabilistic analysis. In this chapter, we present some formalization that allows us to reason about the expectation, variance and tail distribution properties of discrete random variables in the HOL theorem prover. This infrastructure can be used to formally specify and verify theorems for statistical properties regarding the discrete random components of systems in HOL. To illustrate the practical effectiveness of the results presented in this chapter, we utilize them to conduct formal probabilistic analysis of the Coupon Collector's problem.

3.1 Introduction

In probabilistic analysis, expectation and variance play a major role in decision making as they tend to summarize the probability distribution characteristics of a random

variable in a single number. For example, it is more convenient to find the most efficient algorithm for an NP-hard problem by comparing the expectations of runtime for the available options rather than comparing their respective probability distribution functions. Similarly, in performance and reliability analysis while looking at the failure rates of a system, it is often the case that we are interested in the probability that a random variable assumes values that are far from its expectation value. Instead of characterizing this probability by a distribution function, we commonly rely on its upper bounds. These bounds, usually termed as the tail distribution bounds, can be computed using the Markov's or the Chebyshev's inequalities along with the expectation and variance relations of the corresponding random variable [7].

Due to its widespread interest, the computation of statistical characteristics has now become one of the core components of every modern probabilistic analysis framework. In this chapter, we present a formalization infrastructure, given in Figure 3.1, that allows us to reason about expectation, variance, and tail distribution properties regarding discrete random variables using a higher-order-logic theorem prover. For this purpose, we mainly build upon the higher-order-logic formalization of the probability theory, given in [36].

The first step in the proposed infrastructure, illustrated in Figure 3.1, is the formalization of an expression for the expectation of a function of a discrete random variable in higher-order logic. Building upon this definition, we formalize the mathematical concept of variance and verify some classical properties of expectation and variance, including the linearity of expectation and variance properties for discrete random variables. Both of these linearity properties are quite useful as they allow us to formally reason about the expectation and variance properties of systems involving multiple random variables. The next step, as shown in Figure 3.1, is to utilize

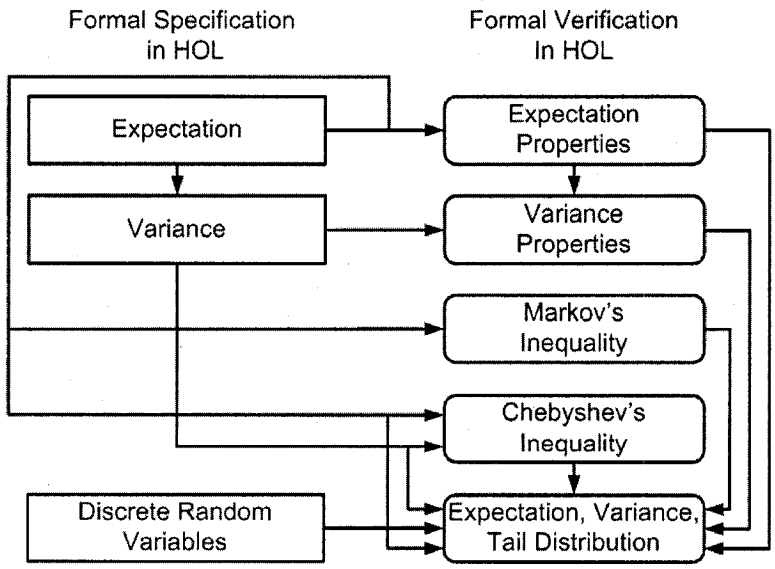

Figure 3.1: Formalization Infrastructure for Reasoning about Statistical Properties

the formal definitions of expectation and variance to verify the Markov's and Chebyshev's inequalities, given in Equations (1.6) and (1.7), for discrete random variables, respectively. The above mentioned formalization and verification allows us to reason about expectation, variance and tail distribution properties of any formalized discrete random variable.

In order to demonstrate the proposed infrastructure, we present the verification of expectation and variance relations for a few widely used discrete random variables: Uniform(m), Bernoulli(p), Geometric(p) and Binomial(m, p), in this chapter. These formally verified expectation and variance relations can also be used along with the formally verified Markov's and Chebyshev's inequalities to formally reason about the tail distribution characteristics of their respective random variables.

Finally, in order to illustrate the practical effectiveness of the formalization presented in this chapter, we utilize the propesed infrastructure to conduct the formal performance analysis of the Coupon Collector's problem [61], which is a well known

commercially used algorithm in computer science. Coupon Collector's problem is motivated by "*collect all n coupons and win*" contests. The problem is to find the number of trials that we need to find all the n coupons, assuming that a coupon is drawn independently and uniformly at random from n possibilities. We first present a formalization of the Coupon Collector's problem using the Geometric(p) random variable. Using this model along with the formalization infrastructure, presented in this chapter, we then illustrate the process of formally reasoning about the tail distribution properties of the Coupon Collector's problem in HOL.

3.2 Expectation for Discrete Random Variables

In this section, we first present a higher-order-logic formalization of the expectation of a function of a discrete random variable that is the first step in the proposed formalization infrastructure for reasoning about statistical properties, given in Figure 3.1. We later utilize this definition to verify a few classical expectation properties in HOL.

3.2.1 Formalization of Expectation

Expectation basically provides the average of a random variable, where each of the possible outcomes of this random variable is weighted according to its probability [7]

$$Ex[X] = \sum_{i} x_i p_X(x_i) \tag{3.1}$$

where $\sum_i$ denotes the summation carried over all the possible values of the random variable X. The above definition only holds if the summation is convergent, i.e., $\sum_i x_i p_X(x_i) < \infty$. Instead of formalizing this general definition of expectation based

on the principles of probability space, we concentrate on one of its variants that deals with discrete random variables that take on values only in the positive integers, i.e., $\{0, 1, 2, \cdots\}$.

This choice has been made mainly because of two reasons. First of all, in most of the engineering and scientific probabilistic analysis problems, we end up dealing with discrete random variables that attain values in positive integers only. For example, consider the cases of analyzing the performance of algorithms [61], cryptographic [52] and communication protocols [49], etc. Secondly, this simplification allows us to utilize the summation of a real sequence to model an expectation function and thus speed up the associated formalization and verification process by a considerable extent.

The expectation for a function of a discrete random variable, which attains values in the positive integers only, is defined as follows [50]

$$Ex[f(R)] = \sum_{n=0}^{\infty} f(n)Pr(R = n) \tag{3.2}$$

where R is the discrete random variable and f represents a function of the random variable R. The above definition only holds if the associated summation is convergent, i.e., $\sum_{n=0}^{\infty} f(n)Pr(R = n) < \infty$.

Equation (3.2) can be formalized in HOL, for a discrete random variable R that attains values in positive integers only and a function f that maps this random variable to a *real* value, as follows

Definition 3.1: *Expectation of Function of a Discrete Random Variable*

```
⊢ ∀ f R. expec_fn f R = suminf(λn.  (f n)ℙ{s | fst(R s) = n})
```

where the mathematical notions of the probability function $\mathbb{P}$ and random variable `R` have been inherited from [36], as presented in Section 2.2. The HOL function `suminf` represents the infinite summation of a *real* sequence [32]. The function `expec_fn`

accepts two parameters, the function f of type $(num \rightarrow real)$ and the positive integer valued random variable R and returns a *real* number.

The expected value of a discrete random variable that attains values in positive integers can now be defined in HOL as a special case of the above definition

Definition 3.2: *Expectation of a Discrete Random Variable*

```
⊢ ∀ R. expec R = expec_fn (λn.  n) R
```

where the lambda abstraction function `(λn.  n)` implements the identity function. The function **expec** accepts a positive integer valued random variable R and returns its expectation as a *real* number.

3.2.2 Verification of Expectation Properties

In this section, we utilize the formal definitions of expectation, developed in the last section, to prove some classical properties of the expectation [81]. These properties not only verify the correctness of our definitions but also play a vital role in verifying the expectation characteristic of discrete random components of probabilistic systems, as will be seen in Section 3.6 for the case of the Coupon's Collector's problem.

Expectation of a Constant

$$Ex[c] = c \tag{3.3}$$

where c is a positive integer. The random variable in this case is the degenerate random variable $R \equiv c$, where $R(s) = c$ for every $s \in$ *sample space*. It can be formally expressed as `unit c`, where the monadic operator **unit** is described in Section 2.2. Using this representation and the definition of expectation, given in Definition 3.2, the above property can be expressed in HOL as follows

Theorem 3.1: *Expectation of a Constant*

$\vdash \forall$ c. expec (unit c) = c

Rewriting the proof goal of the above property with Definition 3.2, we get

$$\lim_{k\to\infty}(\sum_{n=0}^{k} n\ \mathbb{P}\{s \mid \texttt{fst(unit c s)} = n\}) = c \tag{3.4}$$

Now, the probability term on the *left-and-side* (LHS) of the above subgoal can be expressed as follows

$$\forall\ n\ c.\ \mathbb{P}\{s \mid \texttt{fst(unit c s)} = n\} = (\texttt{if}\ (c = n)\ \texttt{then 1 else 0}) \tag{3.5}$$

and the proof is based on the basic probability theory laws and the measurability property of the random variable unit c. Using this property, the subgoal of Equation (3.4) can be rewritten as follows

$$\lim_{k\to\infty}(\sum_{n=0}^{k} n\ (\texttt{if}\ (c = n)\ \texttt{then 1 else 0})) = c \tag{3.6}$$

The summation on the *right-hand-side* (RHS) of the above subgoal can be proved to be convergent since its value remains the same for all values of n that are greater than c. Using this fact and the summation properties of a *real* sequence the above subgoal can be verified in HOL, which concludes the proof of Theorem 3.1.

Linearity of Expectation for Discrete Random Variables

$$Ex[\sum_{i=1}^{n} R_i] = \sum_{i=1}^{n} Ex[R_i] \tag{3.7}$$

where R represents a sequence of n discrete random variables. According to the linearity of expectation property, the expectation of a sum of random variables equals the sum of their individual expectations. It is one of the most important properties of

expectation as it allows us to verify the expectation properties of random behaviors involving multiple random variables without going into the complex verification of their joint probability distribution properties. Thus, its verification is a significant step towards using HOL as a successful probabilistic analysis framework.

We split the verification of linearity of expectation property in two major steps. Firstly, we verify the property for two discrete random variables and then extend the results by induction to prove the general case. The linearity of expectation property can be defined for any two discrete random variables X and Y as follows.

$$Ex[X + Y] = Ex[X] + Ex[Y] \tag{3.8}$$

To prove the above relationship in HOL, we proceed by first defining a function that models the summation of two random variables.

Definition 3.3: *Summation of Two Random Variables*

```
⊢ ∀ X Y.
    sum_two_rv X Y = bind X (λa.  bind Y (λb.  unit (a + b)))
```

The function, `sum_two_rv`, accepts two random variables and returns one random variable that represents the sum of the two argument random variables. It is important to note that the above definition implicitly ensures that the call of the random variable Y is independent of the result of the random variable X. This is true because the infinite Boolean sequence that is used for the computation of Y is the remaining portion of the infinite Boolean sequence that has been used for the computation of X. This characteristic led us to prove that the function `sum_two_rv` preserves *strong function independence*, which is the most significant property in terms of verifying properties on probabilistic functions.

Lemma 3.1: *sum_two_rv Preserves Strong Function Independence*

```
⊢ ∀ X Y. X ∈ indep_fn ∧ Y ∈ indep_fn
  ⇒ ((sum_two_rv X Y) ∈ indep_fn)
```

The above property can be verified in HOL using the fact that the function `sum_two_rv` accesses the infinite Boolean sequence using the `unit` and `bind` operators.

Now, the linearity of expectation property for two discrete random variables, which preserve *strong function independence*, with *well-defined* expectation values, i.e., the summation in their expectation definition is convergent, can be stated in HOL using the `sum_two_rv` function as follows.

Lemma 3.2: *Linearity of Expectation for Two Discrete Random Variables*

```
⊢ ∀ X Y. X ∈ indep_fn ∧ Y ∈ indep_fn ∧
summable(λn.  n ℙ{s | fst(X s) = n}) ∧
summable(λn.  n ℙ{s | fst(Y s) = n})
  ⇒ (expec (sum_two_rv X Y) = expec X + expec Y)
```

where `summable` accepts a *real* sequence and returns *True* if the infinite summation of this sequence is convergent (i.e., `summmable M` $= \exists x. \lim_{k\to\infty}(\sum_{n=0}^{k} M(n)) = x$).

Rewriting the proof goal of Lemma 3.2 with the definitions of the functions `expec`, `sum_two_rv` and `summable`, simplifying it with some infinite summation properties and removing the monad notation, we reach the following subgoal in HOL.

$$(\lim_{k\to\infty}(\sum_{n=0}^{k}(n\ \mathbb{P}\ \{s \mid fst(X\ s) = n\})) = p)\ \wedge\ (\lim_{k\to\infty}(\sum_{n=0}^{k}(n\ \mathbb{P}\ \{s \mid fst(Y\ s) = n\})) = q)$$
$$\Rightarrow\ (\lim_{k\to\infty}(\sum_{n=0}^{k}(n\ \mathbb{P}\{s \mid fst(X\ s)\ +\ fst(Y\ (snd\ (X\ s)) = n\})))) = (p + q)) \tag{3.9}$$

The set in the conclusion of the above implication can be proved to be equal to the countable union of a sequence of events as follows

$$\begin{aligned}
&\forall\, \mathtt{X}\ \mathtt{Y}\ \mathtt{n}.\ \mathtt{X} \in \mathtt{indep_fn} \ \wedge\ \mathtt{Y} \in \mathtt{indep_fn} \\
&\quad \Rightarrow \{\mathtt{s} \mid \mathtt{fst}(\mathtt{X}\ \mathtt{s}) + \mathtt{fst}(\mathtt{Y}\ (\mathtt{snd}(\mathtt{X}\ \mathtt{s}))) = \mathtt{n}\} \\
&\qquad = \bigcup_{\mathtt{i} \leq \mathtt{n}} \{\mathtt{s} \mid (\mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{i}) \wedge (\mathtt{fst}(\mathtt{Y}(\mathtt{snd}(\mathtt{X}\ \mathtt{s}))) = \mathtt{n} - \mathtt{i})\}
\end{aligned} \tag{3.10}$$

using the properties verified in the HOL theory of sets. All the events in the above sequence of events are mutually exclusive. Thus, Equation (3.10) along with the additive law of probability, given in the HOL theory of probability, can be used to simplify the subgoal, given in Equation (3.9), as follows.

$$\begin{aligned}
&(\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}}(\mathtt{n}\ \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{n}\})) = \mathtt{p}) \ \wedge\ (\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}}(\mathtt{n}\ \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}(\mathtt{Y}\ \mathtt{s}) = \mathtt{n}\})) = \mathtt{q}) \\
&\Rightarrow \lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}}(\mathtt{n}\ \sum_{\mathtt{i}=0}^{\mathtt{n}+1}\mathbb{P}\{\mathtt{s} \mid (\mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{i}) \wedge (\mathtt{fst}(\mathtt{Y}(\mathtt{snd}(\mathtt{X}\ \mathtt{s}))) = \mathtt{n} - \mathtt{i})\})) = (\mathtt{p} + \mathtt{q})
\end{aligned} \tag{3.11}$$

Next, we found a real sequence that is easier to handle and has the same limit value as the real sequence given in the conclusion of the above implication.

$$\begin{aligned}
&(\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}}\mathtt{n}(\sum_{\mathtt{i}=0}^{\mathtt{n}+1}\mathbb{P}\{\mathtt{s} \mid (\mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{i}) \ \wedge\ (\mathtt{fst}(\mathtt{Y}\ (\mathtt{snd}(\mathtt{X}\ \mathtt{s}))) = \mathtt{n} - \mathtt{i})\}))) = \\
&(\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{a}=0}^{\mathtt{k}}\sum_{\mathtt{b}=0}^{\mathtt{k}}(\mathtt{a} + \mathtt{b})(\mathbb{P}\{\mathtt{s} \mid (\mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{a}) \ \wedge\ (\mathtt{fst}(\mathtt{Y}\ (\mathtt{snd}(\mathtt{X}\ \mathtt{s}))) = \mathtt{b})\})))
\end{aligned} \tag{3.12}$$

Using this new real sequence and rearranging the terms based on summation properties given in the HOL theories of *real* numbers, we can rewrite the subgoal, given in Equation (3.11), as follows.

$$(\lim_{k\to\infty}(\sum_{n=0}^{k}(n\ \mathbb{P}\ \{s \mid \mathtt{fst(X\ s)} = n\})) = p)\ \wedge\ (\lim_{k\to\infty}(\sum_{n=0}^{k}(n\ \mathbb{P}\ \{s \mid \mathtt{fst(Y\ s)} = n\})) = q)$$
$$\Rightarrow (\lim_{k\to\infty}(\sum_{a=0}^{k}\sum_{b=0}^{k} a(\mathbb{P}\{s \mid (\mathtt{fst(X\ s)} = a)\ \wedge\ (\mathtt{fst(Y\ (snd(X\ s)))} = b)\})) = p)\ \wedge$$
$$(\lim_{k\to\infty}(\sum_{a=0}^{k}\sum_{b=0}^{k} b(\mathbb{P}\{s \mid (\mathtt{fst(X\ s)} = a)\ \wedge\ (\mathtt{fst(Y\ (snd(X\ s)))} = b)\})) = q) \tag{3.13}$$

The two limit expressions in the conclusion of the above implication can now be proved to be *True* using some elementary properties in the HOL theories of probability, sets and *real* numbers, which also concludes the proof for Lemma 3.2.

The next step is to generalize Lemma 3.2 to verify the linearity of expectation property, given in Equation (3.7), using induction. For this purpose, we define a function that models the summation of a list of discrete random variables.

Definition 3.4: *Summation of n Random Variables*

```
⊢ (sum_rv_lst [] = unit 0) ∧
    ∀ h t.  (sum_rv_lst (h::t) =
      bind h (λa.  bind (sum_rv_lst t) (λb.  unit (a + b))))
```

The function, `sum_rv_lst`, accepts a list of random variables and returns their sum as a single random variable. Just like the function, `sum_two_rv`, the function `sum_rv_lst` also preserves *strong function independence*, if all random variables in the given list preserve it. This property can be verified using the fact that it accesses the infinite Boolean sequence using the `unit` and `bind` primitives only.

Lemma 3.3: *sum_rv_lst Preserves Strong Function Independence*

```
⊢ ∀ L. (∀ R. (mem R L) ⇒ R ∈ indep_fn)
    ⇒ ((sum_rv_lst L) ∈ indep_fn)
```

where the predicate mem is defined in the HOL list theory and returns *True* if its first argument is an element of the list that it accepts as the second argument.

Now, the linearity of expectation property for n discrete random variables, which preserve *strong function independence* and for which the infinite summation in the expectation definition converges, can be stated in HOL as follows

Theorem 3.2: *Linearity of Expectation Property*

$$\vdash \forall \texttt{L}.\ (\forall \texttt{R}.\ (\texttt{mem R L}) \Rightarrow ((\texttt{R} \in \texttt{indep\_fn}) \wedge (\texttt{summable}\ (\lambda \texttt{n}.\ \ \texttt{n}\ \mathbb{P}\{\texttt{s} \mid \texttt{fst(R s)} = \texttt{n}\}))))$$
$$\Rightarrow (\texttt{expec (sum\_rv\_lst L)} = \sum_{n=0}^{\texttt{length L}} \texttt{(expec (el (length L - (n+1)) L))})$$

where the function length, defined in the HOL list theory, returns the length of its list argument and the function el, also defined in the list theory, accepts a positive integer number, say n, and a list and returns the n^{th} element of the given list. Thus, the LHS of Theorem 3.2 represents the expectation of the summation of a list L of random variables. Whereas, the RHS represents the summation of the expectations of all elements in the same list L. Theorem 3.2 can be proved by applying induction on the list argument of the function sum_rv_lst, and simplifying the subgoals using Lemmas 3.2 and 3.3.

Expectation of a Discrete Random Variable Multiplied by a Constant

$$Ex[aR] = aEx[R] \tag{3.14}$$

where R is a discrete random variable that attains values in the positive integers only and a is a positive integer. This property can be expressed in HOL for a random variable R that preserves *strong function independence* and has a *well-defined* expected value as follows

Theorem 3.3: *Expectation of a Discrete Random Variable Multiplied by a Constant*

```
⊢ ∀ R a.  R ∈ indep_fn ∧
   summable(λn.  n ℙ{s | fst(R s) = n})
⇒ expec (bind R (λm.  unit (a m))) = a (expec R)
```

The HOL proof proceeds by first performing case analysis on the variable a. For the case when a is 0, the RHS of the proof goal becomes 0. Whereas, using the definition of expectation, the LHS reduces to the expression

$$\lim_{k\to\infty}(\sum_{n=0}^{k} n\ \mathbb{P}\{s|\ 0\ =\ n\}) \tag{3.15}$$

which is also equal to 0 as $\forall n.n\ \mathbb{P}\{s\ |\ 0 = n\} = 0$, since $\forall n.\ \ 0 < n \Rightarrow \mathbb{P}\ \{s\ |\ 0 = n\}=0$. On the other hand, when a is not equal to 0, i.e., $(0 < a)$, the proof goal may be simplified as follows

$$\lim_{k\to\infty}(\sum_{n=0}^{k} n\ \mathbb{P}\{s|\ a\ fst(R\ s) = n\}) = a \lim_{k\to\infty}(\sum_{n=0}^{k} n\ \mathbb{P}\{s|\ a\ fst(R\ s) = a\ n\}) \tag{3.16}$$

using the definition of expectation and the multiplication cancelation property of positive integers. Next, we proved in HOL that

$$\forall k.(\sum_{n=0}^{k} n\ \mathbb{P}\{s|\ a\ fst(R\ s) = n\}) = a(\sum_{n=0}^{B(k)} n\ \mathbb{P}\{s|\ a\ fst(R\ s) = a\ n\}) \tag{3.17}$$

where B(k) = if (k MOD a = 0) then (k DIV a) else ((k DIV a) + 1) and MOD and DIV represent the *modulo* and *division* functions for positive integers in HOL. This allows us to rewrite our proof goal as follows

$$\lim_{k\to\infty} \mathtt{a}\ (\sum_{n=0}^{B(k)} \mathtt{n}\ \mathbb{P}\{\mathtt{s}|\ \mathtt{a}\ \mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{a}\ \mathtt{n}\}) = \mathtt{a} \lim_{k\to\infty} (\sum_{n=0}^{k} \mathtt{n}\ \mathbb{P}\{\mathtt{s}|\ \mathtt{a}\ \mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{a}\ \mathtt{n}\}) \tag{3.18}$$

which can be proved using the properties of limit of a real sequence in HOL [32], since both of the real sequences in the above equation converge to the same value as the value of k becomes very very large. This concludes the proof of the expectation property given in Theorem 3.3.

Expectation of a Discrete Random Variable Added and Multiplied by Constants

$$Ex[a + bR] = a + bEx[R] \tag{3.19}$$

This property allows us to express the expectation value of a positive integer valued random variable R added and multiplied by two positive integers a and b, respectively, in terms of the expectation of the random variable R. It can be expressed in HOL for a random variable R that preserves *strong function independence* and has a *well-defined* expected value as follows.

Theorem 3.4: *Expectation of a Discrete Random Variable Added and Multiplied by Constants*

```
⊢ ∀ R a b.  R ∈ indep_fn ∧
  summable(λn.  n ℙ{s | fst(R s) = n})
⇒ expec (bind R (λm.  unit (a + b m))) =
  a + b (expec R)
```

Theorem 3.4 can be proved in HOL using the expectation properties, given in Theorems 3.1, 3.2 and 3.3.

3.3 Variance for Discrete Random Variables

In this section, we utilize the formal definition of expectation of a function of a random variable, developed in Section 3.2, to define a variance function for discrete random variables that attain values in positive integers only. We later utilize this definition to verify a couple of classical variance properties in HOL. This is the second step in the proposed formalization infrastructure for reasoning about statistical properties, given in Figure 3.1.

3.3.1 Formal Specification of Variance

In the field of probabilistic analysis, it is often desirable to summarize the essential properties of distribution of a random variable by certain suitably defined measures. In the previous section, we formalized one such measure, i.e., the expectation, which yields the weighted average of the possible values of a random variable. Quite frequently, along with the average value, we are also interested in finding how typical is the average value or in other words the chances of observing an event far from the average. One possible way to measure the variation, or spread, of these values is to consider the quantity $Ex[|R - Ex[R]|]$, where $||$ denote the *abs* function. However, it turns out to be mathematically inconvenient to deal with this quantity, so a more tractable quantity called *variance* is usually considered, which returns the expectation of the square of the difference between R and its expectation [7].

$$Var[R] = Ex[(R - Ex[R])^2] \tag{3.20}$$

Now, we formalize this definition of variance in HOL for the case of discrete random variables that can attain values in the positive integers only. For this purpose, we utilize the definitions of expectation, given in Definitions 3.1 and 3.2.

Definition 3.5: *Variance of a Discrete Random Variable*

$\vdash \forall$ R. variance R = expec_fn (λn. (n - expec R)^2) R

The function, variance, accepts a discrete random variable R that attains values in the positive integers only and returns its variance as a *real* number.

3.3.2 Verification of Variance Properties

In this section, we prove two of the most significant and widely used properties of the variance function [61]. These properties not only verify the correctness of our definition but also play a vital role in verifying the variance properties of discrete random variables as will be seen in Section 3.5.

Variance in Terms of Moments

$$Var[R] = Ex[R^2] - (Ex[R])^2 \tag{3.21}$$

where R is a discrete random variable that can attain values in the positive integers only. This alternative definition of variance is much easier to work with than the previous one and thus aids significantly in the process of verifying variance properties for discrete random variables. This property can be stated in HOL using the formal definition of variance and expectation as follows.

Theorem 3.5: *Variance in Terms of Moments*

$\vdash \forall$ R. R $\in$ indep_fn $\wedge$

summable(λn. n $\mathbb{P}${s | fst(R s) = n}) $\wedge$

summable(λn. n^2 $\mathbb{P}${s | fst(R s) = n})

$\Rightarrow$ (variance R = expec_fn (λn. n^2) R - (expec R)^2)

The assumption in Theorem 3.5 ensures that the random variable R preserves the *strong function independence* and its expectation and second moment are *well-defined.* The theorem can be proved by using the function definitions of `expec_fn`, `expec` and `variance` along with some arithmetic reasoning and properties from the HOL *real* number theories.

Linearity of Variance for Independent Discrete Random Variables

$$Var[\sum_{i=1}^{n} R_i] = \sum_{i=1}^{n} Var[R_i] \tag{3.22}$$

where R represents a sequence of n independent discrete random variables. Like the linearity of expectation property, the linearity of variance property also allows us to verify the variance properties of probabilistic systems involving multiple random variables without going into the complex verification of their joint probability distribution properties.

The proof steps for the linearity of variance property are quite similar to the proof steps for the linearity of expectation property. We split the verification task in two major steps. Firstly, we verify the property for two discrete random variables and then extend the results by induction to prove the general case. The linearity of variance property can be defined for any two independent discrete random variables X and Y as follows

$$Var[X + Y] = Var[X] + Var[Y] \tag{3.23}$$

Using the function `sum_two_rv`, given in Definition 3.3, the linearity of variance property for two independent discrete random variables, which attain values in the positive integers only, preserve the *strong function independence* and have *well-defined* expectation and second moment, can be stated in HOL as follows.

Lemma 3.4: *Linearity of Variance for Two Discrete Random Variables*

$$
\begin{aligned}
&\vdash \forall\ \mathtt{X}\ \mathtt{Y}.\ \mathtt{X} \in \mathtt{indep\_fn} \wedge \mathtt{Y} \in \mathtt{indep\_fn} \wedge \\
&\quad (\mathtt{summable}(\lambda \mathtt{n}.\ \ \mathtt{n}\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{n}\})) \wedge \\
&\quad (\mathtt{summable}(\lambda \mathtt{n}.\ \ \mathtt{n}\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}(\mathtt{Y}\ \mathtt{s}) = \mathtt{n}\})) \wedge \\
&\quad (\mathtt{summable}(\lambda \mathtt{n}.\ \ \mathtt{n}^2\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{n}\})) \wedge \\
&\quad (\mathtt{summable}(\lambda \mathtt{n}.\ \ \mathtt{n}^2\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}(\mathtt{Y}\ \mathtt{s}) = \mathtt{n}\})) \\
&\Rightarrow (\mathtt{variance}\ (\mathtt{sum\_two\_rv}\ \mathtt{X}\ \mathtt{Y}) = \mathtt{variance}\ \mathtt{X} + \mathtt{variance}\ \mathtt{Y})
\end{aligned}
$$

Rewriting the above theorem with the definitions of the functions `variance`, `expec_fn`, `expec` and `summable`, simplifying it with some infinite summation properties and Theorem 3.2 and removing the monad notation, we reach the following subgoal.

$$
\begin{aligned}
&(\lim_{k\to\infty} \sum_{n=0}^{k} (\mathtt{n}\ \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{n}\}) = \mathtt{p}) \wedge (\lim_{k\to\infty} \sum_{n=0}^{k} (\mathtt{n}\ \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}(\mathtt{Y}\ \mathtt{s}) = \mathtt{n}\}) = \mathtt{q}) \wedge \\
&(\lim_{k\to\infty} \sum_{n=0}^{k} (\mathtt{n}^2\ \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{n}\}) = \mathtt{r}) \wedge (\lim_{k\to\infty} \sum_{n=0}^{k} (\mathtt{n}^2\ \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}(\mathtt{Y}\ \mathtt{s}) = \mathtt{n}\}) = \mathtt{t}) \wedge \\
&(\lim_{k\to\infty} (\sum_{n=0}^{k} ((\mathtt{n} - \mathtt{expec}\ \mathtt{X})^2\ \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}(\mathtt{X}\ \mathtt{s}) = \mathtt{n}\})) = \mathtt{u}) \wedge \\
&(\lim_{k\to\infty} (\sum_{n=0}^{k} ((\mathtt{n} - \mathtt{expec}\ \mathtt{Y})^2\ \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}(\mathtt{Y}\ \mathtt{s}) = \mathtt{n}\})) = \mathtt{v}) \\
&\Rightarrow \lim_{k\to\infty} \sum_{n=0}^{k} ((\mathtt{n} - (\mathtt{expec}\ \mathtt{X} + \mathtt{expec}\ \mathtt{Y}))^2)\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}(\mathtt{X}\ \mathtt{s}) + \mathtt{fst}(\mathtt{Y}\ (\mathtt{snd}\ (\mathtt{X}\ \mathtt{s}))) = \mathtt{n}\} \\
&= \ (\mathtt{u} + \mathtt{v})
\end{aligned}
\tag{3.24}
$$

Using the uniqueness of the limit value of a real sequence, and some properties of summation of real sequences, it can be proved in a straight forward manner that $\mathtt{u} = \mathtt{r} - \mathtt{p}^2$ and $\mathtt{v} = \mathtt{t} - \mathtt{q}^2$ under the given assumptions in the above subgoal. This allows us to rewrite the above subgoal as follows.

$$
\begin{aligned}
&(\lim_{k\to\infty}\sum_{n=0}^{k}(n\ \mathbb{P}\ \{s \mid \mathtt{fst}(X\ s) = n\}) = p) \wedge\ (\lim_{k\to\infty}\sum_{n=0}^{k}(n\ \mathbb{P}\ \{s \mid \mathtt{fst}(Y\ s) = n\}) = q) \wedge \\
&(\lim_{k\to\infty}\sum_{n=0}^{k}(n^2\ \mathbb{P}\ \{s \mid \mathtt{fst}(X\ s) = n\}) = r) \wedge\ (\lim_{k\to\infty}\sum_{n=0}^{k}(n^2\ \mathbb{P}\ \{s \mid \mathtt{fst}(Y\ s) = n\}) = t) \\
&\Rightarrow \lim_{k\to\infty}\sum_{n=0}^{k}((n^2 - 2(p+q)n + (p+q)^2)\ \mathbb{P}\{s \mid \mathtt{fst}(X\ s)\ +\ \mathtt{fst}(Y\ (\mathtt{snd}\ (X\ s))) = n\}) \\
&= (r + t - (p^2 + q^2))
\end{aligned}
\tag{3.25}
$$

Next we split the real sequence of the conclusion, in the above subgoal, in a sum of three real sequences, corresponding to the terms n^2, $-2(p+q)n$ and $(p+q)^2$ found on the LHS. Now, using the results of Theorem 3.2 along with some probability laws, it can be shown that the second and third sequences out of these three converge to $(-2(p+q)(p+q))$ and $(p+q)^2$, respectively. This allows us to rewrite the subgoal of Equation (3.25) as follows.

$$
\begin{aligned}
&(\lim_{k\to\infty}(\sum_{n=0}^{k} n\ \mathbb{P}\ \{s \mid \mathtt{fst}(X\ s) = n\}) = p) \wedge\ (\lim_{k\to\infty}\sum_{n=0}^{k}(n\ \mathbb{P}\ \{s \mid \mathtt{fst}(Y\ s) = n\}) = q) \wedge \\
&(\lim_{k\to\infty}\sum_{n=0}^{k}(n^2\ \mathbb{P}\ \{s \mid \mathtt{fst}(X\ s) = n\}) = r) \wedge\ (\lim_{k\to\infty}\sum_{n=0}^{k}(n^2\ \mathbb{P}\ \{s \mid \mathtt{fst}(Y\ s) = n\}) = t) \\
&\Rightarrow \lim_{k\to\infty}\sum_{n=0}^{k}((n^2)\ \mathbb{P}\{s \mid \mathtt{fst}(X\ s)\ +\ \mathtt{fst}(Y\ (\mathtt{snd}\ (X\ s))) = n\}) = (r + t + 2pq)
\end{aligned}
\tag{3.26}
$$

Just like the proof of the linearity of expectation property, we replace the real sequence in the conclusion of the above subgoal by a real sequence that is simpler to handle and shares the same limit value as this one, under the given assumptions.

$$(\lim_{k\to\infty}(\sum_{n=0}^{k} n^2(\mathbb{P}\{s \mid \texttt{fst(X s)} + \texttt{fst(Y (snd(X s)))} = n\}))) =$$

$$(\lim_{k\to\infty}(\sum_{a=0}^{k}\sum_{b=0}^{k}(a^2+ab)(\mathbb{P}\{s \mid (\texttt{fst(X s)} = a) \wedge (\texttt{fst(Y (snd(X s)))} = b)\} +$$

$$\mathbb{P}\{s \mid (\texttt{fst(X s)} = b) \wedge (\texttt{fst(Y (snd(X s)))} = a)\})))$$

(3.27)

The subgoal given in Equation (3.26) can now be proved using the above result and some arithmetic reasoning in HOL, which concludes the proof of Lemma 3.4.

The next step is to generalize Lemma 3.4 to verify the linearity of variance property for n discrete random variables (Equation (3.22)), which can be stated in HOL as follows.

Theorem 3.6: *Linearity of Variance Property*

```
⊢ ∀ L. (∀ R. (mem R L) ⇒ ((R ∈ indep_fn) ∧
  (summable (λn.  n ℙ{s | fst(R s) = n}))∧
  (summable (λn.  n² ℙ{s | fst(R s) = n}))))
⇒ (variance (sum_rv_lst L) =
  ∑(length L) n=0 (variance (el (length L - (n+1)) L)))
```

Theorem 3.6 can be proved by applying induction on the list argument of the function `sum_rv_lst`, and simplifying the subgoals using Lemmas 3.3 and 3.4.

3.4 Markov and Chebyshev's Inequalities

In this section, we present the verification of Markov and Chebyshev's inequalities in HOL using the formal definitions of expectation and variance, given in the last two

sections. The verification of Markov and Chebyshev's inequalities form the third and fourth steps, respectively, in the proposed formalization infrastructure for reasoning about statistical properties, given in Figure 3.1.

3.4.1 Verification of Markov's Inequality

Markov's inequality utilizes the definition of expectation to obtain a weak tail bound.

$$Pr(X \geq a) \leq \frac{Ex[X]}{a} \tag{3.28}$$

It can be expressed in HOL for a discrete random variable that preserves *strong function independence*, has a *well-defined* expected value and attains values in positive integers only as follows.

Theorem 3.7: *Markov's Inequality*

$$\begin{aligned} \vdash\ & \forall\ \mathtt{R\ a.}\ \ (\mathtt{0} < \mathtt{a}) \wedge (\mathtt{R} \in \mathtt{indep\_fn}) \wedge \\ & (\mathtt{summable}(\lambda \mathtt{n.}\ \ \mathtt{n}\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}\ (\mathtt{R\ s}) = \mathtt{n}\})) \\ \Rightarrow\ & \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}\ (\mathtt{R\ s}) \geq \mathtt{a}\} \leq \frac{(\mathtt{expec\ R})}{\mathtt{a}} \end{aligned}$$

where a represents a *real* number.

We proceed with the proof of Theorem 3.7 in HOL by rewriting its proof goal with the definition of expectation, given in Definition 3.2,

$$\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) \geq \mathtt{a}\} \leq \frac{\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}}(\mathtt{n}\ \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) = \mathtt{n}\}))}{\mathtt{a}} \tag{3.29}$$

Now, the set on the LHS of the above inequality can be expressed as follows

$$\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) \geq \mathtt{a}\} = \{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) \geq \lceil \mathtt{a} \rceil\} \tag{3.30}$$

where $\lceil \mathtt{x} \rceil$ denotes the ceiling of x, which represents the closest integer for a real number x that is greater than or equal to x. The above equation is *True* because the random variable R acquires values in positive integers only. Thus, all possible values of the random variable R that are greater than a are also greater than or equal to $\lceil a \rceil$ and vice-versa. Equation (3.30) can now be used, along with some arithmetic reasoning in HOL, to rewrite our proof goal (Equation (3.29)) as follows

$$\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) \geq \lceil \mathtt{a} \rceil\} \leq \lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}}(\frac{\mathtt{n}}{\lceil \mathtt{a} \rceil}\ \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\})) \tag{3.31}$$

Next, we use the complement law of the probability function $P(\overline{A}) = 1 - P(A)$, which is formally verified in [36], to rewrite the LHS of the above inequality as $1 - \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) < \lceil \mathtt{a} \rceil\}$. The expression $\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) < \lceil \mathtt{a} \rceil\}$ can be further simplified using the additive law of probability $P(A \cup B) = P(A) + P(B)$, also verified in [36], as $\sum_{\mathtt{n}=0}^{\lceil \mathtt{a} \rceil} \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\}$. This simplification allows us to rewrite the subgoal, given in Equation (3.31), as follows

$$1 - \sum_{\mathtt{n}=0}^{\lceil \mathtt{a} \rceil} \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\} \leq \lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}}(\frac{\mathtt{n}}{\lceil \mathtt{a} \rceil}\ \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\})) \tag{3.32}$$

It can be proved in HOL that $\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}} \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\}) = 1$, which allows us to rewrite the LHS of the above inequality as the limit value of the *real* sequence $(\lambda \mathtt{k}.\ \sum_{\mathtt{n}=\lceil \mathtt{a} \rceil}^{\mathtt{k}} \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\})$ as k approaches infinity. Similarly, the expression $\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=\lceil \mathtt{a} \rceil}^{\mathtt{k}}(\frac{\mathtt{n}}{\lceil \mathtt{a} \rceil}\ \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\}))$ can be proved to be less than or equal to the RHS of the above inequality, which allows us to rewrite the subgoal, given in Equation (3.32), as follows

$$\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=\lceil \mathtt{a} \rceil}^{\mathtt{k}} \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\}) \leq \lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=\lceil \mathtt{a} \rceil}^{\mathtt{k}}(\frac{\mathtt{n}}{\lceil \mathtt{a} \rceil}\ \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\})) \tag{3.33}$$

Now, we verify in HOL that for all values of k, the expression $(\sum_{\mathtt{n}=\lceil \mathtt{a} \rceil}^{\mathtt{k}} \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\})$, found on the LHS of the above inequality, is less than or equal to the expression $(\sum_{\mathtt{n}=\lceil \mathtt{a} \rceil}^{\mathtt{k}} (\frac{\mathtt{n}}{\lceil \mathtt{a} \rceil}\ \mathbb{P}\ \{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mathtt{n}\}))$, found on its RHS. This reasoning allows us to prove the limit relationship, given in Equation (3.33), between these expressions using the properties of limit of a real sequence [32] and thus conclude the proof of Markov's inequality, given in Theorem 3.7.

3.4.2 Verification of Chebyshev's Inequality

Chebyshev's inequality utilizes the variance and the expectation characteristics to derive a significantly stronger tail bound than the one obtained by Markov's inequality.

$$Pr(|X - Ex[X]| \geq a) \leq \frac{Var[X]}{a^2} \tag{3.34}$$

We verify the Chebyshev's inequality by first verifying one of its variants [7]

$$Pr(|X - Ex[X]| \geq a\sigma_X) \leq \frac{1}{a^2} \tag{3.35}$$

where σ_X denotes the standard deviation function, which returns the square root of variance for a random variable X. This property can be expressed in HOL as follows

Lemma 3.5: *Chebyshev's Inequality in terms of Standard Deviation*

$$\begin{aligned}
&\vdash \forall\ \mathtt{R}\ \mathtt{a}.\ \ (\mathtt{0} < \mathtt{a}) \wedge (\mathtt{0} < \mathtt{variance\ R}) \wedge (\mathtt{R} \in \mathtt{indep_fn}) \wedge \\
&\quad (\mathtt{summable}(\lambda \mathtt{n}.\ \ \mathtt{n}\ \mathbb{P}\{\mathtt{s}\ |\ \mathtt{fst}\ (\mathtt{R}\ \mathtt{s}) = \mathtt{n}\})) \wedge \\
&\quad (\mathtt{summable}(\lambda \mathtt{n}.\ \ \mathtt{n}^2\ \mathbb{P}\{\mathtt{s}\ |\ \mathtt{fst}\ (\mathtt{R}\ \mathtt{s}) = \mathtt{n}\})) \\
&\Rightarrow \mathbb{P}\ \{\mathtt{s}\ |\ \mathtt{abs}\ (\mathtt{fst}\ (\mathtt{R}\ \mathtt{s})\ -\ \mathtt{expec\ R}) \geq \mathtt{a\ std_dev\ R}\} \leq \tfrac{1}{\mathtt{a}^2}
\end{aligned}$$

for a discrete random variable that preserves *strong function independence*, has a *well-defined* expected value and attains values in positive integers only as follows.

Whereas, the HOL function `abs`, defined in [32], returns the absolute value of a real number. The HOL function `std_dev`, defined as follows, returns the square root of the variance for a discrete random variable, which attains values in positive integers only

Definition 3.5: *Standard Deviation of a Discrete Random Variable*

```
⊢ ∀ R. std_dev R = sqrt (variance R)
```

where the HOL function `sqrt`, defined in [32], returns the square root of a *real* number. It is important to note that we have used the assumption `0 < variance R` in Lemma 3.5 because variance is a positive quantity and there is no point in calculating the tail distribution bound for random variables with variance equal to 0.

We proceed with the proof of Lemma 3.5 in HOL by splitting its proof goal, using the transitivity property of $\leq$, i.e., $(a \leq b \wedge b \leq c \Rightarrow a \leq c)$, into two subgoals as follows

$$\begin{aligned} &\mathbb{P}\{\mathtt{s}|\mathtt{abs}(\mathtt{fst}(\mathtt{R\ s}) - \mu_{\mathtt{R}}) \geq \mathtt{a}\sigma_{\mathtt{R}}\} \leq \\ &\quad \mathbb{P}\{s|(\mathtt{fst}(\mathtt{R\ s})) \geq \mu_{\mathtt{R}} + \mathtt{a}\sigma_{\mathtt{R}}\} + \mathbb{P}\{\mathtt{s}|(\mathtt{fst}(\mathtt{R\ s})) \leq \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}}\} \end{aligned} \tag{3.36}$$

$$\mathbb{P}\{\mathtt{s}|(\mathtt{fst}(\mathtt{R\ s})) \geq \mu_{\mathtt{R}} + \mathtt{a}\sigma_{\mathtt{R}}\} + \mathbb{P}\{\mathtt{s}|(\mathtt{fst}(\mathtt{R\ s})) \leq \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}}\} \leq \frac{1}{\mathtt{a}^2} \tag{3.37}$$

where the symbols μ_R and σ_R denote the HOL functions for expectation and standard deviation for a random variable R.

The sets $\{\mathtt{s}|(\mathtt{fst}(\mathtt{R\ s})) \geq \mu_{\mathtt{R}} + \mathtt{a}\sigma_{\mathtt{R}}\}$ and $\{\mathtt{s}|(\mathtt{fst}(\mathtt{R\ s})) \leq \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}}\}$, found on the RHS of Equation (3.36), can be proved to be disjoint because the term $a\sigma_R$ is greater than 0. Thus, using the additive law of probability, we can rewrite Equation (3.36) as follows.

$$\mathbb{P}\{\mathtt{s}|\mathtt{abs}(\mathtt{fst}(\mathtt{R}\ \mathtt{s}) - \mu_{\mathtt{R}}) \geq \mathtt{a}\sigma_{\mathtt{R}}\} \leq$$
$$\mathbb{P}\{\mathtt{s}|(\mathtt{fst}(\mathtt{R}\ \mathtt{s})) \geq \mu_{\mathtt{R}} + \mathtt{a}\sigma_{\mathtt{R}}\} \cup \{\mathtt{s}|(\mathtt{fst}(\mathtt{R}\ \mathtt{s})) \leq \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}}\} \tag{3.38}$$

Now, using arithmetic reasoning, it can be proved in HOL that the set on the LHS of the inequality in Equation (3.38) is a subset of the set that appears on the RHS. This allows us to verify Equation (3.38) using the increasing probability law $P(A \subseteq B) \Rightarrow P(A) \leq P(B)$ and thus conclude the proof of Equation (3.36).

The next step in the verification of Lemma 3.5 is to prove the inequality given in Equation (3.37). We proceed in this direction by replacing the terms on the LHS of the inequality in Equation (3.37) as follows

$$\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) \geq \lceil \mu_{\mathtt{R}} + \mathtt{a}\sigma_{\mathtt{R}} \rceil\} +$$
$$\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) < \lceil \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}} \rceil\} \cup \{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}}\} \leq \frac{1}{\mathtt{a}^2} \tag{3.39}$$

The above step is valid due to the transitivity property of $\leq$, as the sum of the terms on the LHS of the inequality in Equation (3.39) is greater than the sum of the terms on the LHS of the inequality in Equation (3.37). This is the case because of the increasing probability law and the fact that the set $\{\mathtt{s}|(\mathtt{fst}(\mathtt{R}\ \mathtt{s})) \geq \mu_{\mathtt{R}} + \mathtt{a}\sigma_{\mathtt{R}}\}$ is a subset of the set $\{\mathtt{s}|(\mathtt{fst}(\mathtt{R}\ \mathtt{s})) \geq \lceil \mu_{\mathtt{R}} + \mathtt{a}\sigma_{\mathtt{R}} \rceil\}$ and the set $\{\mathtt{s}|(\mathtt{fst}(\mathtt{R}\ \mathtt{s})) \leq \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}}\}$ is a subset of the set $\{\mathtt{s}|(\mathtt{fst}(\mathtt{R}\ \mathtt{s})) < \lceil \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}} \rceil\} \cup \{\mathtt{s}|(\mathtt{fst}(\mathtt{R}\ \mathtt{s})) = \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}}\}$. Next, we can rewrite Equation (3.39), using arithmetic reasoning, as follows

$$\sigma_{\mathtt{R}}^2\mathtt{a}^2(\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) \geq \lceil \mu_{\mathtt{R}} + \mathtt{a}\sigma_{\mathtt{R}} \rceil\} +$$
$$\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) < \lceil \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}} \rceil\} \cup \{\mathtt{s}|\mathtt{fst}(\mathtt{R}\ \mathtt{s}) = \mu_{\mathtt{R}} - \mathtt{a}\sigma_{\mathtt{R}}\}) \leq \sigma_{\mathtt{R}}^2 \tag{3.40}$$

where the symbol $\sigma_{\mathtt{R}}^2$ denotes the variance of random variable R. In order to prove the

above inequality we try to verify the following relationship regarding its second term on the LHS.

$$\sigma_R^2 a^2(\mathbb{P}\{s|fst(R\ s) < \lceil \mu_R - a\sigma_R \rceil\} \cup \{s|fst(R\ s) = \mu_R - a\sigma_R\}) \leq \sum_{n=0}^{\lceil \mu_R + a\sigma_R \rceil} (n - \mu_R)^2 \mathbb{P}\{s|fst(R\ s) = n\} \tag{3.41}$$

The two sets, in the union, on the LHS of the above inequality are disjoint, which allows us to rewrite the expression on the LHS as a sum of two probabilities, using the additive law of probability. The first probability term, out of these two terms, can then be expressed as $\sum_{n=0}^{\lceil \mu_R - a\sigma_R \rceil} \sigma_R^2 a^2 \mathbb{P}\{s|fst(R\ s) = n\}$ using the additive law of probability. Whereas, the expression on the RHS of the above inequality can be split into the sum of two terms, using the definition of the summation function in HOL, as follows

$$\sum_{n=0}^{\lceil \mu_R - a\sigma_R \rceil} \sigma_R^2 a^2 \mathbb{P}\{s|fst(R\ s) = n\} + \sigma_R^2 a^2 \mathbb{P}\{s|fst(R\ s) = \mu_R - a\sigma_R\}) \leq \sum_{n=0}^{\lceil \mu_R - a\sigma_R \rceil} (n - \mu_R)^2 \mathbb{P}\{s|fst(R\ s) = n\} + \sum_{n=\lceil \mu_R - a\sigma_R \rceil}^{\lceil \mu_R + a\sigma_R \rceil - \lceil \mu_R - a\sigma_R \rceil} (n - \mu_R)^2 \mathbb{P}\{s|fst(R\ s) = n\} \tag{3.42}$$

Now the above inequality can be proved in HOL, as both the terms on the LHS of the above equation are less than or equal to the corresponding two terms on the RHS. This result allows us to rewrite the inequality, given in Equation (3.40), as follows

$$\sigma_R^2 a^2 \mathbb{P}\{s|fst(R\ s) \geq \lceil \mu_R + a\sigma_R \rceil\} + \sum_{n=0}^{\lceil \mu_R + a\sigma_R \rceil} (n - \mu_R)^2 \mathbb{P}\{s|fst(R\ s) = n\} \leq \sigma_R^2 \tag{3.43}$$

using the transitivity property of $\leq$. Using the definition of variance and rearranging

the terms, based on arithmetic reasoning, the above equation can be rewritten as follows

$$\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) \geq \lceil \mu_{\mathtt{R}} + \mathtt{a}\sigma_{\mathtt{R}} \rceil\} \leq$$
$$\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}} \frac{(\mathtt{n}-\mu_{\mathtt{R}})^2}{\sigma_{\mathtt{R}}^2\mathtt{a}^2}\ \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) = \mathtt{n}\}) - \sum_{\mathtt{n}=0}^{\lceil \mu_{\mathtt{R}}+\mathtt{a}\sigma_{\mathtt{R}} \rceil} \frac{(\mathtt{n}-\mu_{\mathtt{R}})^2}{\sigma_{\mathtt{R}}^2\mathtt{a}^2}\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) = \mathtt{n}\} \tag{3.44}$$

The probability term on the LHS of the above inequality can be expressed in terms of the limit of the *real* sequence $(\lambda \mathtt{k}.\ \sum_{\mathtt{n}=\lceil \mu_{\mathtt{R}}+\mathtt{a}\sigma_{\mathtt{R}} \rceil}^{\mathtt{k}} \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) = \mathtt{n}\})$, using the same reasoning as was used for the case of the proof of Markov's inequality in Equations (3.31) to (3.33). Similarly, the expression on the RHS of the above inequality can also be expressed in terms of a limit of a *real* sequence, which allows us to rewrite Equation (3.44) as follows

$$\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=\lceil \mu_{\mathtt{R}}+\mathtt{a}\sigma_{\mathtt{R}} \rceil}^{\mathtt{k}} \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) = \mathtt{n}\}) \leq \lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=\lceil \mu_{\mathtt{R}}+\mathtt{a}\sigma_{\mathtt{R}} \rceil}^{\mathtt{k}} \frac{(\mathtt{n}-\mu_{\mathtt{R}})^2}{\sigma_{\mathtt{R}}^2\mathtt{a}^2}\ \mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) = \mathtt{n}\}) \tag{3.45}$$

It can be verified in HOL that for all values of k, the expression $\sum_{\mathtt{n}=\lceil \mu_{\mathtt{R}}+\mathtt{a}\sigma_{\mathtt{R}} \rceil}^{\mathtt{k}} \mathbb{P}\{\mathtt{s}|\ \mathtt{fst}(\mathtt{R\ s}) = \mathtt{n}\}$, found on the LHS of the above inequality, is less than or equal to the expression $\sum_{\mathtt{n}=\lceil \mu_{\mathtt{R}}+\mathtt{a}\sigma_{\mathtt{R}} \rceil}^{\mathtt{k}} \frac{(\mathtt{n}-\mu_{\mathtt{R}})^2}{\sigma_{\mathtt{R}}^2\mathtt{a}^2}\mathbb{P}\{\mathtt{s}|\mathtt{fst}(\mathtt{R\ s}) = \mathtt{n}\}$, found on its RHS. This reasoning allows us to prove the limit relationship, given in Equation (3.45), between these expressions using the properties of limit of a real sequence, formalized in [32], which completes the proof of the inequality given in Equation (3.37) and thus concludes the proof of Lemma 3.5 as well.

Lemma 3.5 can now be used to verify the Chebyshev's inequality, given in Equation (3.34), in HOL as a special case when the constant a is assigned the value $\frac{\mathtt{a}}{\mathtt{std\_dev\ R}}$.

The corresponding HOL theorem can be expressed for a discrete random variable that preserves *strong function independence*, has *well-defined* first and second moments and attains values in positive integers only, as follows

Theorem 3.8 *Chebyshev's Inequality*

$$\vdash \forall\ \mathtt{R\ a.}\quad (\mathtt{0} < \mathtt{a}) \wedge (\mathtt{0} < \mathtt{variance\ R}) \wedge (\mathtt{R} \in \mathtt{indep\_fn}) \wedge$$
$$(\mathtt{summable}(\lambda \mathtt{n.}\quad \mathtt{n}\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst\ (R\ s)} = \mathtt{n}\})) \wedge$$
$$(\mathtt{summable}(\lambda \mathtt{n.}\quad \mathtt{n}^2\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst\ (R\ s)} = \mathtt{n}\}))$$
$$\Rightarrow \mathbb{P}\ \{\mathtt{s} \mid \mathtt{abs\ (fst\ (R\ s)\ -\ expec\ R)} \geq \mathtt{a}\} \leq \frac{\mathtt{variance\ R}}{\mathtt{a}^2}$$

Theorems 3.7 and 3.8 represent the HOL theorems corresponding to Markov's and Chebyshev's inequalities and thé results are found to be in good agreement with the existing theoretical paper-and-pencil counterparts given in Equations (3.28) and (3.34), respectively. These formally verified theorems allow us to reason about tail distribution bounds within the HOL theorem prover as will be demonstrated in Section 3.6.

3.5 Verification of Expectation and Variance for Discrete Random Variables

In this section, we utilize the formal definitions of expectation and variance, given in Definitions 3.2 and 3.5, respectively, to verify the expectation and variance properties of Uniform(m), Bernoulli(p), Geometric(p) and Binomial(m, p) random variables in HOL. The formally verified expectation and variance relations of these discrete random variables can in turn be used, along with the formally verified Markov and Chebyshev's inequalities presented in the last section, to formally reason about the tail distribution properties of their respective random variables.

3.5.1 Uniform(m) Random Variable

The Uniform(m) random variable assigns equal probability to each element in the set $\{0, 1, \cdots, (m-1)\}$ and thus ranges over a finite number of positive integers. A sampling algorithm for the Uniform(m) random variable has been formalized in [36] and is verified to be correct by proving the corresponding PMF property in HOL

$$\vdash \forall\ \mathtt{m\ x.}\quad \mathtt{x} < \mathtt{m} \Rightarrow \mathbb{P}\ \{\mathtt{s}\ |\ \mathtt{fst\ (prob\_unif\ m\ s)} = \mathtt{x}\} = \tfrac{1}{\mathtt{m}}$$

where `prob_unif` represents the higher-order-logic function for the Uniform(m) random variable.

Next, we formally verify the expectation characteristic for the Uniform(m) random variable, which can be expressed in HOL as follows.

Theorem 3.9: *Expectation of Uniform(m) Random Variable*

$$\vdash \forall\ \mathtt{m.}\quad \mathtt{expec\ (\lambda s.\ \ prob\_unif\ (m+1)\ s)} = \tfrac{\mathtt{m}}{2}$$

We proceed with the proof of this theorem in HOL by rewriting it with the definition of expectation

$$\lim_{\mathtt{k}\to\infty}(\sum_{\mathtt{n}=0}^{\mathtt{k}} \mathtt{n}\ \mathbb{P}\{\mathtt{s}\mid \mathtt{fst(prob\_unif\ (m+1)\ s)} = \mathtt{n}\}) = \frac{\mathtt{m}}{2} \tag{3.46}$$

Next, we verified in HOL that the Uniform(m) random variable can never acquire a value greater than or equal to m using its PMF property.

$$\vdash \forall\ \mathtt{m\ x.\ (m+1)} \leq \mathtt{x} \Rightarrow \mathbb{P}\ \{\mathtt{s}\mid \mathtt{fst(prob\_unif\ (m+1)\ s)} = \mathtt{x}\} = \mathtt{0} \tag{3.47}$$

This property allows us to rewrite the infinite summation of Equation (3.46) in terms of a finite summation over $(m+1)$ values using the properties verified in the HOL theory of limit of a *real* sequence.

$$\sum_{n=0}^{m+1} n\ \mathbb{P}\{s \mid \texttt{fst(prob\_unif}\ (m+1)\ s) = n\} = \frac{m}{2} \tag{3.48}$$

The above equation can be verified using the PMF of the Uniform(m) random variable along with some basic properties of the summation function in HOL.

Next, we formally verify the variance characteristic for the Uniform(m) random variable, which can be expressed in HOL as follows.

Theorem 3.10: *Variance of Uniform(m) Random Variable*

$$\vdash \forall\ m.\quad \texttt{variance}\ (\lambda s.\quad \texttt{prob\_unif}\ (m+1)\ s) = \tfrac{(m+1)^2-1}{12}$$

The proof goal of Theorem 3.10 can be simplified using the variance relation given in Theorem 3.5 and the definition of expectation of a function of a random variable (Definition 3.1) as follows

$$\sum_{n=0}^{\infty} n^2 \mathbb{P}\{s|\texttt{fst(prob\_unif}\ (m+1)\ s) = n\} - (\texttt{expec}\ (\lambda s.\texttt{prob\_unif}\ (m+1)\ s))^2 = \frac{(m+1)^2-1}{12} \tag{3.49}$$

Now, the second moment of the Uniform(m) random variable, i.e., the first term on the LHS of the above equation, can be verified in HOL to be equal to $\frac{m(2m+1)}{2}$, using the same approach as was used for the verification of its expectation relation in Theorem 3.9. This result, along with Theorem 3.9, and some arithmetic reasoning, allows us to verify Equation (3.49) and thus Theorem 3.10 in HOL.

3.5.2 Bernoulli(p) Random Variable

The Bernoulli(p) random variable models an experiment with two outcomes; success and failure, whereas the parameter p represents the probability of success. A sampling

algorithm of the Bernoulli(p) random variable has been formalized in [36] as the function `prob_bern` such that it returns $True$ with probability p and $False$ otherwise. It has also been verified to be correct by proving the corresponding PMF property in HOL.

$$\vdash \forall\ \mathtt{p}.\ \ 0 \leq \mathtt{p} \wedge \mathtt{p} \leq 1 \Rightarrow \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}\ (\mathtt{prob\_bern}\ \mathtt{p}\ \mathtt{s})\} = \mathtt{p}$$

The Bernoulli(p) random variable ranges over 2 values of *Boolean* data type. The expectation property of these kind of discrete random variables, which range over a finite number of values of a different data type than positive integers, can be verified using our approach by mapping all their values to distinct positive integers. In the case of Bernoulli(p) random variable, we redefined the function `prob_bern` such that it returns positive integers 1 and 0 instead of the Boolean quantities $True$ and $False$ with the same probability, respectively, i.e., the range of the random variable was changed from *Boolean* data type to positive integers. It is important to note that this redefinition does not change the distribution properties of the given random variable. The expectation property for this alternate definition of Bernoulli(p) random variable, `prob_bernN`, can be expressed in HOL as follows

Theorem 3.11: . *Expectation of Bernoulli(p) Random Variable*

$$\vdash \forall\ \mathtt{p}.\ \ 0 \leq \mathtt{p} \wedge \mathtt{p} \leq 1 \Rightarrow \mathtt{expec}\ (\lambda \mathtt{s}.\ \ \mathtt{prob\_bernN}\ \mathtt{p}\ \mathtt{s}) = \mathtt{p}$$

Theorem 3.11 can now be verified using the same procedure used for the case of random variables that range over a finite number of positive integers, such as the Uniform(m) random variable. In the case of Bernoulli(p) random variable, we were able to replace the infinite summation in the definition of expectation with the summation of the first two values of the corresponding *real* sequence using the HOL theory of limit of a *real* sequence. This substitution along with the PMF property of

the Bernoulli(p) random variable and some arithmetic reasoning allowed us to verify Theorem 3.11 in HOL.

We also verified the variance of the Bernoulli(p) random variable in HOL, using a similar approach that we used for the verification of the variance relation for the Unform(m) random variable and the HOL theorem is given below

Theorem 3.12: *Variance of Bernoulli(p) Random Variable*

```
⊢ ∀ p.  0 ≤ p ∧ p ≤ 1
    ⇒ variance (λs.  prob_bernN p s) = p (1-p)
```

3.5.3 Geometric(p) Random Variable

The Geometric(p) random variable can be defined as the index of the first success in an infinite sequence of Bernoulli(p) trials [19]. Therefore, the Geometric(p) distribution may be sampled by extracting random bits from the function `prob_bern`, explained in the previous section, and stopping as soon as the first *False* is encountered and returning the number of trials performed till this point. The Geometric(p) random variable ranges over a countably infinite number of positive integers and is thus different from the random variables that we have considered so far.

Based on the above sampling algorithm, we modeled the Geometric(p) random variable using the *probabilistic while loop* [36] in HOL as follows

Definition 3.6: *A Sampling Algorithm for Geometric(p) Distribution*

```
⊢ ∀ p s.  prob_geom_iter p s =
  bind (prob_bern (1-p)) (λb.  unit (b, suc (snd s)))
⊢ ∀ p.  prob_geom_loop p = prob_while fst (prob_geom_iter p)
⊢ ∀ p.  prob_geom p = bind (bind (unit (T, 1))
  (prob_geom_loop p)) (λs.  unit (snd s - 1))
```

It is important to note that p, which represents the probability of success for the Geometric(p) or the probability of obtaining $False$ from the Bernoulli(p) random variable, cannot be assigned a value equal to 0 as this will lead to a non-terminating while loop.

We verify the PMF property of the Geometric(p) random variable using the fact that the function `prob_geom` preserves *strong function independence* along with some theorems from probability and set theories in HOL.

Theorem 3.13: *PMF of Geometric(p) Random Variable*

$$\vdash \forall\ \mathtt{n\ p.}\quad \mathtt{0 < p} \wedge \mathtt{p} \leq \mathtt{1} \Rightarrow$$
$$\mathbb{P}\ \{\mathtt{s\ |\ fst\ (prob\_geom\ p\ s) = (n + 1)}\} = \mathtt{p\ (1 - p)^n}$$

The expectation theorem for the Geometric(p) random variable can now be expressed in HOL as follows

Theorem 3.14: *Expectation of Geometric(p) Random Variable*

$$\vdash \forall\ \mathtt{p.}\quad \mathtt{0 < p} \wedge \mathtt{p} \leq \mathtt{1} \Rightarrow \mathtt{expec\ (\lambda s.\ \ prob\_geom\ p\ s)} = \frac{1}{p}$$

Rewriting the above proof goal with the definition of expectation and simplifying using the PMF relation for the Geometric(p) random variable along with some arithmetic reasoning, we reach the following subgoal.

$$\lim_{k\to\infty}(\sum_{n=0}^{k}((n+1)p(1-p)^n)) = \frac{1}{p} \tag{3.50}$$

Substituting $1-q$ for p and after some rearrangement of the terms, based on arithmetic reasoning, the above subgoal can be rewritten as follows.

$$\lim_{k\to\infty}(\sum_{n=0}^{k}((n+1)q^n)) = \frac{1}{(1-q)^2} \tag{3.51}$$

Now, using the properties of summation of a real sequence in HOL, we proved the following relationship

$$\forall q\ k.\ \sum_{n=0}^{k}((n+1)q^n) = \sum_{n=0}^{k}(\sum_{i=0}^{k} q^i - \sum_{i=0}^{n} q^i) \tag{3.52}$$

which allows us to rewrite the subgoal under consideration, given in Equation (3.51) as follows.

$$\lim_{k\to\infty}(\sum_{n=0}^{k}(\sum_{i=0}^{k} q^i - \sum_{i=0}^{n} q^i)) = \frac{1}{(1-q)^2} \tag{3.53}$$

The above subgoal can now be proved using the summation of a finite geometric series along with some properties of summation and limit of *real* sequences available in the *real* number theories in HOL. This also concludes the proof of Theorem 3.14 in HOL.

The variance property of Geometric(p) random variable can be stated in HOL as follows.

Theorem 3.15: *Variance of Geometric(p) Random Variable*

$\vdash \forall$ p. 0 < p $\wedge$ p $\leq$ 1

$\Rightarrow$ (variance (λs. prob_geom p s) = $\frac{1-p}{p^2}$)

We utilize the variance property, given in Theorem 3.5, to verify Theorem 3.15. The foremost step in this regard is to verify the second moment relationship for the Geometric(p) random variable.

$$\forall\ p.\ 0 < p\ \wedge\ p \leq 1\ \Rightarrow\ (\texttt{expec\_fn}(\lambda n.\ n^2(\lambda s.\ \texttt{prob\_geom}\ p\ s)) = \frac{2}{p^2} - \frac{1}{p}) \tag{3.54}$$

Rewriting the above proof goal with the definition of function `expec_fn` and simplifying using the PMF relation of the Geometric random variable along with some properties from HOL *real* number theories, we reach the following subgoal.

$$\lim_{k\to\infty}(\sum_{n=0}^{k}((n+1)^2p(1-p)^n)) = \frac{2}{p^2} - \frac{1}{p} \tag{3.55}$$

Now, substituting $1 - q$ for p and after some rearrangement of the terms, based on arithmetic reasoning, the above subgoal can be rewritten as follows.

$$\lim_{k\to\infty}(\sum_{n=0}^{k}((n+1)^2q^n)) = \frac{2}{(1-q)^3} - \frac{1}{(1-q)^2} \tag{3.56}$$

Using the properties of summation of a real sequence in HOL, we prove the following

$$\forall q\ k.\ \sum_{n=0}^{k}((n+1)^2q^n) = \sum_{n=0}^{k}((2n+1)(\sum_{i=0}^{k}q^i - \sum_{i=0}^{n}q^i)) \tag{3.57}$$

which allows us to rewrite the subgoal under consideration, given in Equation 3.56, as follows.

$$\lim_{k\to\infty}(\sum_{n=0}^{k}((2n+1)(\sum_{i=0}^{k}q^i - \sum_{i=0}^{n}q^i))) = \frac{2}{(1-q)^3} - \frac{1}{(1-q)^2} \tag{3.58}$$

The above subgoal can now be proved using the summation of a finite geometric series along with some properties of summation and limit of real sequences available in the *real* number theories in HOL. This concludes the proof of the second moment relation for the Geometric(p) random variable, which can now be used along with Theorems 3.5 and 3.14 and some arithmetic reasoning to prove Theorem 3.15 in HOL.

3.5.4 Binomial(m, p) Random Variable

The Binomial(m, p) random variable models an experiment that counts the number of successes in a finite number, m, of independent Bernoulli trials, with a success probability equal to p [19]. Therefore, the Binomial(m, p) distribution may be sampled by an algorithm in HOL that sums m independent outcomes of the `prob_bernN` random

variable, which models the Bernoulli(p) random variable with outcomes 0 and 1, as described in Section 3.5.2. We formalize it in HOL by first defining a function that recursively returns a list of m Bernoulli(p) random variables.

Definition 3.7: *List of m Bernoulli(p) Random Variables*

```
⊢ ∀ p.  bern_lst 0 p = [] ∧
 (∀ n p.  bern_lst (suc n) p =
   prob_bernN p ::  (bern_lst n p))
```

Now, the Binomial(m, p) random variable can be modeled as the sum of all elements in the list modeled by the HOL function `bern_lst`, such that the result of each one of these random variables is independent of one another. This can be done using the function `sum_rv_lst`, given in Definition 3.4, as follows

Definition 3.8: *Binomial(m,p) Random Variable*

```
⊢ ∀ m p.  prob_bino m p = sum_rv_lst (bern_lst m p)
```

We verify the correctness of this definition by proving its PMF in HOL

Theorem 3.16: *PMF of Binomial(m,p) Random Variable*

```
⊢ ∀ m p n.  0 ≤ p ∧ p ≤ 1
  ⇒ ℙ {s | fst (prob_bino m p s) = n}
    = (binomial m n) p^n (1 - p)^(m-n)
```

using the properties verified in the HOL libraries corresponding to the probability and set theories. The HOL function (`binomial m n`), in the above theorem, represents the term $\frac{m!}{n!(m-n)!}$.

The expectation theorem for the Binomial(m, p) random variable can now be expressed in HOL, as follows.

Theorem 3.17: *Expectation of Binomial(m,p) Random Variable*

$\vdash \forall$ m p. 0 < p $\wedge$ p $\leq$ 1

$\Rightarrow$ expec (λs. prob_bino m p s) = m p

Instead of using the definition of expectation directly, we use the linearity of expectation property, given in Theorem 3.2, to prove the above theorem. This way, we do not need to deal with the summation involving the `binomial` function in HOL, which saves a considerable amount of proof effort. Since, the Binomial(m, p) random variable represents the sum of m Bernoulli(p) random variables, the linearity of expectation property allows us to rewrite the LHS of the proof goal in Theorem 3.17 as the sum of m expectation values of the Bernoulli(p) random variable. Now, using the fact that the expectation of the Bernoulli(p) random variable is equal to p, as given in Theorem 3.11, Theorem 3.17 can be verified in HOL.

In a similar way, we can also verify the variance relation for the Binomial(m, p) in HOL using the linearity of variance property, given in Theorem 3.6.

Theorem 3.18: *Variance of Binomial(m,p) Random Variable*

$\vdash \forall$ m p. 0 < p $\wedge$ p $\leq$ 1

$\Rightarrow$ variance (λs. prob_bino m p s) = m p (1 - p)

The formalization and verification of the Binomial(m, p), presented in this section, illustrates one of the main strengths of mechanical theorem proving, i.e., the reusability of existing definitions and theorems to develop and prove new and more complex definitions and theorems. This approach greatly speeds up the formal verification process and allows us to take the work further than would have been possible starting from scratch, without compromising on the soundness of the results.

3.6 Probabilistic Analysis of Coupon Collector's Problem

In this section, we utilize the HOL formalization presented so far in this chapter to conduct the probabilistic analysis of the Coupon Collector's problem [61] as a case study. Firstly, we present a brief overview of the algorithm and present its formalization in HOL. This is followed by the details about the verification steps.

3.6.1 Algorithm Description

The Coupon Collector's problem 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.

This simple problem arises in many different scenarios. For example, suppose that packets are sent in a stream from source to destination host along a fixed path of routers and the destination host needs to know all routers that the stream of data has passed through. This may be done by appending the identification of each router to the packet header but this is not a practical solution as usually we do not have this much room available. An alternate way of meeting this requirement is to store the identification of only one router, uniformly selected at random between all routers on the path, in each packet header. Then, from the point of view of the destination host, determining all routers on the path is like a Coupon Collector's problem.

Our first goal is to verify, using HOL, 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, we then reason about the tail distribution properties of the Coupon Collector's problem using the formally verified Markov's and Chebyshev's inequalities.

3.6.2 Formal Specification in HOL

The Coupon Collector's problem can be formalized by modeling the total number of trials required to obtain all n unique coupons, say X, as a sum of the number of trials required to obtain each distinct coupon, i.e., $X = \sum_{i=1}^{n} X_i$, where X_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 X into the sum of n random variables $X_1, X_2 \cdots, X_n$ is that each X_i can be modeled as a Geometric(p) random variable. Based on the above model, the expectation and variance relations for the Coupon Collector's problem can be verified using the linearity of expectation and variance properties, given in Theorems 3.2 and 3.6, and the expectation and variance of the Geometric(p) random variable, given in Theorems 3.14 and 3.15, respectively.

The Coupon Collector's problem is modeled in HOL by identifying the coupons with unique positive integers, such that the first coupon acquired by the coupon collector is identified as number 0 and after that each different kind of a coupon acquired with subsequent numbers in numerological order. The coupon collector saves these coupons in a list of positive integers. The following function accepts the number of distinct coupons acquired by the coupon collector and recursively generates the corresponding coupon collector's list.

Definition 3.9: *Coupon Collector's List*

```
⊢ (coupon_lst 0 = []) ∧
  ∀ n.  (coupon_lst (n + 1) = n ::  (coupon_lst n))
```

The next step is to define a list of Geometric random variables, such that each

one of its elements represents an X_i, mentioned above. The probability of success for each one of these Geometric random variables is different from one another and depends on the number of different coupons acquired so far. Since, every coupon is drawn independently and uniformly at random from the n possibilities and the coupons are identified with positive integers, we can use the Uniform(n) random variable to model each trial of acquiring a coupon. Now we can define the probability of success for a particular Geometric random variable as the probability of the event when the Uniform(n) random variable generates a new value, i.e., a value that is not already present in the coupon collector's list. Using this probability of success, the following function generates the required list of Geometric random variables

Definition 3.10: *Geometric Variable List for Coupon Collector's Problem*

```
⊢ ∀ n.  (geom_rv_lst [] n = [prob_geom 1]) ∧
  ∀ h t n.  (geom_rv_lst (h::t) n =
    (prob_geom ℙ{s | ¬(mem (fst(prob_unif n s)) (h::t))}) ::
      (geom_rv_lst t n))
```

where the functions `prob_geom` and `prob_unif` model the Geometric(p) and Uniform (n) random variables, respectively, and are given in the last section. The `geom_rv_lst`, accepts two arguments; a list of positive integers that represents the coupon collector's list and a *positive integer* number that represents the total number of coupons in the Coupon Collector's problem. It returns, a list of Geometric random variables that can be added up to model the coupon collecting process of the already acquired coupons in the given list. The base case in the above recursive definition corresponds to the condition when the coupon collector does not have any coupon and thus the probability of success, i.e., the probability of acquiring a new coupon, is 1.

Using the above definitions along with the function `sum_rv_lst`, given in Definition 3.4, the Coupon Collector's problem can be formally modeled in HOL as follows.

Definition 3.11: *Probabilistic Algorithm for Coupon Collector's Problem*

```
⊢ ∀ n.  (coupon_collector (n + 1) =
   (sum_rv_lst (geom_rv_lst (coupon_lst n) (n + 1)))
```

The function, `coupon_collector`, accepts a positive integer greater than 0, $n+1$, which represents the total number of different coupons that are required to be collected. It returns the number of trials for acquiring these $n+1$ distinct coupons.

3.6.3 Probabilistic Analysis in HOL

The first step towards the verification of statistical properties for the Coupon Collector's problem is to verify the relation for the probability of acquiring a new coupon.

Lemma 3.6: *Probability of Acquiring a New Coupon*

```
⊢ ∀ L n.  (dist_lst L) ∧ (∀a.  mem a L ⇒ (a < (n + 1)))
  ⇒ (ℙ {s | ¬(mem (fst(prob_unif (n + 1) s)) L)}
    = 1 − (length L)/(n+1))
```

where the predicate `dist_lst` returns *True* if all elements in its argument list are distinct. Thus, the assumption in the above theorem ensures that all elements in the given list of positive integers are distinct and are less than $(n+1)$. The coupon collector's list, modeled by the function `coupon_lst`, satisfies both assumptions in Lemma 3.6 for any given argument. Therefore, the probability of success for the Geometric random variable, which models the acquiring process of a new coupon when the coupon collectors list is exactly equal to L, is $1 - \frac{length\ L}{(n+1)}$. The expectation of such a Geometric random variable can be easily verified to be equal to $\frac{n+1}{(n+1)-(length\ L)}$, by

Theorem 3.14. This result along with the linearity of expectation property, given in Theorem 3.2, can now be used to verify the expectation of the number of trials to collect all distinct coupons.

Theorem 3.19: *Expectation of Coupon Collector's Problem*

$\vdash \forall$ n. expec (coupon_collector (n + 1)) = (n + 1) $(\sum_{i=0}^{n+1} \frac{1}{i+1})$

Next, we build upon the above results to formally reason about the tail distribution properties of the number of trials required to acquire all coupons in HOL. For this purpose, we utilize the formally verified Markov's and Chebyshev's inequalities, which have been verified in Theorems 3.7 and 3.8, respectively. The first step in this regard is to have access to formal proofs for the expectation and variance relations for the events of interest. The expectation has already been verified (Theorem 3.19) and thus we proceed by verifying a relationship for the variance first.

Instead of verifying the exact value of the variance for the number of trials required to acquire all coupons, we verify an upper bound for this variance

Theorem 3.20: *Variance Upper Bound of Coupon Collector's Problem*

$\vdash \forall$ n. variance (coupon_collector (n + 1))

$\leq$ ((n + 1)^2) $(\sum_{i=0}^{n+1} (\frac{1}{(i+1)^2}))$

The formal proof for the above theorem is based on the definition of the function coupon_collector, the linearity of variance property, given in Theorem 3.6, the result of Lemma 3.6, and the variance of Geometric random variable, verified in Theorem 3.15, along with some arithmetic reasoning.

Now, using the above mentioned results, we can formally verify the following two tail distribution bounds for the Coupon Collector's problem based on the formally verified Markov's and Chebyshev's inequalities, respectively.

Theorem 3.21: *Weak Tail Distribution Bound for the Coupon Collector's Problem*

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

Theorem 3.22: *Stronger Tail Distribution Bound for the Coupon Collector's Problem*

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

The script corresponding to the formalization and verification of the Coupon Collector's problem translated to approximately 1000 lines of HOL code and we had to spent about 100 man-hours on this project.

Thus, we have been able to formally verify the tail distribution bounds for the number of trials required to acquire all distinct coupons in the Coupon Collector's problem. Our results exactly match the results of the analysis based on paper-and-pencil proof techniques [61] 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. It is also worth mentioning at this point that it is due to the formally verified linearity of expectation and variance properties and the Markov's and Chebyshev's inequalities that the complex task of reasoning about tail distribution bounds of the Coupon Collector's problem, which involves multiple random variables, was simply proved in HOL using summation over the expectation or variance of a single Geometric(p) random variable.

3.7 Summary and Discussions

The formalization and verification, presented in this chapter, can be utilized to formally reason about expectation, variance and tail distribution properties for positive integer valued discrete random variables. Statistical properties play a vital role in probabilistic analysis and thus the ability of their verification in a theorem-proving environment can be regarded as a significant step towards a successful theorem-proving based probabilistic analysis framework.

We started off by presenting a formal definition of expectation for a function of a discrete random variable that can attain values in positive integers only. Building upon this definition, we formalized the mathematical concept of variance and were able to verify some classical properties of expectation and variance in HOL. The chapter also includes the verification of Markov's and Chebyshev's inequalities and the expectation and variance relations for some commonly used discrete random variables. To the best of our knowledge, this is the first time that an automated reasoning approach regarding the statistical properties of discrete random variables has been presented in the open literature.

An alternative approach that can be used to formalize the expectation of a random variable in higher-order logic is based on the mathematical concept of probability space. Since every random variable can be expressed as a real-valued function defined on the sample space, S, we can formalize expectation in terms of the probability space $(S, \mathfrak{F}, P)$, where $\mathfrak{F}$ is the sigma field of subsets of S, and P is the probability measure. The main benefit of this approach is that it leads to the formalization of the general definition of expectation, given in Equation (3.1), for discrete random variables. On the other hand, in this approach we require the formal definition of a summation function for functions with domain in the sample space S. Such definition does not exist in

the available HOL theories and thus needs to be formalized from scratch. It would be an interesting future work to formalize this summation and define a higher-order-logic definition of expectation based on the concept of probability space. A formal link may then be established between this generalized definition and the formal definition of expectation for discrete random variables with positive integers as their co-domain, presented in this thesis. Such a relationship would further strengthen the soundness of the definitions presented in this thesis.

Computer science is one of the key application areas of probabilistic analysis. Therefore, we target this domain in the case study for the formalization and verification given in this chapter, as we present the probabilistic analysis of the Coupon Collector's problem. We first formalize the Coupon Collector's problem using the summation of a list of Geometric(p) random variables. Using the formal definitions of expectation and variance, presented in this chapter, we also develop higher-order-logic theorems for the expectation, variance and tail distribution properties for the number of trials to acquire all distinct coupons. These theorems are then verified in HOL using the formally verified properties of expectation or variance, given in this chapter, and the proof details have been provided. This example illustrates the flow of a complete theorem proving based formal probabilistic analysis process.

The statistical property verification infrastructure presented in this thesis can also be used to verify the expectation properties of a number of other positive integer valued random variables, e.g., Binomial, Logarithmic and Poisson [41] and commercial computation problems, such as the Quicksort [61], the Chinese Appetizer and the Hat-Check problems [28].

Chapter 4

Continuous Random Variables

This chapter presents a framework that can be used to formalize any continuous random variable for which the inverse of the CDF can be expressed in a closed mathematical form. The framework also allows us to formally verify probability distribution properties of these random variables. These capabilities can be used to formally specify and verify theorems for probabilistic properties regarding continuous random components of systems in HOL. To illustrate the practical effectiveness of the proposed framework, we present the formalization of Exponential(λ), Uniform(a, b), Rayleigh(λ) and Triangular($0, a$) random variables. We also present a simple case study of probabilistic analysis regarding roundoff error in a digital processor, which utilizes the continuous Uniform(a, b) random variable.

4.1 Introduction

Hurd's methodology [36] for the verification of probabilistic algorithms has been successfully used for the formalization and verification of some discrete random variables in HOL. The algorithms for these discrete random variables are either guaranteed to

terminate or satisfy probabilistic termination, meaning that the probability that the algorithm terminates is 1. Thus, they can be expressed by either well formed recursive functions or the *probabilistic while loop* [36]. On the other hand, the modeling of continuous random variables requires non-terminating programs and hence calls for a different approach.

In this chapter, we propose a methodology for the formalization of continuous random variables in HOL. Our methodology utilizes the verification framework, presented in [36], and is based on the concept of the nonuniform random number generation [20], which is the process of obtaining random variates of arbitrary distributions using a Standard Uniform random number generator. The main advantage of this approach is that we only need to formalize one continuous random variable from scratch, i.e., the Standard Uniform random variable, which can be used to model other continuous random variables by formalizing the corresponding nonuniform random number generation method.

Based on the above methodology, we now present a framework, illustrated in Figure 4.1, for the formalization of continuous probability distributions for which the inverse of the CDF can be represented in a closed mathematical form. Firstly, we formally specify the Standard Uniform random variable and verify its correctness by proving the corresponding CDF and measurability properties. The next step is the formalization of the CDF and the verification of its classical properties. Then we formally specify the mathematical concept of the inverse function of a CDF. This formal specification, 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) [20], which is a well known nonuniform random

generation technique for generating nonuniform random variates for continuous probability distributions for which the inverse of the CDF can be represented in a closed mathematical form. At this point, the formalized Standard Uniform random variable can be used to formally specify any such continuous random variable. Whereas, the CDF of the formally specified continuous random variables can be verified, based on simple arithmetic reasoning, using the formal proof of the ITM.

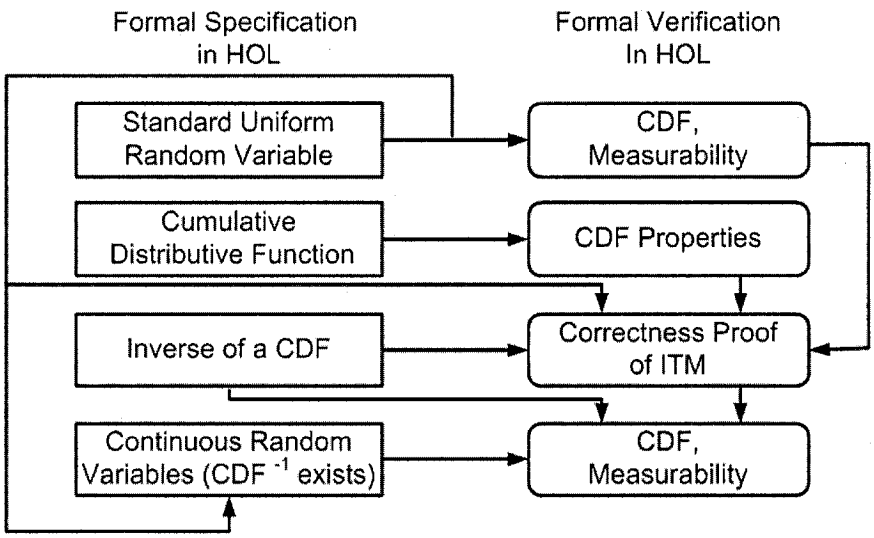

Figure 4.1: Formalization Framework for Continuous Random Variables

The next three sections of this chapter present the HOL formalization of the three major steps, given in Figure 4.1, i.e., the Standard Uniform random variable, the CDF and the ITM. Then, for illustration purposes, we utilize this formalization to formalize and verify the Exponential(λ), Uniform(a, b), Rayleigh(λ) and Triangular($0, a$) random variables based on the framework of Figure 4.1.

4.2 Standard Uniform Random Variable

In this section, we present the formalization and verification of the Standard Uniform distribution that is the first step in the proposed framework for the formalization of continuous probability distributions as shown in Figure 4.1. Standard Uniform random variable is a continuous random variable for which the probability that it will belong to a subinterval of [0,1] is proportional to the length of that subinterval. It can be characterized by the CDF as follows:

$$Pr(X \leq x) = \begin{cases} 0 & \text{if } x < 0; \\ x & \text{if } 0 \leq x < 1; \\ 1 & \text{if } 1 \leq x. \end{cases} \tag{4.1}$$

4.2.1 Formal Specification

Standard Uniform random variable can be formalized in a number of different ways using the various formal semantics of probabilistic programs available in the literature of theoretical computer science. In order to minimize the effort and speed up the formalization process, our intent is to find a solution that enables us to build upon the existing work of Hurd [36].

It is a well known mathematical fact, see [23] for example, that a Standard Uniform random variate can be modeled by an infinite sequence of random bits (informally coin flips) as follows

$$\sum_{k=0}^{\infty} (\frac{1}{2})^{k+1} X_k \tag{4.2}$$

where X_k denotes the outcome of the k^{th} random bit; $True$ or $False$ represented as 1 or 0 respectively. The mathematical relation of Equation (4.2) presents a sampling

algorithm for the Standard Uniform random variable which is quite consistent with Hurd's formalization methodology, i.e, it allows us to model the Standard Uniform random variable by a deterministic function with access to the infinite Boolean sequence. The specification of this sampling algorithm in higher-order logic is not very straight forward though. Due to the infinite sampling, it cannot be modeled by either of the approaches proposed in [36], i.e., a recursive function or the *probabilistic while loop*. We approach this problem by splitting the corresponding mathematical relation into two steps. The first step is to mathematically represent a discrete version of the Standard Uniform random variable

$$(\lambda n. \sum_{k=0}^{n-1} (\frac{1}{2})^{k+1} X_k) \tag{4.3}$$

This lambda abstraction function accepts a positive integer n and generates an n-bit Standard Uniform random variable using the computation principle of Equation (4.2). The continuous Standard Uniform random variable can now be represented as a special case of Equation (4.3) when n tends to infinity

$$\lim_{n \to \infty} (\lambda n. \sum_{k=0}^{n-1} (\frac{1}{2})^{k+1} X_k) \tag{4.4}$$

The advantage of expressing the sampling algorithm of Equation (4.2) in these two steps is that now it can be specified in HOL. The mathematical relationship of Equation (4.3) can be specified in HOL by a recursive function using Hurd's methodology as it consumes a finite number of random bits, i.e., n. Then, the formalization of the mathematical concept of limit of a *real* sequence [32] in HOL can be used to specify the mathematical relation of Equation (4.4).

Next, we present the HOL formalization of the above steps. We first formalize a discrete Standard Uniform random variable that produces any one of the equally

spaced 2^n dyadic rationals in the interval $[0, 1-(\frac{1}{2})^n]$ with the same probability $(\frac{1}{2})^n$. It can be formally specified in HOL as a recursive function as follows.

Definition 4.1: *Discrete Standard Uniform Random Variable*

```
⊢ (std_unif_disc 0 = unit 0) ∧
  ∀ n.  (std_unif_disc (suc n) = bind (std_unif_disc n)
    (λm.  bind
      sdest (λb.  unit (if b then ((1/2)^(n+1) + m) else m))))
```

The function, `std_unif_disc`, models an n-bit discrete Standard Uniform random variable based on the principle of Equation (4.2) by simply converting the first n random bits $B_0, B_1, B_2, \ldots B_{n-1}$ of the infinite Boolean sequence to their equivalent real number with the binary representation $0.B_0B_1B_2 \ldots B_{n-1}$. It returns a pair such that the first component is the n-bit discrete Standard Uniform random variable and the second component is the unused portion of the infinite Boolean sequence. The following properties for the discrete Standard Uniform random variable may be proved by induction on its argument n.

Lemma 4.1: *Range of Discrete Standard Uniform Random Variable*

```
⊢ ∀ n s.  0 ≤ fst (std_unif_disc n s) ≤ 1 - (1/2)^n
```

Lemma 4.2: *CDF of Discrete Standard Uniform Random Variable*

```
⊢ ∀ m n x.  ℙ {s | fst (std_unif_disc n s) ≤ x} =
  if (x < 0) then 0 else
    if (x ≥ 1) then 1 else
      if (x = m/2^n) then sucm/2^n
        else 0
```

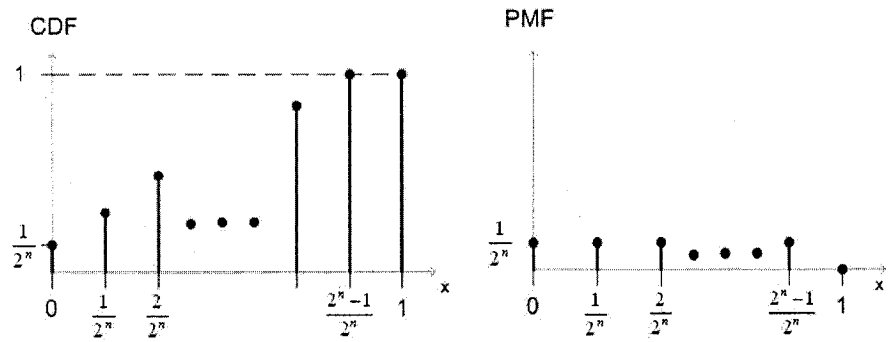

Figure 4.2: Distribution Characteristics for the Function `std_unif_disc`

Lemma 4.3: *PMF of Discrete Standard Uniform Random Variable*

$\vdash \forall$ m n x. $\mathbb{P}$ {s | fst (std_unif_disc n s) = x} =

if (x < 0) then 0 else

if (x $\geq$ 1) then 0 else

if (x = $\frac{m}{2^n}$) then $\frac{1}{2^n}$

else 0

The term $\frac{m}{2^n}$ in Lemmas 4.2 and 4.3 represents all the dyadic rationals in the interval $[0, 1-(\frac{1}{2})^n]$ since the variable m belongs to the HOL datatype *num* for positive integers $\{0, 1, 2, \cdots\}$. Collectively Lemmas 4.1, 4.2 and 4.3, illustrated in Figure 4.2, formally prove that the first component of the function `std_unif_disc` is a discrete uniform random variable.

The function `std_unif_disc` can now be used to model the *real* sequence of Equation (4.3). We proved in HOL that this sequence is convergent, i.e., it approaches a unique value when n tends to infinity.

Lemma 4.4: *Discrete Standard Uniform Random Variable Convergence*

$\vdash \forall$ s. convergent (λn. fst (std_unif_disc n s))

where `convergent` represents the HOL predicate that returns $True$ if its argument is a convergent *real* sequence [32]. Based on Lemma 4.4, we are able to formally specify the Standard Uniform random variable in HOL according to Equation (4.4).

Definition 4.2: *Standard Uniform Random Variable*

```
⊢ ∀ s.  std_unif_cont s = lim (λn.  fst (std_unif_disc n s))
```

where `lim M` represents the HOL formalization of the limit of a *real* sequence M (i.e., $lim\ M = \lim_{n\to\infty} M(n)$) [32]. The following properties may be proved using the *real* analysis theorems [32] and the function definition for `std_unif_disc`.

Lemma 4.5: *Range of Standard Uniform Random Variable*

```
⊢ ∀ s.  0 ≤ std_unif_cont s ≤ 1
```

Lemma 4.6: *Relationship Between Discrete and Continuous Standard Uniform Random Variables*

```
⊢ ∀ s n.  fst (std_unif_disc n s) ≤ std_unif_cont s
  ≤ fst(std_unif_disc n s)+(1/2)^n
```

Lemma 4.5 formally shows that the value for the function `std_unif_cont` always lies in the *real* interval [0,1]. The minimum and maximum values of 0 and 1 correspond to the cases when all elements of the infinite Boolean sequence s are $False$ or $True$, respectively. Lemma 4.6 highlights the relationship between the values of the first component of the function `std_unif_disc` and the function `std_unif_cont`, i.e., if the value for the former is a, then the value of the later lies in the interval $[a, a + (\frac{1}{2})^n]$.

$$\texttt{(fst (std\_unif\_disc n s) = a)} \Rightarrow \texttt{(a} \leq \texttt{(std\_unif\_cont s)} \leq \texttt{a} + (\frac{1}{2})^{\texttt{n}}) \qquad (4.5)$$

4.2.2 Formal Verification

The formalized Standard Uniform random variable, `std_unif_cont`, can be formally verified in HOL by proving its CDF to be equal to the theoretical value given in Equation (4.1) and its PMF to be equal to 0, which is an intrinsic characteristic of all continuous random variables.

We begin with the CDF verification and the corresponding theorem can be expressed in HOL as follows

Theorem 4.1: *CDF of the Standard Uniform Random Variable*

```
⊢ ∀ x.
    ℙ {s | std_unif_cont s ≤ x} =
      if (x < 0) then 0 else (if (x < 1) then x else 1)
```

The proof for the cases (x<0) and (1≤x) is a straightforward implication of Lemma 4.5, which states that for all infinite Boolean sequences the value of the function `std_unif_cont` lies in the interval [0,1]. The probability that the function `std_unif_cont` acquires a value less than 0 is 0 as there is no infinite Boolean sequence that satisfies this condition. Similarly, the probability that the function `std_unif_cont` acquires a value less than or equal to 1 is 1 since all infinite Boolean sequences fulfill this condition.

$$\begin{aligned} &\forall x.\ (x < 0) \Rightarrow\ \mathbb{P}\ \{s \mid \texttt{fst}\ (\texttt{std\_unif\_cont}\ s) \leq x\} = 0 \\ &\forall x.\ (1 \leq x) \Rightarrow\ \mathbb{P}\ \{s \mid \texttt{fst}\ (\texttt{std\_unif\_cont}\ s) \leq x\} = 1 \end{aligned} \tag{4.6}$$

Evaluating the probability of Theorem 4.1 for the interval [0,1) is a surprisingly difficult problem in the HOL theorem prover. However, given that we have evaluated the CDF for the first component of the function `std_unif_disc`, which represents

a discrete Standard Uniform random variable, a reasonable approach is to find a discrete approximation to the CDF of the function `std_unif_cont`, which represents the Standard Uniform random variable. The key to this approach is to be able to express the CDF of the function `std_unif_cont` in terms of the CDF of the first component of the function `std_unif_disc`. In order to do this, we need to identify the closest values of the first component of `(std_unif_disc n s)` corresponding to any given value of `(std_unif_cont s)`. We known from Lemmas 4.1, 4.2 and 4.3 that the first component of `(std_unif_disc n s)` is always a dyadic rational with denominator 2^n, in the interval $[0, 1-(\frac{1}{2})^n]$. On the other hand, the function `std_unif_cont` can attain any *real* value in the interval [0,1]. Therefore, the two values of the first component of `(std_unif_disc n s)` that are closest to any given value, say y, of the function `std_unif_cont` are the two consecutive dyadic rationals (with denominators 2^n) such that the smaller dyadic rational is less than y and the greater dyadic rational is greater than or equal to y. The mathematical concept of *ceiling*, that represents the smallest integer number greater than or equal to a *real* number, can be used in identifying these dyadic rationals. We verified in HOL that the above mentioned dyadic rationals are $\frac{\lceil 2^n y \rceil - 1}{2^n}$ and $\frac{\lceil 2^n y \rceil}{2^n}$

Lemma 4.7: *Closest Dyadic Rationals (Denominator 2^n) to a Real Number*

$$\vdash \forall\ \mathtt{n}\ \mathtt{y}.\quad (\mathtt{0} \leq \mathtt{y}) \Rightarrow \frac{\lceil 2^{\mathtt{n}}\mathtt{y} \rceil - \mathtt{1}}{2^{\mathtt{n}}} < \mathtt{y} \leq \frac{\lceil 2^{\mathtt{n}}\mathtt{y} \rceil}{2^{\mathtt{n}}}$$

for any positive *real* number y. Here $\lceil z \rceil$ denotes the HOL definition for the ceiling function that returns the smallest *num* value greater than or equal to its *real* argument z.

Now we will show how to express the CDF of the function `std_unif_cont` in terms of the CDF of the first component of the function `std_unif_disc` using the above dyadic rationals. It is important to note that the set {`s |fst(std_unif_disc n`

$\mathtt{s}) \leq \frac{\mathtt{m}}{2^{\mathtt{n}}}\}$ contains all the infinite Boolean sequences for which the value of the first n bits based on the algorithm implemented by the function `std_unif_disc` is less than or equal to the dyadic rational $\frac{\mathtt{m}}{2^{\mathtt{n}}}$. Using Equation (4.5), we can say that the value produced by the algorithm implemented by the function `std_unif_cont`, for any infinite Boolean sequence that is present in the set $\{\mathtt{s} \mid \mathtt{std\_unif\_disc\ n\ s} \leq \frac{\mathtt{m}}{2^{\mathtt{n}}}\}$, must be less than or equal to $\frac{\mathtt{m}+1}{2^{\mathtt{n}}}$. We used this useful reasoning along with Lemma 4.7 to prove the following result in HOL.

$$\begin{aligned} \forall \mathtt{x\ n}.(0 \leq \mathtt{x}) \Rightarrow \{\mathtt{s} \mid \mathtt{fst(std\_unif\_disc\ n\ s)} \leq \frac{\lceil 2^{\mathtt{n}}\mathtt{x} \rceil - 2}{2^{\mathtt{n}}}\} \subseteq \\ \{\mathtt{s} \mid \mathtt{std\_unif\_cont\ s} \leq \mathtt{x}\} \subseteq \\ \{\mathtt{s} \mid \mathtt{fst(std\_unif\_disc\ n\ s)} \leq \frac{\lceil 2^{\mathtt{n}}\mathtt{x} \rceil}{2^{\mathtt{n}}}\} \end{aligned} \tag{4.7}$$

The first set $\{\mathtt{s} \mid \mathtt{fst(std\_unif\_disc\ n\ s)} \leq \frac{\lceil 2^{\mathtt{n}}\mathtt{x} \rceil - 2}{2^{\mathtt{n}}}\}$, based on the above reasoning, contains all the infinite Boolean sequences for which the value produced by the algorithm implemented by the function `std_unif_cont` lies in the interval $[0, \frac{\lceil 2^{\mathtt{n}}\mathtt{x} \rceil - 1}{2^{\mathtt{n}}}]$. This set is a subset of the set $\{\mathtt{s} \mid \mathtt{std\_unif\_cont\ s} \leq \mathtt{x}\}$, which contains all the infinite Boolean sequences for which the value produced by the algorithm implemented by the function `std_unif_cont` lies in the interval $[0, x]$, as $\frac{\lceil 2^{\mathtt{n}}\mathtt{x} \rceil - 1}{2^{\mathtt{n}}}$ is always less than x according to Lemma 4.7. Similarly, the set $\{\mathtt{s} \mid \mathtt{std\_unif\_cont\ s} \leq \mathtt{x}\}$ is a subset of the set $\{\mathtt{s} \mid \mathtt{fst(std\_unif\_disc\ n\ s)} \leq \frac{\lceil 2^{\mathtt{n}}\mathtt{x} \rceil}{2^{\mathtt{n}}}\}$, which contains all the infinite Boolean sequences for which the value produced by the algorithm implemented by the function `std_unif_cont` lies in the interval $[0, \frac{\lceil 2^{\mathtt{n}}\mathtt{x} \rceil + 1}{2^{\mathtt{n}}}]$, as x is always less than or equal to $\frac{\lceil 2^{\mathtt{n}}\mathtt{x} \rceil + 1}{2^{\mathtt{n}}}$ according to Lemma 4.7.

Equation (4.7) and the monotonic property of the probability function $\mathbb{P}$, formalized in [36], which states that the probability of a measurable set is always less than or equal to the probability of its measurable superset $A \subseteq B \Rightarrow \mathbb{P}(A) \leq \mathbb{P}(B)$,

can be used to obtain the required relationship between the CDFs of `std_unif_cont` and the first component of the function `std_unif_disc`. But, in order to use the monotonic property in HOL, we must prove all the sets in Equation (4.7) to be measurable, i.e., they are in $\mathcal{E}$. It has been shown in [36], that if a function accesses the infinite Boolean sequence using only the `unit`, `bind` and `sdest` primitives then the function is guaranteed to preserve *strong function independence* and thus leads to measurable sets. The function `std_unif_disc` satisfies this condition and thus Hurd's formalization framework can be used to prove

$$\forall\ \mathtt{x\ n.\ measurable}\ \{\mathtt{s}\mid \mathtt{fst}\ (\mathtt{std\_unif\_disc\ n\ s})\ \leq \mathtt{x}\} \tag{4.8}$$

On the other hand, the definition of the function `std_unif_cont` involves the `lim` function and thus the corresponding sets cannot be proved to be measurable in a very straightforward manner. Therefore, in order to prove this, we leveraged the fact that each set in the sequence of sets $(\lambda \mathtt{n}.\{\mathtt{s}\mid \mathtt{fst(std\_unif\_disc\ n\ s)} \leq \mathtt{x}\})$ is a subset of the set before it, in other words, this sequence of sets is a monotonically decreasing sequence. Thus, the countable intersection of all sets in this sequence can be proved to be equal to the set $\{\mathtt{s}\mid \mathtt{std\_unif\_cont\ s} \leq \mathtt{x}\}$

$$\forall\ \mathtt{x}.\ \{\mathtt{s}\mid \mathtt{fst}\ (\mathtt{std\_unif\_cont\ s}) \leq \mathtt{x}\} = \bigcap_{\mathtt{n}} (\lambda \mathtt{n}.\{\mathtt{s}\mid \mathtt{fst}\ (\mathtt{std\_unif\_disc\ n\ s}) \leq \mathtt{x}\}) \tag{4.9}$$

Now the set $\{\mathtt{s}\mid \mathtt{std\_unif\_cont\ s} \leq \mathtt{x}\}$ can be proved to be measurable since measurable sets are closed under countable intersections [36] and all sets in the sequence $(\lambda \mathtt{n}.\{\mathtt{s}\mid \mathtt{fst(std\_unif\_disc\ n\ s)} \leq \mathtt{x}\})$ are measurable according to Equation (4.8).

$$\forall\, \texttt{x. measurable}\ \{\texttt{s} \mid (\texttt{std\_unif\_cont s}) \leq \texttt{x}\} \tag{4.10}$$

Equations (4.7), (4.8) and (4.10) can now be used along with the monotonic law of probability to obtain the desired relationship between the CDFs. The result can be further simplified by using the CDF relation for the first component of the function `std_unif_disc` given in Lemma 4.2.

$$\forall \texttt{x n.}\ (0 \leq \texttt{x}) \wedge (\texttt{x} < 1) \Rightarrow \frac{\lceil 2^{\texttt{n}}\texttt{x} \rceil - 1}{2^{\texttt{n}}} \leq \mathbb{P}\ \{\texttt{s} \mid \texttt{std\_unif\_cont s} \leq \texttt{x}\} \leq \frac{\lceil 2^{\texttt{n}}\texttt{x} \rceil + 1}{2^{\texttt{n}}} \tag{4.11}$$

As n approaches infinity both the fractions in Equation (4.11) approach x. This fact led us to prove the CDF relation of Theorem 4.1 for the interval [0,1) in HOL. Theorem 4.1 proves that the CDF of the function `std_unif_cont` is the same as the theoretical value of the CDF for a Standard Uniform random variable given in Equation (4.1) and thus is a formal argument to support the correctness of the fact that the function `std_unif_cont` models a Standard Uniform random variable.

Using similar reasoning as above, we also proved that the PMF of the function `std_unif_cont` is equal to 0.

Theorem 4.2: *PMF of the Standard Uniform Random Variable*

$$\vdash \forall\, \texttt{x.}\quad \mathbb{P}\{\texttt{s} \mid \texttt{std\_unif\_cont s = x}\} = 0$$

It follows from Theorem 4.2 that every outcome of the function `std_unif_cont` has a probability 0; which is a unique characteristic of all continuous random variables. Thus, Theorem 4.2 can be used to formally regard the function `std_unif_cont` as a continuous random variable.

4.3 Cumulative Distribution Function

In this section, we present the formal specification of the CDF and the verification of CDF properties in the HOL theorem prover. The definitions and theorems given in this section are applicable to both discrete and continuous random variables. CDF and its properties have been an integral part of the classical probability theory since its early development in the 1930s. The properties are mentioned in most of the probability theory texts, e.g,, [41], and have been used successfully in performing probabilistic analysis of random systems using paper-and-pencil proofs. Our main contribution is the formalization of these properties in a mechanical theorem prover. Besides being the second step in the proposed methodology for the formalization of continuous probability distributions, as shown in Figure 4.1, this formalization plays a vital role in reasoning about probabilistic properties of random variables within the framework of a sound theorem-prover environment.

4.3.1 Formal Specification of CDF

It follows from Equation (1.1) that the CDF can be formally specified in HOL by a higher-order-logic function that accepts a *real*-valued random variable and a *real* number argument and returns the probability of the event when the given random variable is less than or equal to the value of the given *real* number. Hurd's formalization of the probability function $\mathbb{P}$, which maps sets of infinite Boolean sequences to *real* numbers between 0 and 1, can be used to formally specify the CDF as follows:

Definition 4.3: *Cumulative Distribution Function*

$$\vdash \forall \texttt{R x.} \quad \texttt{cdf R x} = \mathbb{P}\ \{\texttt{s} \mid \texttt{R s} \leq \texttt{x}\}$$

where R represents the random variable that accepts an infinite Boolean sequence and returns a *real* number. The set $\{\mathtt{s} \mid \mathtt{R\ s} \leq \mathtt{x}\}$ is the set of all infinite Boolean sequences, s, that satisfy the condition $(\mathtt{R\ s} \leq \mathtt{x})$.

4.3.2 Formal Verification of CDF Properties

Using the formal specification of the CDF, we are able to verify the classical CDF properties [41] within the HOL theorem prover. The properties are verified under the assumption that the set $\{s \mid R\ s \leq x\}$, where R represents the random variable under consideration, is measurable for all values of x. The formal proofs for these properties not only ensure the correctness of our CDF specification but also play a vital role in proving the correctness of the ITM as will be discussed in Section 4.4.

CDF Bounds

For any real number x,

$$0 \leq F_R(x) \leq 1 \tag{4.12}$$

This property states that if we plot the CDF against its *real* argument x, then the graph of the CDF, F_R, is between the two horizontal lines $y = 0$ and $y = 1$. In other words, the lines $y = 0$ and $y = 1$ are the bounds for the CDF F_R.

The above characteristic can be verified in HOL using the fact that the CDF is basically a probabilistic quantity along with the basic probability law, verified in [36], that states that the probability of an event is always less than or equal to 1 and greater than or equal to 0 ($0 \leq \mathbb{P}(A) \leq 1$).

Theorem 4.3: *CDF Bounds*

$\vdash \forall\ \mathtt{R\ x.}\ \ \mathtt{(measurable\ \{s \mid R\ s \leq x\})} \Rightarrow \mathtt{(0 \leq cdf\ R\ x \leq 1)}$

CDF is Monotonically Increasing

For any two real numbers a and b,

$$if\ a < b,\ then\ F_R(a) \leq F_R(b) \tag{4.13}$$

In mathematics, functions between ordered sets are monotonic if they preserve the given order. Monotonicity is an inherent characteristic of CDFs and the CDF value for a *real* number argument a can never exceed the CDF value of a *real* number argument b if a is less than b.

Using the set theory in HOL, it can be proved that for any two *real* numbers a and b, if $a < b$ then the set of infinite Boolean sequences $\{\mathtt{s} \mid \mathtt{R\ s} \leq \mathtt{a}\}$ is a subset of the set $\{\mathtt{s} \mid \mathtt{R\ s} \leq \mathtt{b}\}$. Then, using the monotone law of the probability function ($A \subseteq B \Rightarrow \mathbb{P}(A) \leq \mathbb{P}(B)$), verified in [36], we proved the monotonically increasing property of the CDF in HOL.

Theorem 4.4: *CDF is Monotonically Increasing*

$$\vdash \forall\ \mathtt{R\ a\ b.}\quad \mathtt{a < b} \wedge \mathtt{(measurable\ \{s \mid R\ s \leq x\})}$$
$$\Rightarrow \mathtt{(cdf\ R\ a \leq cdf\ R\ b)}$$

Interval Probability

For any two real numbers a and b,

$$if\ a < b,\ then\ Pr(a < R \leq b) = F_R(b) - F_R(a) \tag{4.14}$$

This property is very useful for evaluating the probability of a random variable, R, lying in any given interval (a,b] in terms of its CDF.

Using the set theory in HOL, it can be proved that for any two *real* numbers a and b, if $a < b$ then the set of infinite Boolean sequences $\{\mathtt{s} \mid \mathtt{R\ s} \leq \mathtt{b}\}$ is equal to

the union of the disjoint sets $\{\mathtt{s} \mid \mathtt{R\ s} \leq \mathtt{a}\}$ and $\{\mathtt{s} \mid (\mathtt{a} < \mathtt{R\ s}) \wedge (\mathtt{R\ s} \leq \mathtt{b})\}$. Now, the above CDF property can be proved in HOL using the additive law of the probability function $(A \cap B = \emptyset \Rightarrow (\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B))$, verified in [36], along with the closed under complements and countable unions property of $\mathcal{E}$.

Theorem 4.5: *Interval Probability in terms of CDF*

$$\vdash \forall\ \mathtt{R\ a\ b.}\quad \mathtt{a < b} \wedge \mathtt{(measurable\ \{s \mid R\ s \leq x\})}$$
$$\Rightarrow \mathtt{(\mathbb{P}\ \{s \mid (a < R\ s) \wedge (R\ s \leq b)\} = cdf\ R\ b - cdf\ R\ a)}$$

CDF at Negative Infinity

$$\lim_{x \to -\infty} F_R(x) = 0;\ \textit{that is},\ F_R(-\infty) = 0 \tag{4.15}$$

This property states that the value of the CDF tends to 0 as its *real* argument approaches negative infinity or in other words the graph of CDF must eventually approach the line $y = 0$ at the left end of the *real* axis.

We used the formalization of limit of a *real* sequence [32] along with the formalization of the mathematical measure theory [36] in HOL to prove this property. The first step is to prove a relationship between the limit value of the probability of a monotonically decreasing sequence of events A_n (i.e., $A_{n+1} \subseteq A_n$ for every n) and the probability of the countable intersection of all events that can be represented as A_n.

$$\forall A_n.\ \lim_{n \to \infty} Pr(A_n) = Pr(\bigcap_n A_n) \tag{4.16}$$

This relationship, sometimes called the *Continuity Property of Probabilities*, can be used to prove the above CDF property by instantiating it with a decreasing sequence of events represented in Lambda calculus as $(\lambda \mathtt{n.}\{\mathtt{s} \mid \mathtt{R\ s} \leq\ -\mathtt{n}\})$; where n has the HOL data type *nat*: $\{0, 1, 2, \ldots\}$. The LHS of Equation (4.16), with this sequence, represents the CDF for the random variable R when its *real* number argument approaches negative infinity and thus is equal to the LHS of our proof goal

in Equation 4.15. Using the monotonically decreasing nature of the events in the sequence (λn.{s | R s $\leq$ $-$n}), the RHS of Equation (4.16), with this sequence, can be proved to be equal to the probability of an empty set. The CDF at negative infinity property can now be proved using the basic probability law ($\mathbb{P}(\{\}) = 0$), verified in [36], which states that the probability of an empty set is 0.

Theorem 4.6: *CDF at Negative Infinity*

$\vdash \forall$ R. (measurable {s | R s $\leq$ x}) $\Rightarrow$ lim (λ n. cdf R (-n)) = 0

where *lim* is the HOL function for the limit of a *real* sequence [32].

CDF at Positive Infinity

$$\lim_{x \to \infty} F_R(x) = 1;\ \textit{that is},\ F_R(\infty) = 1 \tag{4.17}$$

This property, quite similar to the last one, states that the value of the CDF tends to 1 as its *real* number argument approaches positive infinity or in other words the graph of CDF must eventually approach the line $y = 1$ at the right end of the real axis.

The HOL proof steps for this property are also quite similar to the last one and this time we use the Continuity Property of Probabilities which specifies the relationship between the limit value of the probability of a monotonically increasing sequence of events A_n (i.e., $A_n \subseteq A_{n+1}$ for every n) and the probability of the countable union of all events that can be represented as A_n.

$$\forall A_n.\ \lim_{n \to \infty} Pr(A_n) = Pr(\bigcup_n A_n) \tag{4.18}$$

In this case, we instantiate Equation 4.18 with an increasing sequence of events represented in Lambda calculus as (λn.{s | R s $\leq$ n}). The countable union of all

events in this sequence is the universal set. The CDF at positive infinity property can now be proved in the HOL theorem prover using the basic probability law ($\mathbb{P}(UNIV) = 1$), verified in [36], which states that the probability of the universal set is 1.

Theorem 4.7: *CDF at Positive Infinity*

$\vdash \forall$ R. (measurable {s | R s $\leq$ x}) $\Rightarrow$ lim (λ n. cdf R n) = 1

CDF is Continuous from the Right

For every real number a,

$$\lim_{x \to a^+} F_R(x) = F_R(a) \tag{4.19}$$

where $\lim_{x \to a^+} F_R(x)$ is defined as the limit of $F_R(x)$ as x tends to a through values greater than a. Since F_R is monotone and bounded, this limit always exists.

In order to prove this property in HOL, we used a decreasing sequence of events represented in Lambda calculus as (λn.{s | R s $\leq$ a + $\frac{1}{(n+1)}$}). This sequence of events has been selected in such a way that if the Continuity Property of Probabilities, given in Equation 4.16, is instantiated with this sequence then its LHS represents the CDF for a random variable, R, when its *real* number argument approaches a through values greater than a. Therefore, with this sequence, the LHS of the Continuity Property of Probabilities is equal to the LHS of our proof goal in Equation 4.19. Using the monotonically decreasing nature of the events in the sequence (λn.{s | R s $\leq$ a + $\frac{1}{(n+1)}$}), it can also be proved that the countable intersection of all events in this sequence is the set {s | R s $\leq$ a}. The CDF can now be proved to be continuous from the right as the RHS of the Continuity Property given in Equation 4.16, with the sequence (λn.{s | R s $\leq$ a + $\frac{1}{(n+1)}$}), represents the CDF of random variable at *real* argument a.

Theorem 4.8: *CDF is Continuous from the Right*

$\vdash \forall$ R a. (measurable {s | R s $\leq$ x})

$\Rightarrow$ lim (λ n. cdf R (a + $\frac{1}{(n+1)}$)) = cdf R a

CDF Limit from the Left

For every real number a,

$$\lim_{x \to a^-} F_R(x) = Pr(R < a) \tag{4.20}$$

where $\lim_{x \to a^-} F_R(x)$ is defined as the limit of $F_R(x)$ as x tends to a through values less than a.

This property is quite similar to the previous one and can be proved by instantiating the Continuity Property of Probabilities, given in Equation 4.18, with an increasing sequence of events represented in Lambda calculus as (λn.{s | R s $\leq$ a$-\frac{1}{(n+1)}$}). The LHS of Equation 4.18, with this sequence, represents the CDF for the random variable R when its *real* number argument approaches a through values less than a and is thus equal to the LHS of our proof goal in Equation 4.20. Using the monotonically increasing nature of the events in the sequence (λn.{s | R s $\leq$ a $-\frac{1}{(n+1)}$}), it can be proved that the countable union of all the events in this sequence is the set {s | R s < a} which led us to prove the theorem stating the CDF limit from the left.

Theorem 4.9: *CDF Limit from the Left*

$\vdash \forall$ R a. (measurable {s | R s $\leq$ x})

$\Rightarrow$ lim (λ n. cdf R (a - $\frac{1}{(n+1)}$)) = $\mathbb{P}$ {s | (R s < a})

4.4 Inverse Transform Method

In this section, we present the formal specification of the inverse function for a CDF and the verification of the ITM in HOL. It is the third step in the proposed framework for the formalization of continuous probability distributions as shown in Figure 4.1. The ITM is based on the following proposition [72].

> *Let U be a Standard Uniform random variable. For any continuous CDF F, the random variable X defined by $X = F^{-1}(U)$ has CDF F, where $F^{-1}(U)$ is defined to be the value of x such that $F(x) = U$.*
>
> Mathematically,

$$Pr(F^{-1}(U) \leq x) = F(x) \tag{4.21}$$

4.4.1 Formal Specification of Inverse of the CDF

We define the inverse function for a CDF in HOL as a predicate `inv_cdf_fn`, which accepts two functions, f and g, of type $(real \rightarrow real)$ and returns $True$ if and only if the function f is the inverse of the CDF g according to the above proposition.

Definition 4.4: *Inverse Functions Predicate*

```
⊢ ∀ f g.  inv_cdf_fn f g =
  (∀x.  (0 < g x ∧ g x < 1) ⇒ (f (g x) = x) ∧
    (∀x.  0 < x ∧ x < 1 ⇒ (g (f x) = x))) ∧
  (∀x.  (g x = 0) ⇒ (x ≤ f (0))) ∧
  (∀x.  (g x = 1) ⇒ (f (1) ≤ x))
```

The predicate `inv_cdf_fn` considers three separate cases, the first one corresponds to the strictly monotonic region of the CDF, i.e., when the value of the CDF

g is between 0 and 1. The next two correspond to the flat regions of the CDF, i.e., when the value of the CDF g is either equal to 0 or 1, respectively. These three cases cover all possible values of a CDF since according to Theorem 4.3 the value of CDF can never be less than 0 or greater than 1.

The inverse of a function f, $f^{-1}(u)$, is defined to be the value of x such that $f(x) = u$. More formally, if f is a one-to-one function with domain X and range Y, its inverse function f^{-1} has domain Y and range X and is defined by $f^{-1}(y) = x \Leftrightarrow f(x) = y$, for any y in Y. The composition of inverse functions yields the following result.

$$f^{-1}(f(x)) = x \; for \; all \; x \in X, \quad f(f^{-1}(x)) = x \; for \; all \; x \in Y \tag{4.22}$$

We use the above characteristic of inverse functions in the predicate `inv_cdf_fn` for the strictly monotonic region of the CDF as the CDF in this region is a one-to-one function. On the other hand, the CDF is not injective when its value is either equal to 0 or 1. Consider the example of some CDF, F, which returns 0 for a *real* argument a. From Theorems 4.3 and 4.4, we know that the CDF F will also return 0 for all *real* arguments that are less than a as well, i.e., $\forall x.\ x \leq a \Rightarrow F(x) = 0$. Therefore, no inverse function satisfies the conditions of Equation (4.22) for the CDF in these flat regions. When using the paper-and-pencil proof approach, this issue is usually resolved by defining the inverse function of a CDF in such a way that it returns the infimum (inf) of all possible values of the *real* argument for which the CDF is equal to a given value, i.e., $f^{-1}(u) = inf\{x | f(x) = u\}$ [20], where f represents the CDF. Even though this approach has been shown to analytically verify the correctness of the ITM [20], it was not found to be sufficient enough for a formal definition in our case. This is due to the fact that in order to simplify the formalization task, Hurd

[36] used the standard *real* numbers $\mathbb{R}$, formalized in HOL by Harrison [32], rather than the extended *real* numbers $\overline{\mathbb{R}} = \mathbb{R} \bigcup \{-\infty, +\infty\}$ to formalize the mathematical measure theory. Thus, if the inf function is used to define the inverse function, then the problem arises for the case when the value of the CDF is equal to 0. For this case, the set $\{x | f(x) = 0\}$ becomes unbounded at the lower end because of the CDF property given in Theorem 4.6 and thus the value of the inverse function becomes undefined. In order to overcome this problem, we use two separate cases for the two flat regions in the predicate `inv_cdf_fn`. According to this definition the inverse function of a CDF is a function that returns the maximum value of all arguments for which the CDF is equal to 0 and the minimum value of all arguments for which the CDF is equal to 1.

4.4.2 Formal Verification of the ITM

The correctness theorem for the ITM can now be expressed in HOL as follows:

Theorem 4.10: *Inverse Transform Method*

```
⊢ ∀ f g x.  (is_cont_cdf_fn g) ∧ (inv_cdf_fn f g)
   ⇒ (ℙ {s | f (std_unif_cont s) ≤ x} = g x)
```

The antecedent of the above implication checks if f is a valid inverse function of a continuous CDF g. The predicate `inv_cdf_fn` has been described in the last section and ensures that the function f is a valid inverse of the CDF g. The predicate `is_cont_cdf_fn` accepts a *real*-valued function, g, of type $(real \rightarrow real)$ and returns $True$ if and only if it represents a continuous CDF. A *real*-valued function can be characterized as a continuous CDF if it is a continuous function and satisfies the CDF properties given in Theorems 4.4, 4.6 and 4.7. Therefore, the predicate `is_cont_cdf_fn` is defined in HOL as follows:

Definition 4.5: *Continuous CDF Predicate*

```
⊢ ∀ g.  is_cont_cdf_fn g =
  (∀ x.  g contl x) ∧
  (∀ a b.  a < b ⇒ g a ≤ g b) ∧
  (lim (λ n.  g (-n)) = 0) ∧
  (lim (λ n.  g (n)) = 1)
```

where (∀ x.g `contl` x) represents the HOL predicate that returns *True* if g is a continuous function [32] for all x.

The conclusion of the implication in Theorem 4.10 represents the correctness proof of the ITM given in Equation (4.21). The function `std_unif_cont` in this theorem is the formal definition of the Standard Uniform random variable, described in Section 4.2. Theorem 4.1 can be used to reduce the proof goal of Theorem 4.10 to the following subgoal:

$$(\mathbb{P}\ \{\mathtt{s} \mid \mathtt{f(std\_unif\_cont\ s)} \leq \mathtt{x}\} = \mathbb{P}\ \{\mathtt{s} \mid \mathtt{std\_unif\_cont\ s} \leq \mathtt{g\ x}\}) \qquad (4.23)$$

Next, we use the theorems of Section 4.2 and 4.3 along with the formalized measure and probability theories in HOL [36] to prove the measurability of the sets that arise in this verification, i.e., they are in $\mathcal{E}$.

Lemma 4.8: *Measurability of Sets in ITM*

```
⊢ ∀ f g x.  (is_cont_cdf_fn g) ∧ (inv_cdf_fn f g)
  ⇒ (measurable {s | f(std_unif_cont s) ≤ x}) ∧
    (measurable {s | (std_unif_cont s) ≤ g x}) ∧
    (measurable {s | f(std_unif_cont s) = x})
```

Equation (4.23) can now be proved using Lemma 4.8, the theorems from Section 4.2 and 4.3 and Hurd's formalization of probability theory in HOL. The main advantage of the formally verified ITM (i.e., Theorem 4.10) is the simplification of the verification task of proving the CDF property of a random variable that is expressed in terms of the Standard Uniform random variable. Originally such verification involves reasoning based on the measure, probability and *real* number theories and the theorems related to the Standard Uniform random variable. Using the formally verified ITM, the CDF verification goal can be broken down to two simpler sub-goals, i.e., (1) verifying that a function g, of type $(real \rightarrow real)$, represents a valid CDF and (2) verifying that another function f, of type $(real \rightarrow real)$, is a valid inverse of the CDF g. The verification of these subgoals only involves some arithmetic reasoning.

4.5 Continuous Random Variables in HOL

In order to demonstrate the framework for the formalization of continuous random variables, given in Figure 4.1, we now present the formal specification of four continuous random variables; Exponential(λ), Uniform(a, b), Rayleigh(λ) and Triangular($0, a$). The framework is then used to verify the correctness of these random variables by proving their corresponding CDF properties in the HOL theorem prover.

4.5.1 Formal Specification of Continuous Random Variables

All continuous random variables for which the inverse of the CDF exists in a closed mathematical form can be expressed in terms of the Standard Uniform random variable according to the ITM proposition given in Section 4.4. We selected four such commonly used random variables: Exponential(λ), Uniform(a, b), Rayleigh(λ) and Triangular($0, a$), which are formally expressed in terms of the formalized Standard

Uniform random variable `std_unif_cont` in Table 4.1 as HOL functions `exp_rv`, `uniform_rv`, `rayleigh_rv` and `triangular_rv`, respectively. Table 4.1 lists the CDF relations of these random variables as well in the last column. The functions `ln`, `exp` and `sqrt` in Table 4.1 are the HOL functions for *logarithm*, *exponential* and *square root*, respectively [32].

Distribution	Formalized Random Variable	CDF
Exponential(l)	$\vdash \forall s\ l.\ exp\_rv\ l\ s\ =$ $-\frac{1}{l}ln(1 - std\_unif\_cont\ s)$	0 if $x \leq 0$; $1 - exp^{-lx}$ if $0 < x$.
Uniform(a, b)	$\vdash \forall s\ a\ b.\ uniform\_rv\ a\ b\ s\ =$ $(b - a)(std\_unif\_cont\ s) + a$	0 if $x \leq a$; $\frac{x-a}{b-a}$ if $a < x < b$; 1 if $b \leq x$.
Rayleigh(l)	$\vdash \forall s\ l.\ rayleigh\_rv\ l\ s\ =$ $l\ sqrt(-2ln(1 - std\_unif\_cont\ s))$	0 if $x \leq 0$; $1 - exp^{\frac{-x^2}{2l^2}}$ if $0 < x$.
Triangular($0, a$)	$\vdash \forall s\ a\ .\ triangular\_rv\ a\ s\ =$ $a(1 - sqrt(1 - std\_unif\_cont\ s))$	0 if $x \leq 0$; $(\frac{2}{a}(x - \frac{x^2}{2a}))$ if $x < a$; 1 if $a \leq x$.

Table 4.1: Continuous Random Variables (for which CDF^{-1} exists)

4.5.2 Formal Verification of Continuous Random Variables

The first step in verifying the CDF property of a continuous random variable, using the correctness theorem of the ITM, is to express the given continuous random variable as $F^{-1}(U\ s)$, where F^{-1} is a function of type $(real \rightarrow real)$ and U represents the formalized Standard Uniform random variable. For example, the Exponential(λ) random variable given in Table 4.1 can be expressed as $(\lambda \mathtt{x}.\ -\frac{1}{\mathtt{l}}\mathtt{ln}(1 - \mathtt{x}))(\mathtt{std\_unif\_cont\ s})$. Similarly, we can express the CDF of the given random variable as $F(x)$, where F is a function of type $(real \rightarrow real)$ and x is a *real* data type variable. For example, the CDF of the Exponential(λ) random variable can be expressed as $(\lambda \mathtt{x.\ if\ x} \leq \mathtt{0\ then\ 0\ else\ 1} - \mathtt{exp}(-\mathtt{lx}))\ \mathtt{x}$.

The next step is to prove that the function F defined above represents a valid continuous CDF and the function F^{-1} is a valid inverse function of the CDF F. The predicates `is_cont_cdf_fn` and `inv_cdf_fn`, defined in Section 4.4, can be used for this verification and the corresponding theorems for the Exponential(λ) random variable are given below

$$\forall \mathtt{l}.\ \mathtt{is\_cont\_cdf\_fn}\ (\lambda \mathtt{x}.\ \mathtt{if\ x}\ \leq\ \mathtt{0\ then\ 0\ else}\ (1 - \mathtt{exp}\ (-\mathtt{lx}))) \tag{4.24}$$

$$\forall \mathtt{l}.\ \mathtt{inv\_cdf\_fn}\ (\lambda \mathtt{x}.\ -\frac{1}{\mathtt{l}}\mathtt{ln}(1 - \mathtt{x}))\ (\lambda \mathtt{x}.\ \mathtt{if\ x}\ \leq\ \mathtt{0\ then\ 0\ else}(1 - \mathtt{exp}(-\mathtt{lx}))) \tag{4.25}$$

The above Equations along with Theorem 4.10 and Lemma 4.8 can be used to verify the CDF and the measurability of the sets corresponding to the given continuous random variable. These theorems for the Exponential(λ) random variable are given below

Theorem 4.11: *CDF for the Exponential Random Variable*

```
⊢ ∀ l x.  (0 < l)
   ⇒ cdf (λs.  exp_rv l s) x =
      if x ≤ 0 then 0 else (1 - exp (-l x))
```

Theorem 4.12: *Measurability for the Exponential Random Variable*

```
⊢ ∀ l x.  (0 < l)
⇒ (measurable {s | exp_rv r s ≤ x}) ∧
   (measurable {s | exp_rv r s = x})
```

The above results allow us to formally reason about interesting probabilistic properties of continuous random variables within a higher-order-logic theorem prover.

The measurability of the sets $\{s|\ F^{-1}(U\ s) \leq x\}$ and $\{s|\ F^{-1}(U\ s) = x\}$ can be used to prove that any set that involves a relational property with the random variable $F^{-1}(U\ s)$, e.g., $\{s \mid F^{-1}(U\ s) < x\}$ and $\{s \mid F^{-1}(U\ s) \geq x\}$, is measurable because of the closed under complements and countable unions property of $\mathcal{E}$. The CDF properties proved in Section 4.3 can then be used to determine probabilistic quantities associated with these sets.

The CDF and measurability properties of the rest of the continuous random variables, given in Table 4.1, can also be proved in a similar way. For illustration purposes the corresponding CDF theorems are given below

Theorem 4.13: *CDF for the Uniform Random Variable*

$\vdash \forall$ a b x. (a < b)

$\Rightarrow$ cdf (λs. uniform_rv a b s) x =

if x $\leq$ a then 0 else (if x < b then $\frac{x-a}{b-a}$ else 1)

Theorem 4.14: *CDF for the Rayleigh Random Variable*

$\vdash \forall$ x l. (0 < l)

$\Rightarrow$ cdf (λs. rayleigh_rv l s) x =

if x $\leq$ 0 then 0 else (1 - $\frac{exp(x^2)}{(2l^2)}$)

Theorem 4.15: *CDF for the Triangular Random Variable*

$\vdash \forall$ a x. (0 < a)

$\Rightarrow$ cdf (λs. triangular_rv a s) x =

if (x $\leq$ 0) then 0 else

(if (x < a) then ($\frac{2}{a}$(x - $\frac{x^2}{2a}$)) else 1)

4.6 Probabilistic Analysis of Roundoff Error in a Digital Processor

The formalized continuous random variables can be utilized in the proposed probabilistic analysis approach to formally model and reason about continuous random phenomenon. In this section, we illustrate this statement by considering the verification of quantitative probabilistic properties related to a simple system that utilizes a continuous random variable.

Consider the roundoff error for a particular digital processor to be uniformly distributed over the interval [-5x10^{-12}, 5x10^{-12}]. An engineering team is interested in verifying that the probability of the event when the roundoff error in this digital processor is greater than 2x10^{-12} is less than 0.33 and the probability that the final result fluctuates by $\pm$1x10^{-12} with respect to the actual value is precisely equal to 0.2. This section illustrates the process of tackling the verification of these probabilistic properties in HOL.

Our approach for the verification of probabilistic properties in HOL revolves around the fact that any probabilistic property related to a random variable can be expressed in terms of its CDF. Consider the case of interval probabilities, i.e., properties that associate probabilities to the event that a random variable falls in a particular interval of the *real* line, like the ones mentioned above for the case of the digital processor. We now show how the CDF can be used to express any interval property by splitting the *real* line in three disjoint intervals; $(-\infty, a]$, $(a, b]$, (b, ∞). Determining the probability for the first interval is quite straightforward since the CDF for a random variable, R, with a *real* argument, a, can be used directly to find the probability that R lies in the interval $(-\infty, a]$. Whereas, the probability that

a random variable, R, will lie in the interval $(a, b]$ can be determined by its CDF values for the *real* arguments a and b as has been proved in Theorem 4.5. For the third interval, we first use the set theory in HOL to prove that for any *real* value b, the set of infinite Boolean sequences $\{\mathtt{s} \mid \mathtt{b} < \mathtt{R\ s}\}$ is the complement of the set $\{\mathtt{s} \mid \mathtt{R\ s} \leq \mathtt{b}\}$. The probability that a random variable, R, lies in the interval (b, ∞) can now be represented in terms of its CDF by using the complement law of the probability function ($\mathbb{P}(S) = 1 - \mathbb{P}(\bar{S})$).

Theorem 4.16: *Interval Probability* (b, ∞) *in terms of CDF*

$$\vdash \forall\ \mathtt{R\ b.}\quad \mathtt{(measurable\ \{s \mid R\ s} \leq \mathtt{b\})}$$
$$\Rightarrow \mathbb{P}\ \mathtt{\{s \mid b < R\ s\} = 1 - (cdf\ R\ b)}$$

We are now in the position of formally verifying the given probabilistic properties by modeling the roundoff error as a continuous Uniform(a, b) random variable, formalized in Section 4.5, in the interval [-5x10^{-12}, 5x10^{-12}].

We proceed to verify the first probabilistic property, which checks if the probability of the event when the roundoff error in this digital processor is greater than 2x10^{-12} is less than 0.33, by instantiating Theorem 4.16 with the random variable (λs. uniform_rv $-$ 5x10^{-12} 5x 10^{-12} s) and the *real* value 2x10^{-12}. Now the property can be verified by simplifying the result using the formally verified CDF relation for the Uniform(a, b) random variable, given in Theorem 4.13, and some arithmetic reasoning in HOL.

Theorem 4.17: *Probability (roundoff error* $>$ *2x*10^{-12}*)* $<$ *0.33*

$$\vdash \mathbb{P}\ \mathtt{\{s \mid 2x10^{-12} < uniform\_rv\ -5x10^{-12}\ 5x10^{-12}\ s\ \}} < 0.33$$

Similarly the second property can be verified by checking if the probability of the continuous Uniform[-5x10^{-12}, 5x10^{-12}] random variable falling in the interval [-1x10^{-12}, 1x10^{-12}] is equal to 0.2. We proceed in this direction by instantiating the

CDF property, verified in Theorem 4.5, with the *real* values -1x10^{-12}, 1x10^{-12} for variables a, b and the random variable (λs. uniform_rv $-$ 5x10^{-12} 5x10^{-12} s) for random variable R. Now the property can be verified by simplifying the result using the formally verified CDF relation for the Uniform(a, b) random variable, given in Theorem 4.13, and some arithmetic reasoning in HOL.

Theorem 4.18: *Probability (roundoff error lies in [-1x10^{-12}, 1x10^{-12}]) = 0.2*

$$\vdash \mathbb{P}\ \{s \mid (\text{-1x10}^{-12} < \text{uniform\_rv -5x10}^{-12}\ \text{5x10}^{-12}\ s) \wedge (\text{uniform\_rv -5x10}^{-12}\ \text{5x10}^{-12}\ s \leq \text{1x10}^{-12})\} = 0.2$$

The above example illustrates the usefulness of formalized continuous random variables in verifying probabilistic quantities with 100% precision. It is a novelty that is not available in the existing computer-based probabilistic analysis approaches.

4.7 Summary and Discussions

In this chapter, we described the construction details of a framework for the formalization and verification of all continuous probability distributions for which the inverse of the CDF can be expressed in a closed mathematical form. We demonstrated the practical effectiveness of our framework by formalizing four continuous probability distributions; Exponential(λ), Uniform(a, b), Rayleigh(λ) and Triangular$(0, a)$ and formally verifying the corresponding CDF relations. The framework is generic and can be used to model other continuous random variables as well, such as, Cauchy, Pareto and Raised Cosine. To the best of our knowledge, this is the first time that a higher-order-logic formalization and formal verification of continuous random variables has been presented.

The continuous random variables are extensively used to model random behaviors in the probabilistic analysis of many engineering and scientific applications [84]. The formalized continuous random variables and the ability to precisely reason about their probability distribution properties pave the path to precisely tackle the analysis of such applications within the sound core of the HOL theorem prover. For illustration purposes, we utilized the formalized Uniform(a, b) random variable to reason about a couple of probabilistic properties regarding the roundoff error in a digital processor.

So far in this thesis, we have described some of the pre-requisites for conducting formal probabilistic analysis using a theorem prover, i.e., the formalization and verification of statistical properties regarding discrete random variables in Chapter 3 and the formalization and verification of probabilistic properties regarding continuous random variables in this chapter. In the next chapter, we build upon these foundations to demonstrate the feasibility of the proposed probabilistic analysis approach by presenting the analysis of a real-world system, i.e., the Stop-and-Wait protocol.

Chapter 5

Case Study: Stop-and-Wait Protocol

Real-time systems usually involve a subtle interaction of a number of distributed components and have a high degree of parallelism, which makes their analysis quite complex. Thus, traditional techniques, such as simulation, or the state-based formal methods usually fail to produce reasonable results. In this chapter, we demonstrate the usefulness and feasibility of the proposed probabilistic analysis approach by utilizing it for the performance analysis of real-time systems. For illustration purposes, we present the analysis of the Stop-and-Wait protocol, which is a classical example of real-time systems. The functional correctness of the protocol is verified by proving that the protocol ensures reliable data transfers. Whereas, the average message delay relation is verified in HOL for the sake of performance analysis.

5.1 Introduction

Real-time systems can be characterized as systems for which the correctness of an operation is dependant not only on its logical correctness but also on the time taken. Some commonly used real-time system applications include embedded systems, digital circuits with uncertain delays and communication protocols. Due to the increased usage of real-time systems in safety critical and extremely sensitive applications such as medicine, transportation and space travel, their correctness and performance has become imperative. The functional verification and performance evaluation tasks in this domain are quite challenging as the present age real-time systems usually involve a subtle interaction of a number of distributed components and have a high degree of parallelism. Thus, traditional techniques, such as simulation, fail to produce reasonable results. On the other hand, formal methods offer a promising solution.

A number of elegant approaches for the formal functional verification of real-time systems can be found in the open literature using state-based or theorem-proving techniques (e.g. [1, 12, 2, 5]). However, most of these existing formal verification tools are only capable of specifying and verifying hard deadlines, i.e., properties where a late response is considered to be incorrect. Recently, several state-based formal approaches have been proposed for the verification of soft deadlines, which lead to probabilistic analysis, for real-time systems (e.g. [46, 11, 47]). However, all these approaches share the same inherent limitation that is the reduced expressive power of their automata based or Petri net based specification formalism. On top of that, either there is no mechanism to verify statistical properties in these techniques or even if it does exist then the underlying infrastructure cannot be regarded as completely formal, as has already been pointed out earlier. For example, Duflot *et al.* [21] used the PRISM model checker to conduct the performance analysis of a CSMA/CD protocol, which

is a real-time communication protocol, based on expectations. The results obtained are approximate as has been clearly stated in [21].

Due to the immaturity of formal methods in verifying soft deadlines and using them for performance evaluation of real-time systems, the current state-of-the-art is based on constructing abstract models that are analyzed by simulation or by applying stochastic process theory for this purpose. Besides the inaccuracy of the results by simulation based methods and the drawbacks associated with paper-and-pencil proof methods, a major limitation of this approach is that the model used for performance analysis is usually quite far in abstraction level from the one used for functional verification. This fact makes the equivalence verification between these two models very difficult, if not impossible, and thus leaves a major hole in the soundness of the functional verification and performance analysis tasks.

The proposed theorem proving based probabilistic analysis approach allows us to overcome the above mentioned limitations. The idea is to formally specify the given real-time system as a logical conjunction of higher-order-logic predicates [12], where each one of these predicates define an autonomous component or process of the given real-time system, while representing the unpredictable or random elements in the system as formalized random variables. The functional correctness and the performance related probabilistic and statistical properties for various parameters for this formal model can then be verified using an interactive theorem prover with the help of the formalization and verification support presented in this thesis so far. Since the analysis is conducted within the core of a mechanical theorem prover, there would be no question about the soundness or the precision of the results. Also, there is no equivalence verification required between the models used for functional verification and performance evaluation as the same formal model is used for both of these analysis

in this approach.

In order to support the above mentioned claims, we present the functional verification and performance analysis of the Stop-and-Wait protocol, which is a real-time system, in this chapter. The Stop-and-Wait protocol utilizes the principles of error detection and retransmission and is a fundamental mechanism for reliable communication between computers. Indeed, it is one of the most important parts of the Internet's *Transmission Control Protocol* (TCP). The main motivation behind selecting the Stop-and-Wait protocol as a case study for our approach is its widespread popularity in the literature regarding real-time system analysis methodologies. Stop-and-Wait protocol and some of its closely related variants have been checked formally for functional verification using both theorem proving and state-based formal approaches (e.g. [82, 12, 33, 8, 25]) and their performance has been analyzed using a number of innovative formal or semi-formal techniques (e.g. [53, 79, 85, 29]). But like other real-time systems, to the best of our knowledge, there is no mechanized approach reported in the literature that utilizes a single model of the Stop-and-Wait protocol and could verify its functional correctness and precisely analyze its performance. We fill this gap in this chapter as we present the functional verification and the performance analysis, based on the precise average (expectation) delay for a single message transmission, for the Stop-and-Wait protocol using the HOL theorem prover. The Stop-and-Wait protocol is a classical example of a real-time system and therefore we believe that the probabilistic analysis approach followed in this case study can be essentially utilized for the probabilistic analysis of any other real-time system as well.

This chapter is organized as follows. We begin by presenting an informal description of the Stop-and-Wait protocol along with its average delay characteristic.

Then, in Section 5.3, we provide a higher-order-logic specification of the Stop-and-Wait protocol. We verify the functional correctness of this specification using the HOL theorem prover in Section 5.4. Then, in Section 5.5, we conduct the performance analysis of the Stop-and-Wait protocol based on its formal specification in HOL. We mainly verify the message delay characteristic of the Stop-and-Wait protocol, which is the most widely used performance measuring parameter for communication protocols, first under noise-free conditions and then with the consideration of the channel noise. Finally, the chapter is concluded in Section 5.6 with some discussions.

5.2 Protocol Description

Stop-and-Wait [49] is a basic *Automatic Repeat Request* (ARQ) protocol that ensures reliable data transfers across noisy channels. In a Stop-and-Wait system, both sending and receiving stations have error detection capabilities. The operation is illustrated in Figure 5.1 using the following notation.

- t_f: Data message transmission time
- t_a: Acknowledgement (ACK) message transmission time
- t_{prop}: One-way signal propagation delay between transmitter and receiver
- t_{proc}: Processing time required for error detection in the received message at both transmitter and receiver
- t_{out}: Timeout period

The transmitter sends a data message to the receiver and spends t_f time units in doing so. It then stops and waits to receive an ACK of reception of that message

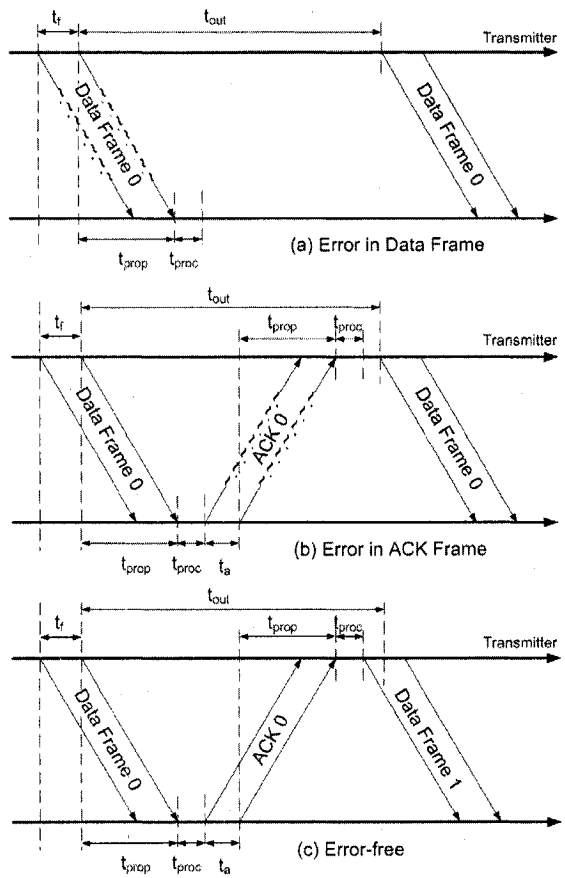

Figure 5.1: Stop-and-Wait Operation

from the receiver. If no ACK is received within a given time out period, t_{out}, the data message is resent by the transmitter and once again it stops and starts waiting for the ACK (Figure 5.1.a). If an ACK is received within the given t_{out} period then the transmitter checks the received message for errors during the next t_{proc} time units. If errors are detected then the ACK is ignored and the data message is resent by the transmitter after t_{out} expires and once again the transmitter stops and waits for the ACK (Figure 5.1.b). Thus, the main idea is that the transmitter keeps on retransmitting the same data message, after a pre-defined time-out period, t_{out}, until and unless it receives a corresponding error-free ACK message from the receiver.

When an error-free ACK message is finally received then the transmitter transmits the next data message in its queue (Figure 5.1.c).

The receiver is always waiting to receive data messages. When a new message arrives, the receiver checks it for errors during the next t_{proc} time units. If errors are detected then the data message is ignored and the receiver continues to be in the wait state (Figure 5.1.a), otherwise it initiates the transmission of an ACK message, which takes t_a time units (Figure 5.1.b,c).

Under the above mentioned conditions, the ACK message cannot be received before $t_{prop} + t_{proc} + t_a + t_{prop} + t_{proc}$ units of time after sending out a data message. It is, therefore, necessary to set $t_{out} \geq 2(t_{prop} + t_{proc}) + t_a$, i.e., the retransmission must not be allowed to start before the expected arrival time of the ACK is lapsed, for reliable communication between transmitter and receiver.

ARQ allows the transmitting station to transmit a specific number, usually termed as *sending window*, of messages before receiving an ACK frame and the receiving station to receive and store a specific number, usually termed as *receiving window*, of error-free messages even if they arrive out-of-sequence. Generally, both the *sending window* and the *receiving window* are assigned the same value, which is termed as the *window size* of the ARQ protocol [80]. The *window size* for the Stop-and-Wait protocol is 1, as can be observed from its transmitter and receiver behavior descriptions given above.

In order to distinguish between new messages and duplicates of previous messages at the receiver or transmitter, a sequence number is included in the header of both data and ACK messages [49]. It has been shown that, for correct ARQ operation, the number of distinct sequence numbers must be at least equal to twice the *window size* [83]. Thus, the simplest and the most commonly used version of the

Stop-and-Wait protocol uses two distinct sequence numbers (0 and 1) and is known as the *Alternating Bit Protocol* (ABP). The transmitter keeps track of the sequence number of the last data message it had sent, its associated timer and the message itself in case a retransmission is required. Whereas, the receiver keeps track of the sequence number of the next data message that it is expecting to receive and discards out-of-sequence data messages. On the other hand, when an in-sequence data message arrives at the receiver, it updates its sequence number by performing a modulo-2 addition with the number 1, i.e., 0 is updated to 1 and 1 is updated to 0, and responds with the corresponding ACK message. Similarly, if an out-of-sequence ACK message appears at the transmitter, it ignores it and retains the sequence number of the last data message it had sent. Whereas, in the case of the reception of an in-sequence ACK message, the sequence number at the transmitter is also updated by performing a modulo-2 addition by 1, which becomes the sequence number of the next data message as well. More details about sequence numbering in the Stop-and-Wait protocol can be found in [49].

The most widely used performance metric for Stop-and-Wait protocol is the time required for the transmitter to send a single data message and know that it has been successfully received at the receiver. In the case of error-free or noiseless channels, which do not reorder or loose messages (Figure 5.1.c), the message transmission delay is given by

$$t_f + t_{prop} + t_{proc} + t_a + t_{prop} + t_{proc} \tag{5.1}$$

On the other hand, in the presence of noise, every damaged or lost message (data or ACK) will cause a retransmission from the transmitter and thus wastes $t_f + t_{out}$ units of time (Figure 5.1.a,b). Whereas, the final successful transmission will take the

amount of time given in Equation (5.1). In order to obtain more concise information about this delay, we consider the probability, p, of a message transmission being in error. This allows us to model the number of retransmissions in the Stop-and-Wait protocol in terms of a Geometric random variable, which returns the number of trials required to achieve the first success, with success probability $1 - p$. Therefore, the delay of the Stop-and-Wait protocol can be mathematically expressed as

$$(t_f + t_{out})(G_{(1-p)} - 1) + t_f + t_{prop} + t_{proc} + t_a + t_{prop} + t_{proc} \tag{5.2}$$

where G_x denotes a Geometric random variable with success probability x. The above representation allows us to express the average delay of a single data message in a Stop-and-Wait protocol using the expectation or average value of a Geometric random variables as follows

$$\frac{(t_f + t_{out})p}{1-p} + t_f + t_{prop} + t_{proc} + t_a + t_{prop} + t_{proc} \tag{5.3}$$

5.3 Formal Specification in HOL

A real-time system and its environment may be viewed as a bunch of concurrent, communicating processes that are autonomous, i.e., they can communicate asynchronously. The behavior of these processes over time may be specified by higher-order-logic predicates on positive integers [12]. Whereas, these positive integers 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. The behavior of a real-time system can now be formally specified by combining the corresponding process specifications (higher-order-logic predicates) using logical conjunction. In a similar way, additional

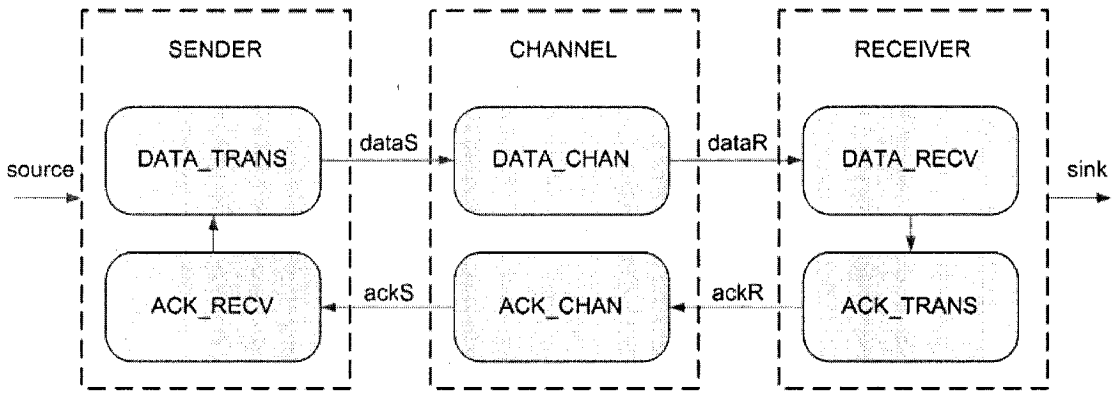

Figure 5.2: Logical Structure of an ARQ Protocol

constraints for the real-time system such as initial conditions or any assumptions, if required to ensure the correct behavior of the model, can also be defined as predicates and combined with its formal specification using logical conjunctions.

Based on the above mentioned approach, we formally specify the Stop-and-Wait protocol as a combination of six processes, as shown in Figure 5.2. The protocol mainly consists of three major modules, i.e., the sender or the transmitter, the receiver and the communication channel. Each one of these modules can be subdivided into two processes as both the sender and the receiver transmit messages and receive them and the channel between the sender and receiver consists of two logical channels: one carrying data messages from the sender to the receiver and one carrying ACK messages in the opposite direction.

In the following, we present the data type definitions, the six higher-order-logic predicates, corresponding to each one of the processes in Figure 5.2, and finally the formal specification of the Stop-and-Wait protocol, which also includes the predicates for assumptions and initial conditions. We include the timing information associated with every action in these predicates so that the corresponding model can be utilized

to reason about the message delay characteristic of the Stop-and-Wait protocol.

5.3.1 Type Definitions

The input to the Stop-and-Wait protocol, `source`, is basically a list of data messages that can be modeled in HOL by a list of α elements

```
source:  α list
```

where α represents any concrete HOL data type such as a record, a character, an integer or an n-bit word. The output of the protocol, `sink`, is also a list of data messages that grows with time as new data messages are delivered to the receiver. It can be modeled in HOL as follows:

```
sink:  time -> α list
```

where *time* is assigned the HOL data-type for positive integers, *num*, and represents physical time in this case. This kind of variable, which is time dependent, is termed as a *history* in this chapter.

The arrows in Figure 5.2 between processes represent information that is shared between the sender, channel and receiver. Data messages are transmitted from the sender to the receiver (`dataS, dataR`) and ACK messages are transmitted in the opposite direction (`ackR, ackS`). These messages are transmitted across the Stop-and-Wait protocol in a form of a packet, which can be modeled in HOL as a pair containing a sequence number and a message element

```
packet:  num x α
```

where *num* is used here for the sequence number and the α represents the message. Since we are dealing with an unreliable channel, the output of a channel may or may

not be a packet. In order to model the no-packet case in HOL, a data-type non_packet is defined, which has only one value, i.e., one. Every message can either be of type packet or of type non_packet.

```
message:  packet + non_packet
```

5.3.2 Data Transmission

The process DATA_TRANS in Figure 5.2 characterizes the data transmission behavior of the Stop-and-Wait protocol and the corresponding predicate is defined as follows.

Definition 5.1: *Data Transmission Behavior for Stop-and-Wait Protocol*

```
⊢ ∀ ws sn dataS s rem i ackS tout tf dtout dtf.
  DATA_TRANS_STOP_WAIT ws sn dataS s rem i ackS
    tout tf dtout dtf =
  ∀ t.
    (if ¬NULL (tli (i t) (rem t)) ∧ i t < ws then
      (if dtf t = 0 then
        (i (t + 1) = i t + 1) ∧
        (dtout (t + 1) = tout - 1) ∧
        (dtf (t + 1) = tf) ∧
        (dataS t = new_packet (mod_n_add (s t) (i t) sn)
          (hdi (i t) (rem t)))
      else
        (i (t + 1) = i t) ∧
        (dtout (t + 1) = tout) ∧
        (dtf (t + 1) = dtf t - 1) ∧
```

```
    (dataS t = set_non_packet))
else
  (dtf (t + 1) = tf) ∧ (dataS t = set_non_packet)) ∧
  (if
    (dtout t = 1) ∨
    good_packet (ackS t) ∧
    mod_n_sub (label (ackS t)) (s t) sn < ws
  then
    (i (t + 1) = i t - 1) ∧ (dtout (t + 1) = tout)
  else
    (i (t + 1) = i t) ∧ (dtout (t + 1) = dtout t - 1))
```

The variables ws and sn represent the *window size* and the number of distinct sequence numbers available for the protocol, respectively. By using these variables in our definitions, instead of their corresponding fixed values of 1 and 2 for the case of the Stop-and-Wait protocol, we attain two benefits. Firstly, it makes our definitions more generic as they can now be used, with minor updates, to formally model the corresponding processes of other ARQ protocols, such as Go-Back-N and Selective-Repeat [49], as well. Secondly, this allows us to establish a logical implication between our definitions for the six processes (Figure 5.2) to the corresponding definitions for the Sliding Window protocol, given in [12]. This relationship can be used to inherit the functional correctness theorem, verified for the Sliding Window protocol in [12], for our Stop-and-Wait protocol model and thus saves us a considerable amount of verification time and effort. More details on this are given in Section 5.4. It is important to note that in order to model the correct behavior for the Stop-and-Wait protocol, we will assign the values of 1 and 2 to the variables ws and sn, respectively, in an assumption

that is used in all of the theorems that we verify for the Stop-and-Wait protocol.

The history `dataS` represents the data messages transmitted by the sender at any particular time. The history `s` represents, modulo `sn`, the sequence number of the first unacknowledged data message. Data remaining to be sent at any time is represented by the history `rem` that has type $time \rightarrow \alpha$. Whereas, the history `i`: $time \rightarrow num$ is used to identify the number of data messages, at any particular time, that have been transmitted by the sender but are still unacknowledged by the receiver. The history `ackS` represents the ACK messages received by the sender at any particular time. The variables `tout` and `tf` hold the values for the t_{out} and t_f delays and histories `dtout` and `dtf` keep track of the timers associated with these delays.

The HOL functions `tli` and `hdi`, in the above definition, accept two arguments, i.e., a list l and a positive integer n, and return the tail of the list l starting from its n^{th} element and the n^{th} element of the list l, respectively. Whereas, the functions `new_packet` and `set_non_packet` declare a message of type `packet` (using its two arguments) and `non_packet`, respectively. The function `label` returns the sequence number of a `packet` and the predicate `good_packet` checks the message type of its argument and returns *False* if it is `non_packet` and *True* otherwise. The functions `mod_n_add` and `mod_n_sub` return the modulo-n, where n is their third argument, addition or subtraction results of their first two arguments, respectively.

The definition of `DATA_TRANS_STOP_WAIT` should be read as follows. At all times `t`, check for the transmission conditions, i.e., there is data available to be transmitted (`¬NULL (tli (i t) (rem t))`) and the number of unacknowledged messages is less than the *window size* (`i t < ws`). If the transmission conditions are satisfied, then wait for the next t_f time units, i.e., decrement the timer `dtf` by one at every

increment of the time until it reaches 0 and during this time maintain the values of histories `i` and `dtout` while holding the transmission of a new packet to the channel. Once t_f time units have elapsed, i.e., the contents of `dtf` timer become 0, then instantly transmit the `(i t)`th message in the window `(hdi (i t) (rem t))` using the sequence number `(mod_n_add (s t) (i t) sn)`. Whereas, in the next increment of time `t` increment the value of the history `i` by 1, activate the timer `dtout`, associated with the t_{out} delay, by decrementing its value by 1 and initialize the timer `dtf`, associated with the t_f delay, to its default value of `tf`. On the other hand, for all times `t` for which one of the transmission conditions is not satisfied, no message is transmitted (`set_non_packet`) and the initial value of the `dtf` timer is maintained. The values of `i` and `dtout`, under the no transmission conditions, depend on the event if the timer `dtout` reaches 1 or an ACK message `(good_packet (ackS t))` is received for a data message that has been sent and not yet acknowledged, i.e., if the difference between the label of `(ackS t)` and the sender's sequence number is less than `ws`, i.e, `(mod_n_sub (label (ackS t)) (s t) sn < ws)`. If this event happens, then the timer `dtout` is initialized to its default value `tout` and the value of `i` is decremented by 1 in the next increment of time `t`. Otherwise, we remain in the wait state until the timer `dtout` expires or a valid ACK is received, while maintaining the value of `i` and decrementing the timer `dtout` by one at every increment of the time `t`.

5.3.3 Data Reception

The process `DATA_RECV` in Figure 5.2 characterizes the data reception behavior, at the receiver end, of the Stop-and-Wait protocol and the corresponding predicate is defined as follows

Definition 5.2: *Data Reception Behavior for Stop-and-Wait Protocol*

```
⊢ ∀ sn dataR sink r.
  DATA_RECV_STOP_WAIT sn dataR sink r =
  ∀ t.
    (if good_packet (dataR t) ∧ (label (dataR t) = r t) then
      (sink (t + 1) = sink t ++ [data (dataR t)]) ∧
      (r (t + 1) = mod_n_add (r t) 1 sn)
    (else
      (sink (t + 1) = sink t) ∧ (r (t + 1) = r t))
```

where the history dataR represents the data messages received by the receiver at any particular time. The history r represents, modulo sn, the sequence number of the data message that the receiver is expecting to receive. The function data returns the data portion of a packet and ++ is the symbol for the list *append* function in HOL.

The definition of DATA_RECV_STOP_WAIT should be read as follows. At all times t, if (dataR t) is not a non_packet, i.e., (good_packet (dataR t)), and the sequence field of the packet (dataR t) is equal to the next number to be output to the sink, i.e., (label (dataR t) = r t), then the data part of the packet is appended to the sink list and r is updated to the sequence number of the next message expected, i.e., (r (t + 1) = mod_n_add (r t) 1 sn). Otherwise if a valid data packet is not received then the output list sink and r retain their old values.

We have assigned a fixed value of 1 to the processing delay (t_p) in order to simplify the understandability of the proofs presented in the next two sections. If required, the processing delay can be made a variable quantity by using a similar approach that we used for t_{out} and t_f delays in the predicate DATA_TRANS_STOP_WAIT.

5.3.4 ACK Transmission

The process ACK_TRANS in Figure 5.2 characterizes the ACK transmission behavior of the Stop-and-Wait protocol and the corresponding predicate is defined as follows

Definition 5.3: *ACK Transmission Behavior for Stop-and-Wait Protocol*

```
⊢ ∀ sn ackR r ackty ack_msg ta dta rec_flag.
 ACK_TRANS_STOP_WAIT sn ackR r ackty ack_msg ta dta rec_flag =
 ∀ t.  (ackty t = ack_msg) ∧
   (if ¬(r t = r (t - 1)) then
     (if dta t = 0 then
      ackR t = new_packet (mod_n_sub (r t) 1 sn) (ackty t) ∧
      (dta (t + 1) = ta) ∧ (rec_flag (t + 1) = F)
     else
      (ackR t = set_non_packet) ∧ (dta (t + 1) = dta t - 1)∧
       (rec_flag (t + 1) = T))
   else
     (if rec_flag t then
       (if dta t = 0 then
        ackR t = new_packet(mod_n_sub (r t) 1 sn)(ackty t)∧
         (dta (t + 1) = ta) ∧ (rec_flag (t + 1) = F)
       else
         (ackR t = set_non_packet) ∧
         (dta (t + 1) = dta t - 1) ∧ (rec_flag (t + 1) = T))
     else
       (ackR t = set_non_packet) ∧ (dta (t + 1) = ta) ∧
       (rec_flag (t + 1) = F)))
```

where the history `ackR` represents the ACK messages transmitted by the sender at any particular time. The history `ackty` represents the data part of the ACK message that could be used to specify properties of protocols, such as negative acknowledgements: a type of acknowledgement message which enables the sender to retransmit messages efficiently. The variable `ack_msg` represents a constant data field that is sent along with every ACK message by the receiving station, as in the Stop-and-Wait protocol the ACK messages do not convey any other information except the reception of a data message. The variable `ta` holds the value for the t_a delay and the history `dta` keeps track of the timer associated with this delay. Whereas, the history `rec_flag` keeps track of the reception of a data message at the receiver until a corresponding ACK message is sent.

The definition of `ACK_TRANS_STOP_WAIT` should be read as follows. At all times `t`, the history `ackty` is assigned the value of the default ACK message for the Stop-and-Wait protocol, i.e., `ack_msg`. For all times `t`, if an in-sequence data message arrives at the receiver $\neg$`(r t = r (t - 1))`, then instantly transmit an ACK message if the contents of the timer `dta` are 0, otherwise do not issue an ACK and retain the information of receiving a valid data in the `rec_flag` while activating the timer associated with t_a by decrementing its value by 1. On the other hand, for all times `t` for which no in-sequence data message arrives at the receiver, check if there exists a valid data message that has successfully arrived at the receiver but has not been acknowledged so far `(rec_flag t)`. If that is the case, then if the timer associated with the delay t_a has expired `(dta t = 0)` then instantly issue the respective ACK message while initializing histories `dta` and `rec_flag` to their default values of `ta` and *False*, respectively. Otherwise wait for the `dta` timer to expire while holding the ACK transmission and the value of history `rec_flag` and decrementing the value of

the timer dta by 1. On the other hand, if there is no valid data arrival or no pending ACK transmission, then the receiver is not allowed to transmit an ACK message and it assigns the histories dta and rec_flag to their default values of ta and *False*, respectively.

5.3.5 ACK Reception

The process ACK_RECV in Figure 5.2 characterizes the ACK reception behavior, at the sending station, of the Stop-and-Wait protocol and the corresponding predicate is defined as follows

Definition 5.4: *ACK Reception Behavior for Stop-and-Wait Protocol*

```
⊢ ∀ ws sn ackS rem s.
  ACK_RECV_STOP_WAIT ws sn ackS rem s =
  ∀ t.
    (if
      good_packet (ackS t) ∧
      mod_n_sub (label (ackS t)) (s t) sn < ws
    then
      (s (t + 1) = mod_n_add (label (ackS t)) 1 sn) ∧
      (rem (t + 1) =
        tli (mod_n_sub (s (t + 1)) (s t) sn) (rem t))
    else
      (s (t + 1) = s t) ∧ (rem (t + 1) = rem t))
```

The sender checks the label of every ACK message it receives to find out if it is one of the messages that has been sent and not yet acknowledged, i.e., if the modulo-*sn* difference between the sequence number of (ackS t) and sender's sequence number

is less than ws, i.e., `(mod_n_sub (label (ackS t)) (s t) sn < ws)`. If this is the case, then the sender slides the window up by updating the sender's history `(s t)` to be the first message not known to be accepted: `(mod_n_add (label (ackS t)) 1 sn)` and by updating `(rem t)`, the list of data remaining to be sent. Otherwise, both histories `s` and `rem` retain their previous values. As in the case of the receiver, we again assigned a fixed value of 1 to the processing delay (t_p).

5.3.6 Communication Channel

The processes `DATA_CHAN` and `ACK_CHAN` in Figure 5.2 characterize the communication channel connecting the sender and receiver in the Stop-and-Wait. For the analysis, we require a channel with fixed propagation delay (t_{prop}). We present two definitions for the communication channel for the Stop-and-Wait protocol; the first one models the channel that is noiseless and the second one models a noisy channel, which may loose packets. The noiseless channel predicate is defined as follows

Definition 5.5: *Noiseless Communication Channel*

```
⊢ ∀ in out d tprop.
  NOISELESS_CHANNEL_STOP_WAIT in out d tprop =
  ∀ t.
    (if t < tprop then
      out t = set_non_packet
    else
      out t = in (t - d t)) ∧
    (0 < tprop) ∧ (d t = tprop)
```

where the histories `in`, `out` and `d` represent the input message, output message and the propagation delay for the channel at a particular time, respectively. The variable

tprop represents the fixed value of channel delay (d t) for all t. According to the above definition, the output from a channel at time t is a copy of the channel's input at time (t - tprop).

Next, we define a predicate that models a noisy channel that looses a message with probability p.

Definition 5.6: *Noisy Communication Channel*

```
⊢ ∀ in out d tprop p bseqt.
  NOISY_CHANNEL_STOP_WAIT in out d tprop p bseqt =
  ∀ t.
    (if t < tprop then
      (out t = set_non_packet) ∧ (bseqt (t + 1) = bseqt t)
    else
      (if good_packet (in (t - d t)) then
        (if ¬fst (prob_bern p (bseqt t)) then
          (out t = in (t - d t))∧
          (bseqt (t + 1) = snd (prob_bern p (bseqt t)))
        else
          (out t = set_non_packet) ∧
          (bseqt (t + 1) = snd (prob_bern p (bseqt t))))
      else
        (out t = set_non_packet) ∧
        (bseqt (t + 1) = bseqt t))) ∧
      (0 < tprop) ∧ (d t = tprop)
```

In the above definition, we utilized the formal definition of the Bernoulli(p) random variable to model the noise effect. The variable p represents the probability of channel

error or getting a *True* from the Bernoulli random variable and the history `bseqt` keeps track of the remaining portion of the infinite Boolean sequence after every call of the Bernoulli random variable. According to the above definition, a valid packet, that arrives at input of the channel, appears at the output of the channel after `tprop` time units with probability $1 - p$.

5.3.7 Stop-and-Wait Protocol

We first define some constraints that are required to ensure the correct behavior of our Stop-and-Wait protocol specification, before giving the actual formalization of the protocol.

Initial Conditions

Initial conditions are usually required for ensuring correct behavior of formal models. In case of the formal specification of real-time systems in HOL, we need to assign appropriate values to the history variables as initial conditions. We used the following initial conditions for the Stop-and-Wait protocol.

Definition 5.7: *Initial Conditions for Stop-and-Wait Protocol*

```
⊢ ∀ source rem s sink r i
    ackR dtout dtf dta tout tf ta rec_flag bseqt bseq.
  INIT_STOP_WAIT source rem s sink r i
    ackR dtout dtf dta tout tf ta rec_flag bseqt bseq =
      (rem 0 = source) ∧ (s 0 = 0) ∧ (sink 0 = []) ∧
      (r 0 = 0) ∧ (i 0 = 0) ∧ (dtout 0 = tout) ∧
      (rec_flag 0 = F) ∧ (ackR 0 = set_non_packet) ∧
      (dtf 0 = tf) ∧ (dta 0 = ta) ∧ (bseqt 0 = bseq)
```

Assumptions

Liveness or Timeliness: While verifying a system, which allows nondeterministic or probabilistic choice between actions, we often need to include additional constraints to make sure that events of interest do occur. This has been done by including a *timeliness* constraint in the specification of the Stop-and-Wait protocol: if the sender's state has not changed over an interval of `maxP` time units, then the sender assumes that the receiver or the channel has crashed and aborts the protocol. A predicate `ABORT` is defined that is *True* only when the protocol aborts and *False* otherwise. Now, the predicate `ABORT` characterizes which `abort` histories satisfy this constraint.

Definition 5.8: *Timeliness Assumption for Stop-and-Wait Protocol*

```
⊢ ∀ abort maxP rem.
  ABORT abort maxP rem =
   ∀ t.  abort t =
      (rem t = rem (t - maxP)) ∧ maxP ≤ t ∧ ¬NULL (rem t)
```

A protocol is said to be live if it is never aborted. This kind of liveness is *assumed* using the following constraint

```
LIVE_ASSUMPTION abort = ∀ t.  ¬(abort t)
```

Window Size and Sequence Numbers: As has been mentioned before, instead of using their exact values of 1 and 2, we used variables *ws* and *sn* to represent the *window size* and distinct sequence numbers for the Stop-and-Wait protocol in the above predicates. This has been done, in order to be able to establish logical implication between the predicates defined in this chapter and the corresponding predicates for the Sliding Window protocol, defined in [12]. Now, we assign the exact values to these variables in an assumption predicate as follows

Definition 5.9: *Window Size and Sequence Number for Stop-and-Wait Protocol*

```
⊢ ∀ ws sn.  WSSN_ASSUM_STOP_WAIT ws sn = (ws = 1) ∧ (sn = 2)
```

The Stop-and-Wait protocol can now be formalized as the logical conjunction of the predicates defined above. We present two specifications corresponding to noiseless or ideal and noisy channel conditions.

Definition 5.10: *Stop-and-Wait Protocol - Noiseless Channel*

```
⊢ ∀ 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.
STOP_WAIT_NOISELESS 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 =
INIT_STOP_WAIT source rem s sink r i
  ackR dtout dtf dta tout tf ta rec_flag ∧
DATA_TRANS_STOP_WAIT ws sn dataS s rem i ackS
  tout tf dtout dtf ∧
NOISELESS_CHANNEL_STOP_WAIT dataS dataR d tprop ∧
DATA_RECV_STOP_WAIT sn dataR sink r ∧
ACK_TRANS_STOP_WAIT sn ackR r ackty ack_msg ta dta rec_flag ∧
NOISELESS_CHANNEL_STOP_WAIT ackR ackS d tprop ∧
ACK_RECV_STOP_WAIT ws sn ackS rem s ∧
ABORT abort maxP rem ∧ WSSN_ASSUM_STOP_WAIT ws sn
```

The higher-order-logic predicate STOP_WAIT_NOISELESS formally specifies the behavior of the Stop-and-Wait protocol under ideal or noiseless conditions as the corresponding predicate for the channel has been used for both data and ACK channels. It is also important to note here that we do not initialize the history bseqt in the predicate INIT_STOP_WAIT as there is no need to use the infinite Boolean sequence in this case. Next, we utilize the noisy channel predicate to formally specify the Stop-and-Wait protocol with a noisy channel as follows

Definition 5.11: *Stop-and-Wait Protocol - Noisy Channel*

```
⊢ ∀ 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 =
INIT_STOP_WAIT source rem s sink r i
  ackR dtout dtf dta tout tf ta rec_flag bseqt bseq ∧
DATA_TRANS_STOP_WAIT ws sn dataS s rem i ackS
  tout tf dtout dtf ∧
NOISY_CHANNEL_STOP_WAIT dataS dataR d tprop p bseqt ∧
DATA_RECV_STOP_WAIT sn dataR sink r ∧
ACK_TRANS_STOP_WAIT sn ackR r ackty ack_msg ta dta rec_flag ∧
NOISELESS_CHANNEL_STOP_WAIT ackR ackS d tprop ∧
ACK_RECV_STOP_WAIT ws sn ackS rem s ∧
ABORT abort maxP rem ∧
WSSN_ASSUM_STOP_WAIT ws sn
```

In the above definition, the data channel has been made noisy while a noiseless channel is used for the ACK messages. This has been done on purpose in order to reduce the length of the performance analysis proof by avoiding some redundancy. On the other hand, this decision does not affect the illustration of the idea behind the performance analysis of the Stop-and-Wait protocol under noisy conditions as we present the complete handling of a noisy channel in one direction. The analysis can be extended to both noisy channels by choosing noisy channel predicates for both channels and then handling the ACK channel in a similar way as the noisy data channel is handled in Section 5.5.2 of this chapter.

5.4 Functional Verification in HOL

The job of an ARQ protocol is to ensure reliable transfer of a stream of data from the sender to the receiver. This requirement can be formally specified as follows [12]

Definition 5.12: *Functional Correctness Requirement for ARQ Protocols*

```
⊢ ∀ source sink.
  REQ source sink =
    (∃ t.  sink t = source) ∧
    ∀ t n.  is_prefix (sink t) (sink (t + n))
```

where the predicate `is_prefix` is *True* if its first list argument is a prefix of its second list argument. According to the predicate `REQ`, an ARQ protocol satisfies its functional requirements only if there exists a time at which the `sink` list becomes equal to the original `source` list, i.e., a time when all the data at the sender is transferred, as is, to the receiver, and the history `sink` is prefix closed.

A generic formal specification of a Sliding Window protocol, which covers all the ARQ protocol variants, has been presented in [12]. The specification is based on the model illustrated in Figure 5.2 and has been shown to satisfy the functional correctness requirement given in the REQ predicate. In order to verify the functional correctness of our specification of the Stop-and-Wait protocol, we benefit from this work instead of conducting the verification from scratch. For this purpose, we defined the predicates for the Stop-and-Wait protocol in such a way that they logically imply the corresponding predicates used for the formal specification of the Sliding Window protocol presented in [12]. This relationship allows us to inherit the functional correctness theorem verified for the specification of the Sliding Window protocol for our Stop-and-Wait protocol specification.

For illustration purposes, consider the example of the data transmission predicate. It has been defined in [12] for the Sliding Window protocol as follows

Definition 5.13: *Data Transmission Behavior for Sliding Window Protocol*

```
⊢ ∀ ws sn dataS s rem i.
  DATA_TRANS_SW ws sn dataS s rem i =
  ∀ t.
    (if ¬NULL (tli (i t) (rem t)) ∧ i t < ws then
      (dataS t = new_packet
          (mod_n_add (s t) (i t) sn) (hdi (i t) (rem t))) ∨
      (dataS t = set_non_packet)
    else
      dataS t = set_non_packet)
```

It can be easily verified in HOL, using Boolean algebra properties, that the predicate DATA_TRANS_STOP_WAIT, given in Section 5.3.2, logically implies the above predicate

```
⊢ ∀ ws ns dataS s rem i ackS tout tf dtout dtf.
  DATA_TRANS_STOP_WAIT ws ns dataS s rem i ackS
    tout tf dtout dtf ⇒ DATA_TRANS_SW ws ns dataS s rem i
```

In a similar way, we were able to prove logical implications between all the predicates used in the formal specification of the Sliding Window protocol and the corresponding predicates used for the formal specification of the Stop-and-Wait protocol. These relationships allowed us to formally verify the functional correctness of both of the formal specifications of the Stop-and-Wait protocol in HOL.

Theorem 5.1: *Functional Correctness of STOP_WAIT_NOISELESS*

```
⊢ ∀ 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.
STOP_WAIT_NOISELESS 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 ∧
LIVE_ASSUMPTION abort ⇒ REQ source sink
```

Theorem 5.2: *Functional Correctness of 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 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 ⇒ REQ source sink
```

It is important to note that the generic specification of the Sliding Window Protocol in [12] is quite general and does not include many details, such as the precise

conditions under which the messages are transmitted or acknowledged and the delays (t_{out}, t_f, t_a, etc.) associated with different operations. Therefore, it cannot be used for reasoning about message delays and thus performance related properties, as such. On the other hand, our formal specification of the Stop-and-Wait protocol is more specific and provides a detailed description of the protocol including the timing behavior associated with different operations.

Another major point that we would like to mention here is that in order to establish the logical implication between the two protocol models, we had to introduce some additional generality in our formal definitions, such as the usage of variables *ws* and *sn* instead of their exact values of 1 and 2 , respectively. Even though, such generalizations are not required for the functional description of the Stop-and-Wait protocol, they do not harm us in any way. They lead to a much faster functional verification, as has been illustrated in this section. On the other hand, they do not affect the formal reasoning related to the performance issues, since the exact values for these variables are assigned in an assumption (`WSSN_ASSUM_STOP_WAIT`) that is a part of our Stop-and-Wait protocol specification.

5.5 Performance Analysis in HOL

Now, we present the verification of the message delay relations for the Stop-and-Wait protocol, given in Equations (5.1) and (5.3), for noiseless and noisy channels, respectively. The verification is based on the two formal specifications of the Stop-and-Wait protocol, `STOP_WAIT_NOISELESS` and `STOP_WAIT_NOISY`, given in the previous section.

5.5.1 Noiseless Channel Conditions

The first and the foremost step in verifying the message delay characteristic for the Stop-and-Wait protocol is to formally specify it. Informally speaking, the message delay refers to the time required for the transmitter to send a single data message and know that it has been successfully received at the receiver. We specify this in higher-order logic as follows

Definition 5.14: *Stop-and-Wait Protocol Delay - Noiseless Channel*

```
⊢ ∀ rem source.  DELAY_STOP_WAIT_NOISELESS rem source =
    @t.  (rem t = TL source) ∧ (rem (t - 1) = source)
```

The above specification returns the time `t` at which the `rem` list is reduced by one element from its initially assigned value of the `source` list. It is indeed precisely equal to the message delay of the first data element in the `source` list.

Based on the above definition of the message delay and the delays associated with the formal specification of the Stop-and-Wait protocol (`STOP_WAIT_NOISELESS`), Equation (5.1) can be formally expressed in HOL as follows

Theorem 5.3: *Stop-and-Wait Protocol Delay Relation - Noiseless Channel*

```
⊢ ∀ 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.
  STOP_WAIT_NOISLESS 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 ∧ ¬(NULL source) ∧
  tprop + 1 + ta + tprop + 1 ≤ tout
    ⇒ (DELAY_STOP_WAIT_NOISELESS rem source =
      tf + tprop + 1 + ta + tprop + 1)
```

It is important to note here that the processing delay, t_p, has been assigned a value of 1 in our model, as explained in the previous section. The two assumptions that we have added to Theorem 5.3 ensure that the source list is not an empty list, i.e., ¬(NULL source), otherwise no data transfer takes place, and the time out period tout is always greater than or equal to its lower bound specified in Section 5.2.

Rewriting the proof goal of Theorem 5.3 with the definition of the Stop-and-Wait protocol delay and removing the Hilbert Choice operator we get the following expression

$$\begin{aligned}
&(\exists\, \mathtt{x}.(\mathtt{rem\ x\ =\ TL\ source})\ \wedge\ (\mathtt{rem\ (x\ -\ 1)\ =\ source}))\ \wedge \\
&(\forall\, \mathtt{x}.\ (\mathtt{rem\ x\ =\ TL\ source})\ \wedge\ (\mathtt{rem\ (x\ -\ 1)\ =\ source}) \\
&\qquad \Rightarrow (\mathtt{x\ =\ tf\ +\ tprop\ +\ 1\ +\ ta\ +\ tprop\ +\ 1}))
\end{aligned} \tag{5.4}$$

The above subgoal is a logical conjunction of two Boolean expressions and it can be proved to be *True* only if there exists a time x for which the conditions (rem x = TL source) and (rem (x - 1) = source) are *True* and the value of any variable x that satisfies these conditions is unique and is equal to tf + tprop + 1 + ta + tprop + 1.

We proceed with the proof of the subgoal, given in Equation (5.4), by assuming the following expression

$$\begin{aligned}
&(\mathtt{rem\ (tf\ +\ tprop\ +\ 1\ +\ ta\ +\ tprop\ +\ 1)\ =\ TL\ source})\ \wedge \\
&(\mathtt{rem\ ((tf\ +\ tprop\ +\ 1\ +\ ta\ +\ tprop\ +\ 1)\ -\ 1)\ =\ source})
\end{aligned} \tag{5.5}$$

to be *True*, which we will prove later, under the given constraints for the Stop-and-Wait protocol. Equation (5.5) leads us to prove the first Boolean expression in our subgoal as now we know an x = (tf + tprop + 1 + ta + tprop + 1) for which the

given conditions are *True*. We verified the second Boolean expression in the subgoal by first proving the monotonically decreasing characteristic of the history `rem` under the given constraints of the Stop-and-Wait protocol, i.e.,

$$\forall\, \mathtt{a\ b.\ a} < \mathtt{b.} \Rightarrow \exists\, \mathtt{c.\ c} + + \mathtt{rem\ b} = \mathtt{rem\ a} \tag{5.6}$$

where ++ represents the list *append* function in HOL. Now, if there exists an `x`, that satisfies the conditions `(rem x = TL source)` and `(rem (x - 1) = source)`, then it may be equal to, less than or greater than `(tf + tprop + 1 + ta + tprop + 1)`. For the later two cases, we reach a contradiction in the assumption list, based on the monotonically decreasing characteristic of the history `rem`, whereas, the case when `x = (tf + tprop + 1 + ta + tprop + 1)` verifies our subgoal of interest, which concludes the proof of Theorem 5.3 under the assumption of Equation (5.5).

Equation (5.5) can now be proved in HOL using the definitions of the predicates in the formal specification of the Stop-and-Wait protocol under noiseless channels. The corresponding HOL proof step sequence is summarized in Table 5.1.

5.5.2 Noisy Channel Conditions

Just like the case of the analysis under noiseless conditions, the message delay, under noisy channel conditions, refers to the time required for the transmitter to send a single data message and know that it has been successfully received at the receiver. Though the delay, in this case, is a random quantity since its value is non-deterministic and depends on the outcomes of a sequence of Bernoulli trials, which are used to model the channel noise as can be seen in the definition of the predicate `NOISY_CHANNEL_STOP_WAIT`. Therefore, the message delay for the Stop-and-Wait protocol under noisy channel needs to be formally specified as a random variable

Number	Formally Verified Statements
1	$\forall t.t \leq tf \Rightarrow (i(t) = 0)$
2	$\forall t.t < tf \Rightarrow (dataS\ t = set\_non\_packet)$
3	$\forall t.t < tf + tprop \Rightarrow (dataR\ t = set\_non\_packet)$
4	$\forall t.t < tf + tprop + 1 \Rightarrow (sink\ t = []) \wedge (r\ t = 0)$
5	$\forall t.t \leq tf + tprop + 1 \Rightarrow (rec\_flag\ t = F) \wedge (dta\ t = ta)$
6	$\forall t.t < tf + tprop + 1 + ta \Rightarrow (ackR\ t = set\_non\_packet)$
7	$\forall t.t < tf + tprop + 1 + ta + tprop \Rightarrow (ackS\ t = set\_non\_packet)$
8	$\forall t.t < tf + tprop + 1 + ta + tprop + 1$ $\Rightarrow (s\ t = 0) \wedge (rem\ t = source)$
9	$(dataS\ tf = new\_packet\ 0\ (HD\ source)) \wedge (i(tf + 1) = 1)$
10	$\forall t.tf < t \wedge t < tf + tprop + 1 + ta + tprop + 1$ $\Rightarrow (tout + tf - t \leq dtout\ t)$
11	$\forall t.tf < t \wedge t < tf + tprop + 1 + ta + tprop + 1$ $\Rightarrow (i\ t = 1) \wedge (dataS\ t = set\_non\_packet)$
12	$dataR(tf + tprop) = new\_packet\ 0\ (HD\ source)$
13	$\forall t.tf + tprop < t \wedge t < tf + tprop + 1 + ta + tprop + 1$ $\Rightarrow (dataR\ t = set\_non\_packet)$
14	$\forall t.tf + tprop + 1 \leq t \wedge t < tf + tprop + 1 + ta + tprop + 1$ $\Rightarrow (r\ t = 1)$
15	$\forall t.tf + tprop + 1 \leq t \wedge t < tf + tprop + 1 + ta$ $\Rightarrow (rec\_flag(t + 1)) \wedge (dta(t + 1) = ta - (t - (tf + tprop)))$
16	$ackR(tf + tprop + 1 + ta) = new\_packet\ 0\ ack\_msg$
17	$rem(tf + tprop + 1 + ta + tprop + 1) = TL\ source$

Table 5.1: HOL Proof Sequence for Equation (5.5)

Definition 5.15: *Stop-and-Wait Protocol Delay - Noisy Channel*

```
⊢ ∀ rem source bseqt.
  DELAY_STOP_WAIT_NOISY rem source bseqt =
    ((@t.  (rem t = TL source) ∧ (rem (t - 1) = source)),
      bseqt @t.
        (rem t = TL source) ∧ (rem (t - 1) = source))
```

where history `bseqt t` represents the unused portion of the infinite Boolean sequence after performing the required number of Bernoulli trials at any given time `t`. The above specification returns a pair with the first element equal to the time `t` that satisfies the two conditions `(rem t = TL source)` and `(rem (t - 1) = source)`, and thus represents the random message delay of the first data element in the `source` list, and the second element is equal to the unused portion of the infinite Boolean sequence at this time instant `t`.

As a first step towards the verification of the average value of the random delay specified in `DELAY_STOP_WAIT_NOISY`, we establish its relationship with the infamous Geometric random variable, which basically returns the number of trials to attain the first success in an infinite sequence of Bernoulli trials [19]. This way, we can benefit from the existing HOL theorems related to the average characteristic of Geometric(p) random variable, such as Theorem 2.14, for the verification of the average value of the message delay of a Stop-and-Wait protocol. This relationship, given in Equation (5.2), can be expressed in HOL using the formal specification of the Stop-and-Wait protocol `STOP_WAIT_NOISY` and the Geometric random variable `prob_geom` as follows

Theorem 5.4: *Stop-and-Wait Protocol Delay in terms of Geometric Random Variable*

```
⊢ ∀ 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 ∧ ¬(NULL source) ∧
tprop + 1 + ta + tprop + 1 ≤ tout ∧
LIVE_ASSUMPTION abort ∧ 0 ≤ p ∧ p < 1
  ⇒ (DELAY_STOP_WAIT_NOISY rem source bseqt =
    ((tf + tout) (fst (prob_geom (1 - p) bseq) - 1) + tf +
    tprop + 1 + ta + tprop + 1,
    snd (prob_geom (1 - p) bseq)))
```

where p represents the probability of channel error, i.e., getting a *True* from the Bernoulli random variable. The first argument of the function `prob_geom` represents the probability of success for the corresponding sequence of the Bernoulli trials, which in the case of our definition of the noisy channel, is equal to the probability of getting a *False* from a Bernoulli trial. The above theorem is proved under the assumption that the value of the probability p always falls in the interval $[0, 1)$. It is not allowed to attain the value 1, in order to avoid the case when the channel always rejects incoming packets and thus leads to no data transfers. The assumption, `LIVE_ASSUMPTION abort` ensures liveness as has been explained in Section 5.3. The other assumptions used in the above theorem are similar to the ones used in Theorem 5.3.

We proceed with the verification of Theorem 5.4 in HOL by first defining the following two recursive functions. The first function, returns *True* if and only if its first argument, say n, represents the positive integer index of a trial, in a sequence of independent Bernoulli trials, that returns a *False* while all Bernoulli trials with lower index values than n have returned a *True*.

Definition 5.16: *N^{th} Bernoulli Trial returns the first False*

```
⊢ ∀ p bseq.
  BERNOULLI_TRIAL_F_IND 0 p bseq = ¬fst (prob_bern p bseq)) ∧
∀ n p bseq.  BERNOULLI_TRIAL_F_IND (suc n) p bseq =
    fst (prob_bern p bseq) ∧
      BERNOULLI_TRIAL_F_IND n p (snd (prob_bern p bseq))
```

The second function returns the value of the **snd** element of the n^{th} Bernoulli trial in a sequence of independent Bernoulli trials, where n represents its first argument. In other words, it basically returns the unused infinite Boolean sequence after n independent Bernoulli trials have been performed using the given infinite Boolean sequence.

Definition 5.17: *Second Component of the N^{th} Bernoulli Trial*

```
⊢ ∀ p bseq.  NTH_BERNOULLI_TRIAL_SND 0 p bseq = bseq) ∧
∀ n p bseq.  NTH_BERNOULLI_TRIAL_SND (suc n) p bseq =
    snd (prob_bern p (NTH_BERNOULLI_TRIAL_SND n p bseq))
```

Under the assumptions of Theorem 5.4, it can be shown that a data message available at the `source` list does finally make through the noisy channel at some time. This can be verified in HOL, for the top element of the `source` list, by proving that there exists some n for which the function `BERNOULLI_TRIAL_F_IND` returns a *True*

$$\exists\ \texttt{n.}\ \texttt{BERNOULLI\_TRIAL\_F\_IND n p bseq} \tag{5.7}$$

for the given values of p and bseq. If a positive integer n exists that satisfies the above condition, then it can be verified in HOL that the Geometric random variable, which returns the number of trials to attain the first success in an independent sequence of Bernoulli(p) trials, with success probability equal to (1 - p) can be formally expressed as follows

$$\begin{aligned}&\forall\ \texttt{n p s.}\ 0 \leq \texttt{p} \wedge \texttt{p} < 1 \wedge \texttt{BERNOULLI\_TRIAL\_F\_IND n p s}\\ &\quad \Rightarrow (\texttt{prob\_geom}\ (1-\texttt{p})\ \texttt{s} = (\texttt{n} + 1,\ \texttt{NTH\_BERNOULLI\_TRIAL\_SND}\ (\texttt{n}+1)\ \texttt{p s}))\end{aligned} \tag{5.8}$$

The HOL proof is based on the formal definition of the function prob_geom and the underlying probability theory principles, presented in [36].

Based on the above results, the proof goal of Theorem 5.4 can be simplified using the definition of DELAY_STOP_WAIT_NOISY and removing the Hilbert Choice operator as follows

$$\begin{aligned}&(\exists\ \texttt{x.}(\texttt{rem x} = \texttt{TL source}) \wedge (\texttt{rem}\ (\texttt{x} - 1) = \texttt{source})) \wedge\\ &(\forall\ \texttt{x.}\ (\texttt{rem x} = \texttt{TL source}) \wedge (\texttt{rem}\ (\texttt{x} - 1) = \texttt{source})\\ &\quad \Rightarrow (\texttt{x} = (\texttt{tf} + \texttt{tout})\texttt{n} + \texttt{tf} + \texttt{tprop} + 1 + \texttt{ta} + \texttt{tprop} + 1)) \wedge\\ &\qquad (\texttt{bseqt x} = \texttt{NTH\_BERNOULLI\_TRIAL\_SND}\ (\texttt{n} + 1)\ \texttt{p bseq})\end{aligned} \tag{5.9}$$

The above subgoal is quite similar to the one that we got after simplifying the proof goal of Theorem 5.3. Therefore, we follow the same proof approach and assume the following expression

```
rem ((tf + tout)n + tf + tprop + 1 + ta + tprop + 1 − 1) = source ∧
 rem ((tf + tout)n + tf + tprop + 1 + ta + tprop + 1) = TL source ∧
(bseqt ((tf + tout)n + tf + tprop + 1 + ta + tprop + 1) =
        NTH_BERNOULLI_TRIAL_SND (n + 1) p bseq)
```

(5.10)

to be *True*, which we will prove later, under the given assumptions of Theorem 5.4. Equation (5.10) leads us to prove the first Boolean expression in the subgoal as now we know an `x = ((tf + tout) n + tf + tprop + 1 + ta + tprop + 1)` for which the given conditions `(rem x = TL source)` and `(rem (x - 1) = source)` are *True*. The second Boolean expression in the subgoal can now be proved using Equation (5.10) along with the monotonically decreasing characteristic of the history `rem` in a similar way as we handled the counterpart while verifying Theorem 5.3.

The next step is to prove Equation (5.10) under the assumptions given in the assumption list of Theorem 5.4. We proceed in this direction by verifying

Lemma 5.1: *Stop-and-Wait Protocol Delay with Generic Initialization*

```
⊢ ∀ bseq v.  INIT_STOP_WAIT_GEN source rem s sink r i
   ackR dtout dtf dta tout tf ta rec_flag bseqt bseq v ∧
BERNOULLI_TRIAL_F_IND n p bseq
⇒ (rem (v + (tf + tout) n +
     tf + tprop + 1 + ta + tprop + 1 - 1) = source) ∧
   (rem (v + (tf + tout) n +
     tf + tprop + 1 + ta + tprop + 1) = TL source) ∧
   (bseqt (v + (tf + tout) n + tf + tprop + 1 + ta + tprop + 1)
     = NTH_BERNOULLI_TRIAL_SND (n + 1) p bseq)
```

under the assumptions of Theorem 5.4. Equation (5.10) is a special case of Lemma 5.1 when $v = 0$. The first assumption in Lemma 5.1, i.e., `INIT_STOP_WAIT_GEN`, provides the status of the histories used in the predicate `STOP_WAIT_NOISY` at time `v` as follows

Definition 5.18: *Generic Initialization Constraint*

```
⊢ ∀ source rem s sink r i ackR dtout dtf dta tout tf ta
    rec_flag bseqt bseq v.
  INIT_STOP_WAIT_GEN source rem s sink r i ackR dtout dtf dta
    tout tf ta rec_flag bseqt bseq v =
  (i v = 0) ∧ (dtout v = tout) ∧ (dtf v = tf) ∧
  (dta v = ta) ∧ (bseqt v = bseq) ∧
  (∀ t.  t ≤ v ⇒ (rem t = source) ∧
    (s t = 0) ∧ (sink t = []) ∧ (r t = 0) ∧
    (rec_flag t = F) ∧ (ackR t = set_non_packet)) ∧
  ∀ t.  v - (tout - 1) ≤ t ∧ t < v ⇒ (i t = 1)
```

It can be proved to be a logical implication of the predicate `INIT_STOP_WAIT`, which is included in the definition of `STOP_WAIT_NOISY` and is thus present in the assumption list of Theorem 5.4, for the case when `v = 0`. Whereas, the second assumption in the assumption list of Lemma 5.1, i.e., `BERNOULLI_TRIAL_F_IND n p bseq`, has already been shown to be a consequence of the assumptions of Theorem 5.4. Thus, Equation (5.10) can be proved as a special case of Lemma 5.1 when the positive integer variable `v` is assigned a value of 0.

Now, in order to complete the formal proof of Theorem 5.4 in HOL, we need to verify Lemma 5.1. We proceed with this proof by applying induction on the positive integer variable n. For the base case, i.e., `n = 0`, we get the following subgoal after some basic arithmetic simplification and using the function definitions of

BERNOULLI_TRIAL_F_IND and NTH_BERNOULLI_TRIAL_SND.

```
INIT_STOP_WAIT_GEN source rem s sink r i
        ackR dtout dtf dta tout tf ta rec_flag bseqt bseq v ∧
  ¬(fst (prob_bern p bseq))
  ⇒ (rem (v + tf + tprop + 1 + ta + tprop + 1 - 1) = source) ∧
      (rem (v + tf + tprop + 1 + ta + tprop + 1) = TL source) ∧
  bseqt (v + tf + tprop + 1 + ta + tprop + 1) = snd(prob_bern p bseq)
```

(5.11)

The assumption ¬fst (prob_bern p bseq) ensures that the noisy data channel allows reliable transmission of the first data message in the first trial. Thus, the base case of Lemma 5.1 becomes similar to the case of a noiseless data channel, as far as the transmission of the first data element of the source list is concerned. Therefore, its proof can be handled in a similar way as the proof of Equation (5.5) as the only difference between the two is the fact that now the initial conditions are defined for an arbitrary positive integer v instead of 0. The corresponding HOL proof step sequence is summarized in Table 5.2. These proofs are based on the definitions of INIT_STOP_WAIT_GEN and the predicates corresponding to the six processes, given in Figure 5.2, for the Stop-and-Wait protocol under a noisy data channel.

In the step case for Lemma 5.1, we get the following subgoal after simplifying with the definitions of BERNOULLI_TRIAL_F_IND and NTH_BERNOULLI_TRIAL_SND

Number	Formally Verified Statements
1	$\forall t.v \leq t \wedge t \leq v + tf \Rightarrow (i(t) = 0)$
2	$\forall t.v - (tout - 1) \leq t \wedge t < v + tf \Rightarrow (dataS\ t = set\_non\_packet)$
3	$\forall t.v \leq t \wedge t < v + tf + tprop \Rightarrow (dataR\ t = set\_non\_packet)$
4	$\forall t.v \leq t \wedge t \leq v + tf + tprop \Rightarrow (bseqt\ t = bs)$
5	$\forall t.v \leq t \wedge t < v + tf + tprop + 1 \Rightarrow (sink\ t = []) \wedge (r\ t = 0)$
6	$\forall t.v \leq t \wedge t \leq v + tf + tprop + 1 \Rightarrow (rec\_flag\ t = F) \wedge (dta\ t = ta)$
7	$\forall t.t < v + tf + tprop + 1 + ta \Rightarrow (ackR\ t = set\_non\_packet)$
8	$\forall t.v \leq t \wedge t < v + tf + tprop + 1 + ta + tprop$ $\Rightarrow (ackS\ t = set\_non\_packet)$
9	$\forall t.v \leq t \wedge t < v + tf + tprop + 1 + ta + tprop + 1$ $\Rightarrow (s\ t = 0) \wedge (rem\ t = source)$
10	$(i(v + tf + 1) = 1) \wedge (dataS(v + tf) = new\_packet\ 0\ (HD\ source))$
11	$\forall t.v \leq t \wedge t \leq v + tf \Rightarrow (dtout\ t = tout)$
12	$\forall t.v + tf < t \wedge t < v + tf + tprop + 1 + ta + tprop + 1$ $\Rightarrow v + tf + tout - t \leq dtout\ t$
13	$\forall t.v + tf + 1 \leq t \wedge t \leq v + tf + tprop + 1 + ta + tprop$ $\Rightarrow (i\ t = 1) \wedge (dataS\ t = set\_non\_packet)$
14	$dataR(v + tf + tprop) = new\_packet\ 0\ (HD\ source)$
15	$\forall t.v + tf + tprop < t \wedge t < v + tf + tprop + 1 + ta + tprop$ $\Rightarrow (dataR\ t = set\_non\_packet)$
16	$(r(v + tf + tprop + 1) = 1) \wedge (dta(v + tf + tprop + 1) = ta)$
17	$\forall t.v + tf + tprop + 1 < t \wedge t < v + tf + tprop + 1 + ta + tprop$ $\Rightarrow (r\ t = 1)$
18	$\forall t.v + tf + tprop + 1 \leq t \wedge t < v + tf + tprop + 1 + ta$ $\Rightarrow (rec\_flag(t + 1)) \wedge (dta(t + 1) = v + ta - (t - (tf + tprop)))$
19	$ackR(v + tf + tprop + 1 + ta) = new\_packet\ 0\ ack\_msg$
20	$ackS(v + tf + tprop + 1 + ta + tprop) = new\_packet0ack\_msg$
21	$rem(v + tf + tprop + 1 + ta + tprop + 1) = TL\ source$
22	$\forall t.v + tf + tprop < t \wedge$ $t < v + tf + tprop + 1 + ta + tprop + 1 + tprop$ $\Rightarrow (bseqt\ t = snd(prob\_bern\ p\ bseq))$

Table 5.2: HOL Proof Sequence for the Base Case of Lemma 5.1

$$\texttt{INIT\_STOP\_WAIT\_GEN source rem s sink r i}$$

$$\texttt{ackR dtout dtf dta tout tf ta rec\_flag bseqt bseq v} \wedge$$

$$(\texttt{fst (prob\_bern p bseq))} \wedge \texttt{NTH\_BERNOULLI\_TRIAL\_F n p (snd (prob\_bern p bseq))}$$

$$\Rightarrow (\texttt{rem}(\texttt{v} + (\texttt{tf} + \texttt{tout})(\texttt{n} + 1) + \texttt{tf} + \texttt{tprop} + 1 + \texttt{ta} + \texttt{tprop} + 1 - 1) = \texttt{source}) \wedge$$

$$(\texttt{rem}(\texttt{v} + (\texttt{tf} + \texttt{tout})(\texttt{n} + 1) + \texttt{tf} + \texttt{tprop} + 1 + \texttt{ta} + \texttt{tprop} + 1) = \texttt{TL source}) \wedge$$

$$(\texttt{bseqt } (\texttt{v} + (\texttt{tf} + \texttt{tout})(\texttt{n} + 1) + \texttt{tf} + \texttt{tprop} + 1 + \texttt{ta} + \texttt{tprop} + 1) =$$

$$(\texttt{NTH\_BERNOULLI\_TRIAL\_SND } (\texttt{n} + 1)\ \texttt{p } (\texttt{snd } (\texttt{prob\_bern p bseq}))))$$

(5.12)

which needs to be proved under the assumption list of Theorem 5.4 along with the statement of Lemma 5.1. The above subgoal can be proved by specializing Lemma 5.1 for the case when `bseq` and `v` are equal to `snd (prob_bern p bseq)` and `(v + tf + tout)`, respectively, if the given initial conditions in the predicate `INIT_STOP_WAIT_GEN` hold for `snd (prob_bern p bseq)` and `(v + tf + tout)`, i.e.,

$$\texttt{INIT\_STOP\_WAIT\_GEN source rem s sink r i ackR dtout dtf dta tout tf ta}$$

$$\texttt{rec\_flag bseqt (snd (prob\_bern p bseq)) (v + tf + tout)}$$

(5.13)

under the assumptions of Theorem 5.4 and the step case of Lemma 5.1. In order to prove Equation (5.13) we need to formally verify the behavior of the histories, used in the predicate `INIT_STOP_WAIT_GEN`, at various points in the interval `[0, v + tf + tout]`. Therefore, we again use the same approach that we used to prove Equation (5.5) and the base case of Lemma 5.1, i.e., to verify the value of these histories using the initial conditions and the definitions of the predicates used for the formal specification

of the Stop-and-Wait protocol. In fact, the first 11 proof lines, given in Table 5.2, for the base case of Lemma 5.1 can be used, as is, for the proof of Equation (5.13) as well, since a message transmission cannot complete before $v + t_f + t_{prop} + 1 + t_a + t_{prop} + 1$ time units are lapsed and the first data message is issued at time `v + tf` in both cases. Thereafter, contrary to the base case of Lemma 5.1, where one of the assumptions assured the reliable transmission of the first data message, in the case of Equation (5.13) we have the assumption `fst (prob_bern p bseq)` that forces the channel to loose the first data message. Thus, the sender keeps on waiting for a valid ACK until the timer associated with the t_{out} delay expires and this is how the initial state at time v is maintained until the time $v + t_f + t_{out}$. We were able to verify this result, and thus Equation (5.13), using the first 11 proof lines, given in Table 5.2, followed by the proof sequence given in Table 5.3. This concludes the proof of Lemma 5.1, which in turn leads to the proof of Theorem 5.4 as well.

Now, we can express the average message delay relation in HOL as follows

Theorem 5.5: *Average Stop-and-Wait Protocol Delay*

```
⊢ ∀ 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 ∧ ¬(NULL source) ∧
  tprop + 1 + ta ≠ tprop + 1 ≤ tout ∧
  LIVE_ASSUMPTION abort ∧ 0 ≤ p ∧ p < 1
    ⇒ expec (DELAY_STOP_WAIT_NOISY rem source bseqt) =
      (tf + tout) p/(1−p) + (tf + tprop + 1 + ta + tprop + 1)
```

Number	Formally Verified Statements
1	$\forall t.v + tf < t \wedge t < v + tf + tout \Rightarrow v + tf + tout - t \leq dtout\ t$
2	$\forall t.v + tf < t \wedge t < v + tf + tout \Rightarrow (i\ t = 1)$
3	$\forall t.v + tf < t \wedge t < v + tf + tout \Rightarrow (dataS\ t = set\_non\_packet)$
4	$\forall t.v \leq t \wedge t < v + tf + tout + tprop$ $\Rightarrow (dataR\ t = set\_non\_packet)$
5	$\forall t.t < v + tf + tout + tprop + 1 \Rightarrow (sink\ t = []) \wedge (r\ t = 0)$
6	$\forall t.v \leq t \wedge t < v + tf + tout + tprop + 1$ $\Rightarrow (rec\_flag\ t = F) \wedge (dta\ t = ta)$
7	$\forall t.t < v + tf + tout + tprop + 1 \Rightarrow (rec\_flag\ t = F)$
8	$\forall t.t < v + tf + tout + tprop + 1 \Rightarrow (ackR\ t = set\_non\_packet)$
9	$\forall t.v \leq t \wedge t < v + tf + tout + tprop + 1$ $\Rightarrow (ackS\ t = set\_non\_packet)$
10	$\forall t.t < v + tf + tout + tprop + 1 \Rightarrow (s\ t = 0) \wedge (rem\ t = source)$
11	$\forall t.v + tf < t \wedge t \leq v + tf + tout \Rightarrow (dtf\ t = tf)$
12	$\forall t.v + tf + tprop < t \wedge t < v + tf + tout + tprop \Rightarrow$ $(bseqt\ t = snd(prob\_bern\ p\ bseq))$
13	$\forall t.v + tf < t \wedge t < v + tf + tout$ $\Rightarrow (dtout\ t = v + tf + tout - t)$
14	$dtout(v + tf + tout) = tout$
15	$i(v + tf + tout) = 0$

Table 5.3: HOL Proof Sequence for Equation (5.13)

The above proof goal can be reduced to the following subgoal using Theorems 2.4 and 5.4 and some arithmetic simplification

$$\begin{aligned} &\forall\, \mathtt{p}.\ 0 < \mathtt{p} \wedge \mathtt{p} \leq 1 \\ &\Rightarrow\ \mathtt{expec}\ (\lambda \mathtt{s}.(\mathtt{fst}\ (\mathtt{prob\_geom}\ \mathtt{p}\ \mathtt{s}) - 1,\ \mathtt{snd}(\mathtt{prob\_geom}\ \mathtt{p}\ \mathtt{s}))) = \frac{1-\mathtt{p}}{\mathtt{p}} \end{aligned} \tag{5.14}$$

which we were able to verify in HOL, using the Geometric random variable `prob_geom` and the formalization infrastructure, given in Chapter 3, and the probability theory principles, formalized in [36].

Theorem 5.5 specifies the average message delay relation of a Stop-and-Wait protocol in terms of individual delays of the various autonomous processes, which are the basic building blocks of the protocol. Thus, it allows us to tweak various parameters of the protocol to optimize its performance for any given conditions. It is important to note here that the result of Theorem 5.5 is not new and the performance analysis of Stop-and-Wait protocols, based on Equation (5.3), existed since the early days of their introduction, however, using theoretical paper-and-pencil proof techniques. On the other hand, to the best of our knowledge, this is the first time that such a relation has been mechanically verified without any loss in accuracy or precision of the results. It therefore provides a superior approach to both paper-and-pencil proofs and simulation based performance analysis techniques. The successful handling of the performance analysis of the Stop-and-Wait protocol in the HOL theorem prover clearly demonstrates the practicability and effectiveness of the proposed probabilistic analysis approach.

5.6 Summary and Discussions

The main objective of this chapter was to demonstrate the feasibility and usefulness of the proposed formal probabilistic analysis approach. For this purpose, we utilized the proposed infrastructure regarding the formalization and verification of statistical properties, given in Chapter 3, to conduct the performance analysis of the Stop-and-Wait protocol, which is a classical example of a real-time system.

A higher-order-logic specification for the Stop-and-Wait protocol is presented, with the noise effect modeled as a Bernoulli random variable. The performance related properties are then formally verified, using this model along with the formalization presented in Chapter 3, in HOL. The accuracy of the results has been one of the primary motivations of the proposed approach. The Stop-and-Wait protocol case study clearly demonstrates this aspect as the results match the ones obtained using traditional paper-and-pencil proof methods and are thus 100% accurate. The next main feature of the proposed approach is the ability to precisely reason about statistical properties, which is something that, to the best of our knowledge, has not been achieved by any existing formal probabilistic analysis technique. Our case study also demonstrates this feature as we presented the formal verification of the classical average message delay relation for the Stop-and-Wait protocol.

The formal functional verification and performance analysis of real-time systems has been an open problem for quite some time. Though, there are computer based tools that can be used to conduct these two tasks individually but this leaves a gap in the completeness of the analysis as two different models at different abstraction levels are used for these tasks and the equivalence verification between these models is not a very straightforward task. This issue can be resolved using the proposed probabilistic analysis methodology as has been observed in the case of the Stop-and-Wait protocol.

We used the same formal model of the Stop-and-Wait protocol for the verification of both functional and performance related properties.

It is also worth mentioning here that due to the fact that the Stop-and-Wait protocol bears most of the essential characteristics of the present day real-time systems, other real-time systems may also be analyzed using the analysis approach presented for the Stop-and-Wait protocol in this chapter. The existing library of formalized discrete and continuous random variables, presented in Chapters 3 and 4 of this thesis, can be utilized for this purpose.

As has been pointed out earlier, one of the major limitations of the proposed probabilistic analysis approach is the associated user interaction, i.e., the user needs to guide the proof tools manually since we are dealing with higher-order logic. The Stop-and-Wait protocol is a real-time system and thus involves a subtle interaction of a number of autonomous components. This fact makes its formalization and verification even more complicated. We had to undertake a couple of steps to combat the complexity of the analysis. First of all, we built upon existing HOL theories whenever it was possible. Secondly, we chose the discrete time domain instead of continuous time for the analysis, which allows us to use the powerful induction technique for verification and thus minimizes the proof effort considerably. The functional verification and performance analysis tasks translated to approximately 6000 lines of HOL code and we had to spend about 300 man-hours on the analysis, presented in this chapter.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have proposed to use higher-order-logic theorem proving for probabilistic analysis as a complementary approach to the state-of-the-art simulation and probabilistic model checking based techniques. The main idea is to use random variables formalized in higher-order logic to model systems, which need to be analyzed, and to verify the corresponding probabilistic and statistical properties in a theorem prover. We believe that because of the formal nature of the models, the analysis will be free of approximation and precision errors and due to the high expressive nature of higher-order logic a wider range of systems can be analyzed. Thus, the proposed approach can prove to very useful for the performance and reliability optimization of safety critical and highly sensitive engineering and scientific applications.

Towards the development of a successful higher-order-logic theorem proving based probabilistic analysis infrastructure, the thesis mainly contributes in two directions. Firstly, it presents a formalization infrastructure that can be utilized to formally express and reason about statistical properties, such as expectation, variance

and tail distribution bounds, for probabilistic systems that involve discrete random variables. Secondly, it describes the construction details of a framework for the formalization and CDF verification of all continuous probability distributions for which the inverse of the CDF can be expressed in a closed mathematical form. These formalized continuous random variables can in turn be used to formally model systems with continuous random components and formally reason about their corresponding probability distribution properties. The work is conducted using the HOL theorem prover and the main reason behind this choice is to be able to utilize the available higher-order-logic formalization of measure and probability theories, presented in [36]. It is important to note that the proposed approach is not specific to the HOL theorem prover, though, and can be adapted to any other higher-order-logic theorem prover, such as Isabelle [65], Coq [17] or PVS [68].

We utilized the above mentioned mathematical foundations to present the formal probabilistic analysis of three examples, i.e., the Coupon Collector's Algorithm, the Roundoff error in a digital processor and the Stop-and-Wait protocol. 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 probabilistic analysis problems by the proposed approach clearly demonstrates its feasibility for real-world probabilistic analysis issues.

The main limitation of the proposed approach is the associated significant user interaction, i.e., the user needs to guide the proof tools manually since we are dealing with higher-order logic, which is known to be non-decidable. Because of this, the proposed approach should not be viewed as an alternative to methods such as simulation and model-checking for the performance analysis of real-time systems but rather as a complementary technique, which can prove to be very useful when precision of the

results is of prime importance.

For our verification, we utilized the HOL theories of Boolean algebra, sets, lists, positive integers, real numbers, measure and probability. Our results can therefore be regarded as a useful indicator of the state-of-the-art in theorem proving. Based on this experience, we can say that formalizing mathematics in a mechanical system is a tedious work that requires deep understanding of both mathematical concepts and theorem-proving. We often came across proving lemmas that are commonly known to be true but their formal proofs could not be found even after browsing quite a few mathematical texts on that specific topic and thus we had to first develop a formal paper-and-pencil proof of these lemmas before translating them to HOL. The automated reasoners aid somewhat in the proof process by automatically verifying some of the first-order-logic goals but most of the times we had to guide the tool by providing the appropriate rewriting and simplification rules. Thus, the HOL code for the formalization presented in this thesis consists of approximately 16,000 lines. On the other hand, we found theorem-proving very efficient in book keeping. For example, it is very common to get confused with different variables and mathematical notations and make human errors when working with large paper-and-pencil proofs, which leads to the loss of a lot of effort. Whereas in the case of mechanical theorem provers such problems do not exist. Another major advantage of theorem proving is that once the proof of a theorem is established, due to the inherent soundness of the approach, it is guaranteed to be valid and the proof can be readily accessed, contrary to the case of paper-and-pencil proofs where we have to explore the enormous amount of mathematical literature to find proofs. Thus, it can be concluded that theorem-proving is a tedious but promising field, which can help mathematicians to cope with the explosion in mathematical knowledge and to save mathematical concepts from

corruption. Also, there are areas, such as security critical software, in military, space travel or medicine applications for example, where theorem-proving will soon become a dire need.

6.2 Future Work

The formalization and verification results, presented in this thesis, open new avenues in using theorem proving for the precise analysis of probabilistic systems as a complement to the probabilistic model checking and simulation techniques. Building on our results, more features can be added to strengthen the capabilities of the theorem proving based probabilistic analysis framework. Some of the future extensions are outlined below.

- The formal verification of Chernoff bounds [61], which are extremely powerful and give exponentially decreasing bounds on the tail distribution, would also be of great benefit. The formally verified Markov's inequality and the formal definition of expectation of a function of a random variable, presented in this thesis, can be utilized for this purpose.

- Formally, a Poisson distribution represents the limit value of a Binomial(m, p) random variable, when m approaches infinity. A higher-order-logic formalization of the Binomial random variable is presented in this thesis, which may be utilized to formalize the Poisson random variable and verify its expectation relation. This result can be in turn used to formalize the mathematical concept of the Poisson process in higher-order logic, which is one of the most frequently used stochastic process [87].

- We mainly targeted the formalization of the continuous random variables for

which the inverse of the CDF can be expressed in a closed mathematical form in this thesis. A potential future direction would be to utilize the formalized Standard Uniform random variable, presented in this thesis, to formalize other continuous probability distributions, for which the inverse CDF is not available in a closed mathematical form, such as, the Normal distribution. This can be done by exploring the formalization of nonuniform random number generation techniques such as Box-Muller and acceptance/rejection [20]. Similarly, the ability to formally reason about multiple continuous random variables would be a very useful extension to the formal probabilistic analysis domain.

- The formalization and verification of statistical properties regarding Continuous random variables needs to be tackled as well. For this purpose, an expectation function may be formalized using the higher-order-logic formalization of Lebesgue integration theory [69]. Building upon this definition, we can formalize the mathematical concept of variance and verify Markov and Chebyshev's inequalities for continuous random variables as has been done for the case of discrete random variables in this thesis.

- The approach presented in this thesis for the analysis of the Stop-and-Wait protocol is quite general and can be extended with various new features and used for other kinds of real-time systems as well. The extension to other ARQ protocols, such as Go-back-N and Selective-Repeat [49], is quite straightforward as they can also be formalized using the process structure, given in Figure 5.2, with modifications in the behavior of some of the histories.

- Based on our results, a number of safety critical probabilistic analysis problems can be analyzed in a formal way. Some of the interesting application areas

include statistical analysis of Digital Signal Processing systems, probabilistic analysis of digital hardware circuits and performance analysis of protocols used for security, cache coherence, and telecommunication systems.

Bibliography

[1] R. Alur. *Techniques for Automatic Verification of Real-Time Systems.* PhD Thesis, Stanford University, Stanford, USA, 1992.

[2] T. Amnell, G. Behrmann, J. Bengtsson, P.R. D'Argenio, A. David, A. Fehnker, T. Hune, B. Jeannet, K. Guldstrand Larsen, M.O. Möller, P. Pettersson, C. Weise, and W. Yi. Uppaal - Now, Next, and Future. In *Modeling and Verification of Parallel Processes*, volume 2067 of *LNCS*, pages 99–124. Springer, 2001.

[3] P. Audebaud and C. Paulin-Mohring. Proofs of Randomized Algorithms in Coq. In *Mathematics of Program Construction*, volume 4014 of *LNCS*, pages 49–68. Springer, 2006.

[4] C. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen. Model Checking Algorithms for Continuous time Markov Chains. *IEEE Transactions on Software Engineering*, 29(4):524–541, 2003.

[5] D. Beyer, C. Lewerentz, and A. Noack. Rabbit: A Tool for BDD-based Verification of Real-Time Systems. In *Computer Aided Verification*, volume 2725 of *LNCS*, pages 122–125. Springer, 2003.

[6] J. Bialas. The σ-Additive Measure Theory. *Journal of Formalized Mathematics*, 2, 1990.

[7] P. Billingsley. *Probability and Measure.* John Wiley, 1995.

[8] J. Billington, G.E. Gallasch, and L. Petrucci. Fast Verification of the Class of Stop-and-Wait Protocols Modelled by Coloured Petri Nets. *Nordic Journal of Computing*, 12(3):251–274, 2005.

[9] B.W. Brown and L.B. Levy. Certification of Algorithm 708: Significant Digit Computation of the Incomplete Beta. In *ACM Trans. on Mathematical Software*, volume 20, pages 393–397, 1994.

[10] C.E. Brown. *Automated Reasoning in Higher-order Logic.* College Publications, 2007.

[11] G. Bucci, L. Sassoli, and E. Vicario. Correctness Verification and Performance Analysis of Real-Time Systems Using Stochastic Preemptive Time Petri Nets. *IEEE Trans. on Software Engineerin*, 31(11):913–927, 2005.

[12] R. Cardell-Oliver. *The Formal Verification of Hard Real-time Systems.* PhD Thesis, University of Cambridge, Cambridge, UK, 1992.

[13] O. Celiku. Quantitative Temporal Logic Mechanized in HOL. In *International Colloquium Theoretical Aspects of Computing*, volume 3722 of *LNCS*, pages 439–453. Springer, 2005.

[14] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[15] E. Clarke, O. Grumberg, and D. Long. Verification Tools for Finite State Concurrent Systems. In *A Decade of Concurrency-Reflections and Perspectives*, volume 803 of *LNCS*, pages 124–175. Springer, 1993.

[16] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking.* The MIT Press, 2000.

[17] COQ. http://pauillac.inria.fr/coq/, 2008.

[18] L. de Alfaro. *Formal Verification of Probabilistic Systems.* PhD Thesis, Stanford University, Stanford, USA, 1997.

[19] M. DeGroot. *Probability and Statistics.* Addison-Wesley, 1989.

[20] L. Devroye. *Non-Uniform Random Variate Generation.* Springer-Verlag, 1986.

[21] M. Duflot, L. Fribourg, T. Hérault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic Model Checking of the CSMA/CD Protocol using PRISM and APMC. In *Proc. 4th Workshop on Automated Verification of Critical Systems*, pages 195–214. Elsevier Science, 2004.

[22] Microsoft Excel. http://office.microsoft.com, 2008.

[23] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 2. Wiley, 1971.

[24] M. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer, 1996.

[25] G.E. Gallasch and J. Billington. A Parametric State Space for the Analysis of the Infinite Class of Stop-and-Wait Protocols. In *Model Checking Software*, volume 3925 of *LNCS*, pages 201–218. Springer, 2006.

[26] M.J.C. Gordon. Mechanizing Programming Logics in Higher-Order Logic. In *Current Trends in Hardware Verification and Automated Theorem Proving*, pages 387–439. Springer, 1989.

[27] M.J.C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic.* Cambridge University Press, 1993.

[28] C.M. Grinstead and J.L. Snell. *Introduction to Probability.* American Mathematical Society, 1997.

[29] F.V.A. Guerra, J.C.A. Figueiredo, and D.D.S. Guerrero. Protocol Performance Analysis Using a Timed Extension for an Object Oriented Petri Net Language. *Electronic Notes in Theoretical Computer Science*, 130:187–209, 2005.

[30] A. Gupta. Formal Hardware Verification Methods: A Survey. *Formal Methods in System Design*, 1(2-3):151–238, 1992.

[31] J. Harrison. Formalized Mathematics. Technical Report 36, Turku Centre for Computer Science, 1996.

[32] J. Harrison. *Theorem Proving with the Real Numbers.* Springer, 1998.

[33] K. Havelund and N. Shankar. Experiments in Theorem Proving and Model Checking for Protocol Verification. In *Industrial Benefit and Advances in Formal Methods*, volume 1051 of *LNCS*, pages 662–681. Springer, 1996.

[34] H. Hermanns, J.P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov Chain Model Checker. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1785 of *LNCS*, pages 347–362. Springer, 2000.

[35] N.J. Higham. *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics, 2002.

[36] J. Hurd. *Formal Verification of Probabilistic Algorithms.* PhD Thesis, University of Cambridge, Cambridge, UK, 2002.

[37] J. Hurd, A. McIver, and C. Morgan. Probabilistic Guarded Commands Mechanized in HOL. *Theoretical Computer Science*, 346:96–112, 2005.

[38] C. Huyghens. *De Ratiociniis in Ludo Aleae.* Excercitationum Mathematicarum Libri Quinque, V., 1657.

[39] B. Jeannet, P.D. Argenio, and K. Larsen. Rapture: A Tool for Verifying Markov Decision Processes. In *Tools Day,* 13^{th} *Int. Conf. Concurrency Theory*, Brno, Czech Republic, 2002.

[40] W.J. Kennedy and J.E. Gentle. *Statistical Computing.* Marcel-Dekker, 1980.

[41] R. Khazanie. *Basic Probability Theory and Applications.* Goodyear, 1976.

[42] L. Knüsel. Computation of the Chisquare and Poisson Distribution. *SIAM Journal on Scientific and Statistical Computing*, 7(3):1022–1036, 1986.

[43] L. Knüsel and B. Bablok. Computation of the Noncentral Gamma Distribution. *SIAM Journal on Scientific and Statistical Computing*, 17:1224–1231, 1996.

[44] D.E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley Professional, 1998.

[45] M. Kwiatkowska, G. Norman, and D. Parker. Quantitative Analysis with the Probabilistic Model Checker PRISM. *Electronic Notes in Theoretical Computer Science*, 153(2):5–31, 2005. Elsevier.

[46] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic Verification of Real-Time Systems with Discrete Probability Distributions. *Theoretical Computer Science*, 282(1):101–150, 2002. Elsevier.

[47] M.Z. Kwiatkowska, G. Norman, and D. Parker. Stochastic Model Checking. In *Formal Methods for Performance Evaluation*, volume 4486 of *LNCS*, pages 220–270. Springer, 2007.

[48] P. L'Ecuyer. Uniform Random Number Generation. *Annals of Operations Research*, 53:77–120, 1994.

[49] A. Leon Garcia and I. Widjaja. *Communication Networks: Fundamental Concepts and Key Architectures.* McGraw-Hill, 2004.

[50] A. Levine. *Theory of Probability.* Addison-Wesley series in Behavioral Science, Quantitative Methods, 1971.

[51] D.J.C. MacKay. Introduction to Monte Carlo Methods. In *Learning in Graphical Models, NATO Science Series*, pages 175–204. Kluwer Academic Press, 1998.

[52] W. Mao. *Modern Cryptography: Theory and Practice.* Prentice Hall, 2003.

[53] M.A. Marson, A. Bianco, L. Ciminiera, R. Sisto, and A. Valenzano. A LOTUS Extension for the Perfomance Analysis of Distributed Systems. *IEEE Trans. on Networking*, 2(2):151–165, 1994.

[54] mathStatica. www.wolfram.com/products/applications/mathstatica, 2008.

[55] B.D. McCullough. Assessing the Reliability of Statistical Software: Part I. *The American Statistician*, 52(4):358–366, 1998.

[56] B.D. McCullough. Assessing the Reliability of Statistical Software: Part II. *The American Statistician*, 53(2):149–159, 1999.

[57] B.D. McCullough. The Accuracy of Mathematica 4 as a Statistical Package. *Computational Statistics*, 15(2):279–299, 2000.

[58] B.D. McCullough and B. Wilson. On the Accuracy of Statistical Procedures in Microsoft Excel 2003. *Computational Statistics and Data Analysis*, 49:1244–1252, 2005.

[59] R. Milner. A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, 17:348–375, 1977.

[60] Minitab. http://www.minitab.com, 2008.

[61] M. Mitzenmacher and E. Upfal. *Probability and Computing.* Cambridge University Press, 2005.

[62] A. Nedzusiak. σ-fields and Probability. *Journal of Formalized Mathematics*, 1, 1989.

[63] D. Parker. *Implementation of Symbolic Model Checking for Probabilistic System.* PhD Thesis, University of Birmingham, Birmingham, UK, 2001.

[64] Y. Patt. Opening and keynote 1. In *Proc. IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, USA, 2004. page 1.

[65] L.C. Paulson. *Isabelle: A Generic Theroem Prover*, volume 828 of *LNCS.* Springer, 1994.

[66] L.C. Paulson. *ML for the Working Programmer.* Cambridge University Press, 1996.

[67] PRISM. www.cs.bham.ac.uk/~dxp/prism, 2008.

[68] PVS. http://pvs.csl.sri.com, 2008.

[69] S. Richter. *Formalizing Integration Theory, with an Application to Probabilistic Algorithms.* Diploma Thesis, Technische Universitat Munchen, Department of Informatics, Germany, 2003.

[70] D.S. Riha, B.H. Thacker, M.P. Enright, L. Huyse, and S.H.K. Fitch. Recent Advances of the NESSUS Probabilistic Analysis Software for Engineering Applications. In *AIAA/ASME/ASCE/AHS/ASC 43rd Structures, Structural Dynamics, and Materials (SDM) Conference*, pages 2002–1268, Denver, USA, 2002. AIAA, Inc.

[71] C. Rose and M.D. Smith. *Mathematical Statistics with Mathematica.* Springer, 2002.

[72] S.M. Ross. *Simulation.* Academic Press, 2002.

[73] J. Rutten, M. Kwaiatkowska, G. Normal, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilisitc Systems*, volume 23 of *CRM Monograph Series.* American Mathematical Society, 2004.

[74] SAS. http://sas.com/technologies/analytics/statistics/stat/index.html, 2008.

[75] K. Sen, M. Viswanathan, and G. Agha. VESTA: A Statistical Model-Checker and Analyzer for Probabilistic Systems. In *Proc. IEEE International Conference on the Quantitative Evaluation of Systems*, pages 251–252, 2005.

[76] M. Sofroniou. Numerics in Mathematica 3.0. *The Mathematica Journal*, 6(4):64–73, 1996.

[77] SPSS. http://www.spss.com/, 2008.

[78] MATLAB Statistics. http://www.mathworks.com/products/statistics, 2008.

[79] L.J. Steggles and P. Kosiuczenko. A Timed Rewriting Logic Semantics for SDL: A case study of the Alternating Bit Protocol. *Electronic Notes in Theoretical Computer Science*, 15:83–104, 1998.

[80] N.V. Stenning. A Data Transfer Protocol. *Computer Networks*, 1:99–110, 1976.

[81] D. Stirzaker. *Elementary Probability.* Cambridge Press, 2003.

[82] I. Suzuki. Formal Analysis of the Alternating Bit Protocol by Temporal Petri Nets. *IEEE Trans. on Software Engineering*, 16(10):1273–1281, 1990.

[83] A.S. Tanenbaum. *Computer Networks.* Prentice-Hall International,, 1996.

[84] K.S. Tridevi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* Wiley-Interscience, 2002.

[85] L. Wells. Performance Analysis Using Coloured Petri Nets. In *Peoc. IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pages 217–222. IEEE Computer Society, 2002.

[86] B. Widrow. Statistical Analysis of Amplitude-quantized Sampled Data Systems. *AIEE Trans. (Applications and Industry)*, 81:555–568, January 1961.

[87] R.D. Yates and D.J. Goodman. *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers.* Wiley, 2005.

Biography

Education

- **Concordia University:** Montreal, Quebec, Canada

  Ph.D candidate, in Electrical Engineering, (Jan. 04 - present)

- **Concordia University:** Montreal, Quebec, Canada

  M.Eng., in Electrical Engineering, (Sept. 99 - Apr. 01)

- **N-W.F.P. University of Engg. and Tech.:** Peshawar, Pakistan

  B.Sc., in Electrical Engineering, (Jan. 93 - Dec. 97)

Work History

- **Hardware Verification Group (HVG):** Concordia University

  Research Assistant, (Jan. 05 - present)

- **LSI Logic Corporation:** Ottawa, Canada

  ASIC Customer Engineer, (Jun. 01 - May 03)

- **Pakistan Revenue Automation (Pvt.) Ltd.:** Islamabad, Pakistan

  Design Engineer, (Jan. 98 - May 99)

Publications

- **Journal Papers** [1]

  Bio-Jr-1 O. Hasan and S. Tahar, "Formalization of the Standard Uniform Random Variable", Theoretical Computer Science, 382(1):71-83, Elsevier, 2007.

[1] Bio:Jr refers to a journal paper in the biography section.

Bio-Jr-2 O. Hasan and S. Tahar, "Using Theorem Proving to Verify Expectation and Variance for Discrete Random Variables", Submitted to Journal of Formal Aspects of Computing, Springer. [18 pages]

Bio-Jr-3 O. Hasan and S. Tahar, "Formal Verification of Tail Distribution Bounds in the HOL Theorem Prover", Submitted to Mathematical Methods in the Applied Sciences, Wiley InterScience. [32 pages]

Bio-Jr-4 O. Hasan and S. Tahar, "Performance Analysis and Functional Verification of the Stop-and-Wait Protocol in HOL", Submitted to Journal of Automated Reasoning, Springer. [31 pages]

- **Refereed Conference Papers** [2]

Bio-Cf-1 O. Hasan and S. Tahar, "Performance Analysis of ARQ Protocols using a Theorem Prover", In Performance Analysis of Systems and Software, IEEE Computer Society, (*To appear*).

Bio-Cf-2 O. Hasan and S. Tahar, "Verification of Tail Distribution Bounds in a Theorem Prover", In Numerical Analysis and Applied Mathematics, AIP Conference Proceedings, Volume 936, 2007, pp. 259-262.

Bio-Cf-3 O. Hasan and S. Tahar, "Verification of Expectation Properties for Discrete Random Variables in HOL", In Theorem Proving in Higher-Order Logics, LNCS 4732, Springer, 2007, pp. 119-134.

Bio-Cf-4 O. Hasan and S. Kort, "Automated Formal Synthesis of Wallace Tree Multipliers", In Circuits and Systems, IEEE International Midwest Symposium, 2007, pp. 293-296.

[2] Bio:Cf refers to a conference paper in the biography section.

Bio-Cf-5 O. Hasan and S. Tahar, "Formalization of Continuous Probability Distributions", In Automated Deduction, LNCS 4603, Springer, 2007, pp. 2-18.

Bio-Cf-6 O. Hasan and S. Tahar, "Verification of Probabilistic Properties in the HOL Theorem Prover", In Integrated Formal Methods, LNCS 4591, Springer, 2007, pp. 333-352.

- **Technical Reports** [3]

Bio-Tr-1 O. Hasan and S. Tahar, "Formal Verification of Expectation and Variance for Discrete Random Variables", ECE Dept., Concordia University, Jun. 2007. [27 pages]

Bio-Tr-2 O. Hasan and S. Tahar, " Formalization of Continuous Probability Distributions", ECE Dept., Concordia University, Feb. 2007. [42 pages]

Bio-Tr-3 O. Hasan and S. Tahar, "Standard Uniform Distribution Theory in HOL-4", ECE Dept., Concordia University, Dec. 2006. [14 pages]

Bio-Tr-4 O. Hasan and S. Tahar, "Formalization of the Standard Uniform Random Variable in HOL", ECE Dept., Concordia University, Dec. 2006. [20 pages]

Prizes and Awards

- Teaching Fellowship, ECE Dept., Concordia University: 2007-08
- International Fee Remission Award, Concordia University: 1999-00, 2000-01
- Quaid-e-Azam Award, Ministry of Education, Pakistan: 1997-98
- Annual Merit Award, N-W.F.P. UET: 1994-95, 1995-96, 1996-97

[3] Bio:Tr refers to a technical report in the biography section.