

Formal Verification of Negative Binomial Distribution in Higher Order Logics

Muhammad Wisal

Department of Computer Engineering

APPROVED:

Dr Amir Hasan

Dr Osman Hasan

Dr Adnan A. Khan

Dr Abdul Khaliq
Chairman of Department ECE CASE

©Copyright

by

Muhammad Wisal

2012

Dedicated to my

FATHER

Muhammad Shoaib Khan

Formal Verification of Negative Binomial Distribution in Higher Order Logics

by

Muhammad Wisal, SP-2009/MSc.CE/039.

THESIS

Presented to the Faculty of the Center of Advance studies in Engineering (CASE)

The University of Engineering and Technology Taxila

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Engineering

University of Engineering and Technology Taxila

February 2012

Acknowledgements

All glory be to Allah, First and foremost, I would like to thank almighty ALLAH for his kindness and blessing which guided me through all the challenges I faced in thesis.

I would like to express my thanks and sincere gratitude to my supervisor (External Co supervisor at NUST), Dr. Osman Hasan, for his strong support, encouragement and guidance through out my research. He was always approachable and his insights about research and immense knowledge in the field of formal methods have strengthened this work significantly. His emphasis for excellence kept me well-directed and focused. His cheerful and enthusiastic encouragement was a source of strength for me to complete this thesis. I would also like to acknowledge his efforts for establishing System Analysis and Verification (SAVE) Lab at SEECS, NUST which is a first formal Methods Lab in Pakistan.

I would like to express my gratitude to my supervisor (at CASE) Dr. Amir Hassan for giving me time and taking interest in my work. I sincerely thank Dr. Abdul Khaliq (chairman Electrical & Computer Engineering Department) and Dr. Adnan A Khan for serving as my thesis committee members. I would also like to thank Dr. Asad Mehmood who guided me and help me alot in starting my thesis at SAVE Lab, SEECS, NUST.

I would like to thank my colleagues in CASE Ihsan Ahmed and Iqbal haider, without their motivation and encouragement, I would not have reached this point. I would like also to thank my SAVE lab friends specially Umair siddique for his movtivation during the tough stages of thesis. My thanks also goes to all those peoples who provided me help on NRCH and other mathematics forums.

Finally, I would like to thank my mother for her prayers and support which always provide me the strength and toughness during the whole of my thesis duration. I wish to thank all my family members, specially my brother Faheem jan for his motivation and support for my higher studies. I would also thank my uncle Tahir Khan and his family for their care and support during thesis.

NOTE: This thesis was submitted to my Supervising Committee on the Fall, 2011.

Abstract

The Negative Binomial random variables are very widely used in the probabilistic analysis of many safety-critical domains, such as transportation, medicine and reliability assurance of software. Such systems are usually analyzed by simulation techniques while modeling the Negative Binomial random variables using pseudo-random number generators. Due to the inherent nature of such analysis, the results can never be termed as accurate, which is extremely undesirable in the case of safety-critical systems where the accuracy of the results is of paramount importance. To overcome this limitation, this thesis presents a formalization of the Negative Binomial random variable based on a measure theoretic formalization of a probability space in higher-order logic. The formalized Negative Binomial random variable is then formally verified based on its corresponding probability mass function and expectation properties using the HOL4 theorem prover. These results facilitate accurate analysis of systems that utilize the Negative Binomial random variable. For illustration purposes, the thesis presents a software reliability analysis example.

Table of Contents

	Page
Acknowledgements	v
Abstract	vii
Table of Contents	viii
Chapter	
1 Introduction	1
1.1 Motivation	1
1.2 Probabilistic Analysis Approaches	1
1.2.1 Simulation	1
1.2.2 Model Checking	3
1.2.3 Theorem Proving	4
1.3 Proposed Methodology	5
1.4 Organization of the Thesis	6
1.5 Related Work	7
2 Preliminaries	8
2.1 HOL4 Theorem Prover	8
2.1.1 Terms	10
2.1.2 Types	10
2.1.3 Inference Rules	11
2.1.4 Theorems	11
2.1.5 Theories	11
2.1.6 Proofs in HOL	12
2.1.7 HOL Notations	12
2.2 Probability Space and Random Variables in HOL4	13
3 HOL4 Formalization of Negative Binomial Random Variable	18

- 3.1 Formal Specification of the Negative Binomial Random Variable 18
- 3.2 Formal Verification of the Negative Binomial Random Variable 20
 - 3.2.1 Probability Mass Function 20
 - 3.2.2 Expectation 25
- 4 Application 28
 - 4.1 Software Reliability Application 28
 - 4.2 Concluding Remarks 29
- 5 Conclusions and Future Work 31
 - 5.1 Conclusions 31
 - 5.2 Future Work 31

Chapter 1

Introduction

1.1 Motivation

Negative Binomial random variable is closely associated with the Binomial random variable that returns the number of successes in k Bernoulli trials. The Negative Binomial random variable returns the number of successes for a given number of failures, say n , in successive Bernoulli trials. This unique distribution is widely used for analyzing systems in many scientific and engineering domains, such as transportation [26] (accident proneness [38, 24, 32]), medicine (disease prevalence in a population [34], [35]), agriculture [3] and software (reliability assessment [39]).

1.2 Probabilistic Analysis Approaches

Due to the vast application Negative Binomial random variable, Traditionally, all the Negative Binomial random variable based probabilistic analysis is conducted using simulation based methods like MRET [7] and PRET. In this section we present analysis approaches used for the analysis of probabilistic system involving negative binomial random.

1.2.1 Simulation

Traditionally, all the Negative Binomial random variable based probabilistic analysis is conducted using simulation based methods like MRET [7] and PRET. The main idea behind this approach is model the Negative Binomial random variable as a function of a pseudo-random number generator and then use this function to construct a model of

the system behavior. Simulation uses Monte Carlo Method [27] techniques to analyze this computer model, idea mainly is to answer a problem using approximations such as for probability distribution of random variable analyzing a large sample space of random variable. Similarly, then approximate statistical quantities, such as mean and variance based on the sampling process data collected, by mathematical modelling these quantities in a computer. Due to the usage of pseudo-random numbers and the inherent nature of simulation, the probabilistic analysis results attained by this technique can never be termed as 100% accurate. It is because of this approximate nature of simulation that different software packages produce different results to the same probabilistic 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 pseudo-random numbers; the numbers seem random but are actually the output of a random number generator (RNG) which is basically a deterministic algorithm these RNG repeats itself after a period (number of calls).
- Enormous amount of CPU time requirement is an another major limitation on simulation based probabilistic analysis in order to attain meaningful estimate of random phenomena. Simulation approach large number of simulations cycles generally required in order to calculate the probabilistic quantities and at times becomes impractical when each simulation step involves extensive computations.
- The third major drawback of simulation is roundoff and truncation errors, due to finite precision of computers to represents number roundoff and truncation is done which make the results of simulation inaccurate.

In safety and financial critical systems, such as medical, transportation and stock exchange we cannot rely on these unreliable results due to fact that it can result serious problem while analyzing, where a mismatch between the actual and predicted system performance may result in disastrous (economical loss or even loss of human lives in extreme cases).

Formal methods [6] are capable of conducting precise system analysis and thus overcome the above mentioned limitations of simulation. 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.

1.2.2 Model Checking

Probabilistic model checking [4, 36] is one of the commonly used formal probabilistic analysis technique. It involves the construction of a precise state-based mathematical model of the given probabilistic system, which is then subjected to exhaustive analysis to formally verify if it satisfies a set of formally represented probabilistic properties. Numerous probabilistic model checking algorithms and methodologies have been proposed in the open literature, e.g., [8, 30], and based on these algorithms, a number of tools have been developed, e.g., E⁺MC² [20], PRISM [33, 25], VESTA [37] and Rapture [23]. Along with results accuracy, the ability to perform the analysis automatically most promising feature of model checking based probabilistic analysis. But huge limitation on model checking is that we can analyze only those systems that can only be expressed as probabilistic finite state machines using model checking. State space explosion [5] is another major limitation of the probabilistic model checking approach. The probabilistic system have a very large space, or sometimes space is even infinite. As with limited resources of time and memory it is impossible to explore the entire state space. We are not aware of any model checking based analysis related to the Negative Binomial random variable. The main reason is in probabilistic model checking have not the capability to model pure mathematical expressions and due to this big limitation PMF, *Cumulative Distribution Function* (CDF) or expectation or variance of a random behavior can be modelled using model checkin based probabilistic analysis, which are basically functions of the range of a random variable. Thus, the probabilistic model checking approach, even though its capability of accurate analysis, is quite limited in terms of handling a variety of systems that involves the Negative Binomial

random variable.

1.2.3 Theorem Proving

The other formal verification analysis approach is theorem proving [14]. It is widely used for formal verification of safety critical systems. These mathematical models can be built on different types of logic, such as, propositional logic, first-order logic or higher-order logic, depending upon the expressibility requirement. For example, the use of higher-order logic is advantageous over first order logic in terms of the availability of additional quantifiers and highly expressive nature of higher-order logic. The approach is to model systems mathematically in appropriate logic and then verify properties of interest using computer based formal tools. Theorem proving uses formal logics as a modeling medium which makes it a very flexible formal way to verify any mathematical model of a system, theorem proving has the capability to formally verify any system that can be modelled mathematically. Theorem provers are built using basic axioms and primitive inference rules where soundness is assured so that all new theorems must be proved using axioms and primitive inference rules or using all already existing inference rules or proved theorems.

The verification effort of a theorem in a theorem prover depends upon the underlying logic. Theorems involving first-order logic require less human interaction but first-order logic lacks the capability to analyze probabilistic systems, we cannot model random variables as functions using first-order logic. Higher-order logic has the capability to model probabilistic systems mathematically but at the cost of user interaction. Higher-order logic theorem proofs are very complex and need guidelines from the user to interact with the theorem prover. We have used higher-order logic to model the Negative Binomial random variable and used this formal model to express its statistical and probabilistic properties (PMF, Mean) in this thesis.

This thesis provides a formalization of the Negative Binomial random variable using higher-order logic, higher-order logic is a system of deduction almost all classical mathematics theories with a precise semantics developed with higher-order logic. Our formalization

builds upon an existing measure theoretic formalization of a probability space [22]. In order to ensure the correctness of our formalization, we verify its corresponding *Probability Mass Function* (PMF) and expectation properties using a computer based higher-order-logic theorem prover HOL4 [16]. It is worth mentioning here that while formally verifying the PMF relation of the Negative Binomial random variable, we explored a plethora of literature including text books and research articles but could not find its detailed mathematical proof. In most case, a brief proof script was provided, which was found to be missing the exact reasoning process (See for example, [11, 12]). We therefore had to develop the paper-and-pencil based proof ourself before verifying it. Thus, the presented proof of the PMF relation of the Negative Binomial random variable can be considered a mathematical research contribution itself. Moreover, its availability in the HOL4 theorem prover ensures the soundness of the work and the proof steps can be traced back to the foundational mathematical theories very easily.

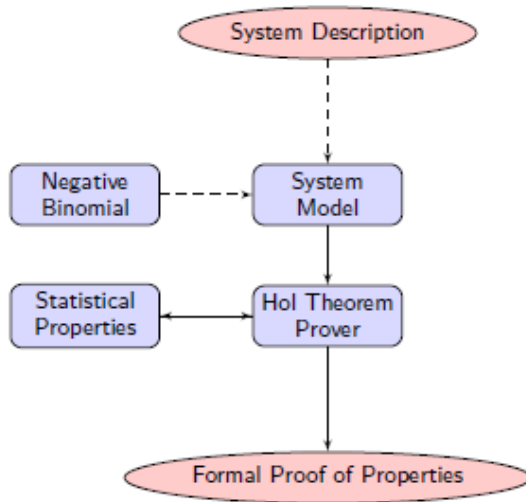
Besides ensuring the correctness of our model, the formally verified PMF and the expectation relations of the Negative Binomial random variable also facilitates the formal analysis of systems involving the Negative Binomial random variable. The main idea is to utilize the formalized Negative Binomial random variable function to construct a formal model of the system that needs to be analyzed. The formally verified PMF and expectation relations of Negative Binomial random variable can then be used to formally reason about any desired probabilistic and statistical properties of the system within the sound core of HOL4. The probabilistic analysis carried out in this way will be free from approximation and precision issues, due to the formal nature of models and theorem proving, and the accuracy of the results will be independent of the CPU time.

1.3 Proposed Methodology

The main objective of this thesis develop formal methods (theorem proving) capability to accurate analysis of safety critical systems involving randomness following Negative

Binomial distribution. Thesis framework is characterizing as:

1. Model The randomness of systems following negative binomial distribution using thesis model.
2. Formalize the characteristics (properties) of systems using statistical properties (probability mass function, Expectation(Mean) formalized in thesis) of Negative binomial in theorem prover.
3. Accurate Results



1.4 Organization of the Thesis

The rest of the thesis is organized as follows: We provide a review of related work in Section 1.5. Chapter 2 presents some preliminaries including a brief introduction to the HOL4 theorem prover and the measure theoretic formalization of probability space that we build upon in this thesis. In Section 3.1, we formally specify the Negative Binomial random variable in higher-order logic. In Section 3.2, we verify the correctness of the above specification by proving its PMF property. To illustrate the usefulness and practical effectiveness

of the formally verified Negative Binomial random variable, we utilize this formalization to formally analyze a software reliability problem in Section 4. Finally, Section 5.1 concludes the thesis.

1.5 Related Work

The foremost criteria for conducting probabilistic analysis within a higher-order-logic theorem prover is to be able to express probabilistic notions, such as probability of an event and random variables, in higher-order logic and reason about the probability distribution and statistical properties of random variables in a higher-order-logic theorem prover. A formalized probability theory provides the foundations for expressing probabilistic notions. A number of authors, including Hurd [22], Mhamdi [28] and Hölze [21], reported higher-order-logic based formalizations of probability theory. The recent works by Mhamdi [28] and Hölzl [21] are based on extended real numbers (including $\pm\infty$) and provide the formalization Lebesgue integral for reasoning about advanced statistical properties. This way, they are more mature than Hurd's [22] formalization of measure and probability theories, which is based on simple real numbers. However, these recent formalizations do not support a particular probability space like the one presented in Hurd's work. Due to this distinguishing feature, Hurd's formalization [22] has been utilized to verify sampling algorithms of a number of commonly used discrete [22] and continuous random variables [18] based on their probabilistic and statistical properties [19, 17]. Due to the availability of a particular probability space as well as the formalization of probability and statistical properties, we build upon Hurd's formalization of measure and probability theories in this thesis to formalize the Negative Binomial random variable.

Chapter 2

Preliminaries

In this section, we give a brief introduction to the HOL4 theorem prover and Hurd's [22] formalization of probability space and random variables. The intent is to introduce the main ideas along with some notation that is going to be used in the rest of the thesis.

2.1 HOL4 Theorem Prover

A theorem prover (proof assistant) is a computer based tool, it is used for proving goal using mathematical reasoning (logic). System and its desired characteristics are modelled mathematically system characteristics are verified using mathematical reasoning. There are mainly two categories of theorem provers automatic and interactive. In an interactive theorem prover goals proved using significant user-computer interaction while automatic theorem prover can automatically perform different proof tasks. Large number of theorem provers are available in open literature but only few of them are used widely for development and there user community is large. Otter [15] LeanTAP, SATURATE, METEOR, Gandalf, MetiTarski and SETHEO [2] are commonly used automated provers. Isabelle, Coq, HOL, HOL Light, ProofPower, ACL2 and MIZAR [15] are commonly used higher-order logic based interactive theorem provers.

This thesis uses the HOL4 theorem prover to conduct all the formalization of probability analysis of systems using Negative binomial random variable. The main reasons behind this choice include formalization of probability space and random variable of Hurd's work, which are fundamental to our work, and the ability to use Hasan's work [19] to formalize the Negative binomial random variable probability mass function and mean.

HOL4 is a theorem prover using higher-order logics, it is capable of conducting proofs interactively. Mike Gordon at Cambridge University developed the first version of HOL in 1980's. Logic is represented in strongly-typed functional programming language ML [?] in the HOL4 for ensuring secure theorem proving goal. HOL4 theorem prover is based on utilizing Church [?] simple type theory and polymorphism of Hindley-Milner [?] to implement higher-order logic. The first version of HOL is called HOL88 and other versions of HOL are HOL90 and HOL98 and HOL4. Verification framework for both hardware and software have already been successfully done in HOL4 theorem prover along with the platform which can be used for the pure mathematics formalization.

Higher-order-logic theorems are represented using abstract data types of Moscow Language (ML) and then ML procedures are executed that operate on these data types values in order to interact with the theorem prover. The core of HOL4 theorem prover is being build using only 5 basic axioms (i.e. A Boolean variable can be True or False: $\forall a.(a=T)\vee(a=F)$) and 8 primitive inference rules (deduced from basic axioms as, i.e. Equality holds transitive property: $\forall a b c.(a=b)\wedge(b=c)\implies(a=c)$), which are implemented as Moscow language (ML) functions. Every new theorem must be proved(verified) by using these basic axioms and primitive inference (deduced from basic axioms) rules or any other already verified theorems/inference rules in HOL4 theorem prover which assures the soundness of HOL4 theorem prover. The availability of many automatic proof procedures [?] and proof assistants to helping out the user makes HOL4 a automatic theorem prover for simple and basic theorem (goal). The user interaction with a proof editor is required for proving theorems (goal) in steps using necessary tactics while some automatic proof procedures are used for proving some steps automatically.

HOL4 users can store a collection of valid types, axioms, constants and theorems as a collection of theory file in computers so that user can reuse there proved theorems for future work. Once stored, user can load these theories in the HOL4 session so that user is able to use corresponding theorems and definitions. In this way user get rid of the tedious process of regenerating these theories using the primitive inference rules and basic

axioms which allow user to efficiently build upon existing results without starting from initial point every time. HOL4 user have already formalized various mathematical concepts have been and saved them as different HOL4 theories (boolean , real, arithmetic etc are different theories available in HOL4 theorem prover). We have used sets, positive integers, *real* numbers, measure and probability theories out of the useful library available in HOL4 theorem prover in this thesis. The availability of mathematical theories in HOL4 theorem prover was one of the basic motivations for the selection of HOL4 theorem prover, so that our work can benefit using these built-in mathematical theories in HOL4 theorem prover.

2.1.1 Terms

HOL4 have mainly terms of four types: variables, constants, lambda-terms and function applications. Letters beginning with a letter or sequences of digits are variables, e.g. b , y , Gamma.hol . The variables and constants have similar syntax, but we cannot bound variables and constants by quantifiers. An identifier type, i.e., constant or a variable, is determined by a theory; e.g. False , True . Applications for representing the evaluation of a function in HOL4 g at an argument y , we can use different terms instead of using g and y , e.g., h and x . In HOL, λ -terms can be used for terms also called lambda abstractions for denoting functions. λ -terms $\lambda x.f(x)$ is representation a function which takes (domain) x and returns (range) $f(x)$.

2.1.2 Types

Lambda calculus has been implemented in HOL4 theorem prover, every term of HOL4 has a unique type which is either basic type (one of the them) or either the result of applying a type constructor to other types. A type must be assigned to each constant and variable and variables thou having same name but different types are considered as different variables. When we enter a term into HOL4, the type is inferred using the type checking algorithm implemented in HOL, e.g., when $(\sim y)$ is entered into HOL4, the HOL4

type checker automatically deduces the type of variable y must have `bool` type because negation (\sim) has a type `bool \rightarrow bool`. we have to mention explicitly term type, e.g., `(x : bool)or(x:real)` If HOL4 cannot deduce automatically the type of a term.

2.1.3 Inference Rules

We represent new theorems as ML functions and Inference rules are procedures for deriving (proving) these new theorems. There are 8 primitive inference rules in HOL4 and rest of rules are derived from these axioms and inference rules. The rules are *Assumption introduction*, *Beta-conversion*, *Reflexivity*, *Substitution*, *Type instantiation*, *Abstraction*, *Modus Ponens* and *Discharging an assumption* [1].

2.1.4 Theorems

A theorem is basically a formalized statement that may be an axiom or follows from theorems by a primitive inference rule. A theorem consists of a finite set of boolean terms with assumptions Ω and a conclusion S is a boolean term. For example, if (Ω, S) is a theorem in HOL then it is written as $\Omega \vdash S$.

2.1.5 Theories

A collection of theorems is called theory. A theory consists of a set of types, constants, type operators, definitions and theorems in HOL4. It is collection of list of already proved theorems from the axioms and definitions. The HOL user can load these theories in order to utilize theory theorems and definitions available in that theory. The availability of HOL theories allows the user to utilize already proved result and focus on extending the its own goals (results) without wasting its efforts on already proved goals results. Theories in HOL4 are organized in a hierarchical fashion and theories can have other theories as parents and all of the types, definitions, constants, axioms and theorems of parent theory can be used in the child theory. For example, one of the basic theory in

HOL4 is `bool` and this is also parent theory of `ind`. We utilized the HOL theories of real numbers, positive integers, measure and probability theories in our thesis work.

2.1.6 Proofs in HOL

In HOL, forward and backward are two types of interactive proof methods. In a forward proof, the HOL user starts from the axioms and primitive inference rules and tries to verify the goals on top of these axioms, rules and using already proved theorems. The forward proof method is a difficult approach for proving goals as low level details of the proof has been required in advance. A goal directed proof or backward method is the reverse approach of the forward proof approach. Forward proof approach is based on the tactic concept; breaks goal into simple and easy subgoals using ML function. In the forward proof approach, the HOL user starts with the desired goal as main goal (theorem) and then using ML-function break main goal into further simpler subgoals. The availability of many proof assistants and automatic proof procedures [?] in HOL which helps the HOL user in automatic proof in simple goals. User interaction is required in interactive theorem verification with HOL proof editor and guides the prover using the necessary tactics until the last step of the proof. In HOL, some part of proof steps are automatically solved while rest part of proof require signification user interaction.

2.1.7 HOL Notations

We have provided some frequently functions and symbols used in HOL4 also provided its mathematical interpretations in Table 2.1, we will use these symbols and functions in our thesis are these all taken from existing HOL4 theories.

HOL Symbol	Standard Symbol	Meaning
\wedge	<i>and</i>	Logical <i>and</i>
\vee	<i>or</i>	Logical <i>or</i>
\neg	<i>not</i>	Logical <i>negation</i>
$::$	<i>cons</i>	Adds a new element to a list
(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
$\lambda x.t$	$\lambda x.t$	Function that maps x to $t(x)$
$\{x P(x)\}$	$\{x P(x)\}$	Set of all x such that $P(x)$
num	$\{0, 1, 2, \dots\}$	Positive Integers data type
real	All Real numbers	Real data type
suc n	$n + 1$	Successor of a <i>num</i>

Table 2.1: HOL Symbols and Functions

2.2 Probability Space and Random Variables in HOL4

A *measure space* is defined as mapping from sample space to a real line, let Ω is a sample space set and Σ represents a σ -algebra of subsets of Ω subsets are *measurable sets* of [10], and μ is a *measure* of domain Σ . A *probability space* is a measure space (Ω, Σ, P) such that the measure, referred to as the probability and denoted by P , of the sample space is 1. we can model *measure space* as a triple (Ω, Σ, μ) .

Measure theory in higher-order-logic theorem prover was modelled by Hurd [22], he define a measure space as a pair (Σ, μ) , whereas the sample space on which this pair is defined is implicitly implied from the higher-order-logic definitions to be equal to the universal set of the appropriate data-type. We have Build our work using Hurd's formalization of probability theory, Hurd's defined probability space as a pair $(\mathcal{E}, \mathbb{P})$, where the set \mathcal{E} is

the domain of \mathbb{P} , set \mathcal{E} is subset of \mathbb{B}^∞ infinite Boolean sequences. \mathbb{P} and \mathcal{E} are defined in HOL using the Carathéodory's extension theorem, Carathéodory's extension theorem ensures that \mathcal{E} is a σ -algebra: closed under complements and countable unions.

The core concepts in probabilistic analysis is to represent randomness of system using random variable, random variable is basically a probabilistic function and thus we can model it as a deterministic function in higher-order logic as \mathbb{B}^∞ is the argument of that deterministic function. These deterministic functions using the result of popping the top-most bit in the infinite Boolean sequence and make random choices and may use as many as random bits of infinite boolean sequence needed during their computation. When the computation terminate, they return result along with the remaining un used part of the infinite Boolean sequence to be used in future by other programs. Thus, a random variable which takes a parameter of α type and returns values of type β can be represented in HOL4 by a function of type:

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

Consider example of a Bernoulli random variable that returns 0 or 1 having probability $\frac{1}{2}$ in both cases. Bernoulli random variable can be formalized in HOL theorem prover as follows:

$$\forall s. \text{ bit } s = \text{if shd } s \text{ then } 1 \text{ else } 0, \text{ stl } s,$$

where `stl` and `shd` functions are the sequence equivalents of the list operations '*tail*' and '*head*', respectively. The function `bit` accepts (domain of function) the infinite Boolean sequence s and returns (range of function) a pair. The first element of the returned pair is either 1 or 0 a random number, which depends upon the top most element of infinite Boolean sequence s , while the second element of the pair is the tail of the sequence which the unused portion of the infinite Boolean sequence.

We can also express the higher-order-logic functions for probabilistic algorithms state-transforming monad which is more general, where the states of the state-transforming monad are the infinite Boolean sequences.

```

 $\forall a s. \text{ unit } a s = (a, s)$ 
 $\forall f g s. \text{ bind } f g s = g (\text{fst } (f s)) (\text{snd } (f s)).$ 

```

The HOL4 functions `snd` and `fst` return the second and first components of their argument, which is respectively a pair. The operator `unit` is used for lifting values to the monad, and the `bind` is the monadic analogue of function application. All monad laws hold for this definition and the benefit of using monadic notation is that its we can write functions without mentioning the boolean sequence that is passed around explicitly, e.g.,

```

bit_monad = bind sdest ( $\lambda b. \text{ if } b \text{ then unit } 1 \text{ else unit } 0$ )

```

represents function `bit` in monadic notation. Function `sdest` gives the head and tail of a sequence as a pair $(shd\ s, stl\ s)$ and $(\lambda x. fx)$ denotes the lambda-abstraction function that maps its argument x to $f(x)$.

Once random variables are formalized, we can use the formalized probability theory infrastructure to reason about their probabilistic properties. For example, the following PMF property can be verified for the function `bit` using the HOL4 theorem prover.

```

 $\vdash \mathbb{P} \{s \mid \text{fst } (\text{bit } s) = 1\} = \frac{1}{2},$ 

```

where $\{x \mid C(x)\}$ represents a set of all x that satisfy the condition C in HOL4.

The measurability and independence of a probabilistic function are central concepts in probability theory. A property `indep`, called *strong function independence*, is introduced [22] such that if $f \in \text{indep}$, then f will be both measurable and independent. It has also been shown 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 [22].

The above approach has been successfully used to formalize most of the commonly-used random variables and verify them based on their corresponding probability distribution properties. In this thesis, we use the Geometric random variable, which is verified using the following PMF relation [19]:

$$\vdash \forall n p. \quad 0 < p \wedge p \leq 1 \Rightarrow \\ \mathbb{P} \{s \mid \text{fst} (\text{prob_geom } p \ s) = (n + 1)\} = p (1 - p)^n$$

The function `prob_geom` models the $\text{Geometric}(p)$ random variable with probability of success p . It returns the number of Bernoulli trials needed to get one success and thus cannot return 0. This is why we have the term $n + 1$ in the above theorem as n can be any positive integer $\{0, 1, 2, 3 \dots\}$. Similarly, the probability p represents the probability of success of a Bernoulli trial and thus needs to be greater than 0 for this theorem to be true as has been specified in the precondition.

Expectation theory plays a vital role in the domain of probabilistic analysis as it is a lot easier to judge performance issues based on the average value of a random variable, which is a single number, rather than its distribution function. The above-mentioned probability theory formalization has also been extended to formally model the expectation for a discrete random variable, which attains values in the positive integers only, as follows.

$$\vdash \forall X. \quad \text{expect } X = \text{suminf } (\lambda n. \quad n \mathbb{P} \{s \mid \text{fst} (X \ s) = n\})$$

The function `suminf` represents the HOL formalization of the infinite summation of a real sequence [13]. The above definition can be used to verify the average values of most of the commonly used discrete random variables [19]. For example, the expectation of the Geometric random variable can be verified as the following theorem:

$$\vdash \forall p. \quad 0 < p \wedge p \leq 1 \Rightarrow \text{expect } (\lambda s. \quad \text{prob_geom } p \ s) = \frac{1}{p}$$

In order to verify the correctness of the formal definition of expectation and facilitate reasoning about expectation properties in probabilistic systems, many widely used expectation properties have been formally verified [19]. Namely being the linearity of expectation, Markov and Chebyshev's inequalities, variance and linearity of variance. For example, the linearity of expectation, which will be utilized in this thesis, has been verified as the following theorem.

$$\vdash \forall X Y. X \in \text{indep_fn} \wedge Y \in \text{indep_fn} \wedge$$

$$\text{summable}(\lambda n. \ n \ \mathbb{P}\{s \mid \text{fst}(X \ s) = n\}) \wedge$$

$$\text{summable}(\lambda n. \ n \ \mathbb{P}\{s \mid \text{fst}(Y \ s) = n\})$$

$$\Rightarrow (\text{expec} (\text{sum_two_rv} \ X \ Y) = \text{expec} \ X + \text{expec} \ Y)$$

where `summable` accepts a *real* sequence and returns *True* if the infinite summation of this sequence is convergent (i.e., `summable M = $\exists x. \lim_{k \rightarrow \infty} (\sum_{n=0}^k M(n)) = x$`). The function `sum_two_rv` models the summation of two independent random variables.

Chapter 3

HOL4 Formalization of Negative Binomial Random Variable

This chapter presents the higher-order logic formalization of the Negative binomial random variable in HOL4, In section 3.1 we present formal specification of negative binomial random variable using geometric random variable, Then we prove the correctness of our specification by formally verifying the probability mass function and expectation of negative binomial random variable in section 3.2 of this thesis.

3.1 Formal Specification of the Negative Binomial Random Variable

Negative Binomial random variable is discrete random variable, i.e., it can acquire finite or countably infinite unique values. It basically provides the number of successes in a sequence of Bernoulli trials before a specified (non-random) number of failures, n , occur. Negative binomial distribution is quite frequently used to model many real-world phenomena. Examples include, software reliability, accident proneness, insurance outcomes and disease prevalence in medical sciences.

In this section, we present a formal specification of the Negative Binomial random variable in higher-order logic. Negative Binomial random variable can be formalized as the sum of geometric random variables, where each geometric random variable returns the index of the first failure in a sequence of Bernoulli trials or in other words the number of

successes achieved before the first failure. Thus, the Negative Binomial random variable Y can be modeled by considering the sum of n Geometric random variables G :

$$Y(n, p) = \sum_{k=0}^n G_k(1 - p) \tag{3.1}$$

where p is the probability of success of a Bernoulli trial. In order to formalize this expression in higher-order logic, we proceed by building upon the formalized probability theory fundamentals described in Section 2. The first step is to define a list of Geometric random variables recursively as follows:

Definition 1:

$$\begin{aligned} &\vdash \forall p. (\text{geo_ram_list } 0 \text{ } p = []) \wedge \\ &\quad (\text{geo_ram_list } (\text{suc } n) \text{ } p = \\ &\quad \quad (\text{prob_geom } p) :: (\text{geo_ram_list } n \text{ } p)) \end{aligned}$$

The function `geo_ram_list` has the data type $(num \rightarrow real \rightarrow ((real \rightarrow B^\infty \rightarrow num \times B^\infty)list))$ as it accepts a natural number n and a real number p , and returns a list of n Geometric random variables all having the success probability p . The operator `::` represents the list operator *cons* and `[]` represents the empty list in HOL4. The function `prob_geometric` models the Geometric random variable [22] using the approach outlined in Section 2.

In order to sum these Geometric random variables, we formalized the following function

Definition 2:

$$\begin{aligned} &\vdash \text{sum_rv_lst } [] = \text{unit } 0) \wedge \\ &\forall h \text{ } t. \text{ sum_rv_lst } (h :: t) = \\ &\quad \text{bind } h \ (\lambda a. \text{ bind } (\text{sum_rv_lst } t) \ (\lambda b . \text{unit } (a + b))) \end{aligned}$$

The function `sum_rv_lst`, accepts a list of random variables and returns their sum as a single random variable. It accesses the infinite Boolean sequence using the monadic operators `unit` and `bind`, which are described in Section 2. The use of these operators

ensure strong function independence for the function `sum_rv_lst`, if all random variables in the given list preserve it. Thus, the summation would be performed over independent random variables inherently.

Finally, the above two functions can be used to model the Negative Binomial random variable as follows:

Definition 3:

$$\vdash \forall n p. (\text{prob_neg_bino } n \ p) = \text{sum_rv_lst } (\text{geo_ram_lst } n \ (1-p))$$

The function `prob_neg_binomial` accepts the number of failures n and the probability of success p of the Bernoulli trials and returns the sum of n independent Geometric random variables with success probability $1 - p$. By choosing the $1 - p$ as the success probability of the individual Geometric random variables, we ensure that they are counting the number of successes before reaching the first failure. This completes our higher-order-logic formalization of the Negative Binomial random variable.

3.2 Formal Verification of the Negative Binomial Random Variable

In this section, we verify the correctness of the proposed specification of the Negative Binomial random variable using its probability mass function (PMF) and expectation properties. We present the major lemmas and theorems regarding this verification in this thesis and more details can be found in [40].

3.2.1 Probability Mass Function

The PMF relation of the Negative Binomial random variable can be expressed as the following higher-order-logic goal in HOL4.

Theorem 1:

$$\vdash \forall x \ n \ p. 0 \leq p \wedge p < 1 \implies$$

$$\mathbb{P} \{s \mid \text{fst}(\text{prob_neg_bino } \text{suc } n \ p \ s) = \text{suc } x\} = \binom{x}{n} p^{\text{suc } n} (1-p)^{x-n}$$

The assumption ensures that the success probability p lies in the interval $[0, 1)$. The case of $p = 1$ has been ignored because in this case we cannot have n failures and thus it makes the whole theorem invalid. Similarly, the theorem is verified for the case when there is at least 1 failure by using $\text{suc } n$ as the argument of the Negative Binomial number. Again, the reason being that there is no point in verifying the PMF for the case when there is no failure. Moreover, the value of the Negative Binomial random variable can never be 0 and thus we have use $\text{suc } x$ in the above theorem. To prove this statement, we verified the following relationship in HOL4

$$\begin{aligned} \vdash \forall x \ n \ p. 0 \leq p \wedge p < 1 \implies \\ \mathbb{P}\{s \mid \text{fst}(\text{prob_neg_bino } (\text{suc } n) \ p \ s) = 0\} = 0 \end{aligned} \tag{3.2}$$

using induction on variable n and using the formally verified PMF relationship of the Geometric random variable, given in Section 2. Now, we proceed with the verification of Theorem 1 by performing induction on the variable n . The base case generates the following subgoal after simplification from Definitions 1-4:

$$\begin{aligned} \vdash \forall x \ p. 0 \leq p \wedge p < 1 \implies \\ \mathbb{P}\{s \mid \text{fst}(\text{prob_geometric_p } p \ s) = \text{suc } x\} = p(1-p)^x \end{aligned} \tag{3.3}$$

which is simply the PMF of the Geometric random variable and thus can be discharged easily. This also shows that the Geometric random variable is a special case of the Negative Binomial random variable with the complementary probability of success when we are interested in finding the number of successes to get the first failure.

Next, consider the subgoal generated by the step case after the induction on variable n in Theorem 1.

$$\begin{aligned}
& \vdash \forall x p. 0 \leq p \wedge p < 1 \wedge (\forall x p. 0 \leq p \wedge p < 1 \Rightarrow \mathbb{P}\{s \mid \text{fst}(\text{prob_neg_bino}(\text{suc } n) p s) = \text{suc } x\} = \\
& \binom{x}{n} p^{\text{suc } n} (1-p)^{x-n} \implies \mathbb{P}\{s \mid \text{fst}(\text{prob_geometric_p } p s) \\
& + \text{fst}(\text{prob_neg_bino}(\text{suc } n) p s) = \text{suc } x\} = \binom{x}{\text{suc } n} p^{\text{suc}(\text{suc } n)} (1-p)^{x-\text{suc } n}
\end{aligned} \tag{3.4}$$

We proceed to verify it by first verifying the following useful probability law

lemma 1:

$$\begin{aligned}
& \vdash \forall x p. \mathbb{P}\{s \mid \text{fst}(\text{prob_geometric_p } p s) + \\
& \quad \text{fst}(\text{prob_neg_bino}(\text{suc } n) p s) = \text{suc } x\} = \\
& \quad \mathbb{P}(\bigcup_{i=0}^{\text{suc } x} (\lambda i. \{s \mid (\text{fst}(\text{prob_geometric_p } p s) = i) \wedge \\
& \quad (\text{fst}(\text{prob_neg_bino}(\text{suc } n) p s) = \text{suc } x - i)\}))
\end{aligned}$$

which allows us to rewrite the subgoal of Equation as follows:

$$\begin{aligned}
& \mathbb{P}(\bigcup_{i=0}^{\text{suc } x} (\lambda i. \{s \mid (\text{fst}(\text{prob_geometric_p } p s) = i) \wedge (\text{fst}(\text{prob_neg_bino} \\
& \quad (\text{suc } n) p s) = \text{suc } x - i)\})) = \binom{x}{\text{suc } n} p^{\text{suc}(\text{suc } n)} (1-p)^{x-\text{suc } n}
\end{aligned} \tag{3.5}$$

Next, we split the set on the left-hand-side of the above subgoal into a union of three sets to get the following subgoal

$$\begin{aligned}
& \mathbb{P}\{s \mid (\text{fst}(\text{prob_geometric_p } p s) = 0) \wedge (\text{fst}(\text{prob_neg_bino}(\text{suc } n) p s) = n)\} \cup \\
& \mathbb{P}(\bigcup_{i=0}^{x-n-1} (\lambda i. \{s \mid (\text{fst}(\text{prob_geometric_p } p s) = \text{suc } i) \wedge (\text{fst}(\text{prob_neg_bino}(\text{suc } n) p s) \\
& = x - i)\})) \cup \mathbb{P}(\bigcup_{i=x-n+1}^{\text{suc } x} (\lambda i. \{s \mid (\text{fst}(\text{prob_geometric_p } p s) = \text{suc } i) \\
& \wedge (\text{fst}(\text{prob_neg_bino}(\text{suc } n) p s) = \text{suc } x - i)\}))
\end{aligned} \tag{3.6}$$

It is important to note here that the above simplification could only be done under the condition that all the events involved are measurable. We verified the measurability of the events associated with the Negative Binomial random variable by using the fact that it accesses the infinite Boolean sequence using the monadic operators `bind` and `unit` only, as illustrated in Section 2. We utilized the measurability theorem for the Geometric random variable from [19] in our proof. Utilizing the same results, we utilized the probability laws of additivity ($Pr(A \cup B) = Pr(A) + Pr(B)$) and independence ($Pr(A \cap B) = Pr(A) * Pr(B)$) to simplify the left-hand-side of our subgoal further as follows:

$$\begin{aligned}
& \mathbb{P}\{s | (\text{fst}(\text{prob_geometric_p p s}) = 0)\} * \mathbb{P}\{s | (\text{fst}(\text{prob_neg_bino}(\text{suc } n) \text{ p s}) = n)\} + \\
& \left(\sum_{i=0}^{x-n-1} (\lambda i. \mathbb{P}\{s | (\text{fst}(\text{prob_geometric_p s}) = \text{suc } i)\} * \mathbb{P}\{s | (\text{fst}(\text{prob_neg_bino}(\text{suc } n) \text{ p s}) \right. \\
& = x - i)\}) \left. \right) + \left(\sum_{i=x-n+1}^{\text{suc } x} (\lambda i. \mathbb{P}\{s | (\text{fst}(\text{prob_geometric_p p s}) = \text{suc } i)\} \right. \\
& \left. * \mathbb{P}\{s | (\text{fst}(\text{prob_neg_bino}(\text{suc } n) \text{ p s}) = \text{suc } x - i)\} \right)
\end{aligned} \tag{3.7}$$

Now, the first probability term is 0 from Equation (3.2). Moreover, we verified that the right most probability term is also 0 based on the behavior of the Negative Binomial random variable as $(\text{As } \forall i. x - n + 1 \leq i \leq \text{suc } x \implies \text{suc } x - i < \text{suc } n)$. We formally verified the following result to represent this characteristic in HOL4:

lemma 2:

$$\begin{aligned}
& \vdash \forall x n p i. (0 \leq p) \wedge (p < 1) \wedge (\text{suc } x - i < \text{suc } n) \implies \\
& \quad \mathbb{P}\{s | (\text{fst}(\text{prob_neg_bino}(\text{suc } n) \text{ p s}) = \text{suc } x - i) = 0\}
\end{aligned}$$

Using the result on the left-hand-side of our subgoal, we are left with the following subgoal:

$$\begin{aligned}
& 0 \leq p \wedge p < 1 \wedge (\forall x p. 0 \leq p \wedge p < 1 \Rightarrow \mathbb{P}\{s \mid \text{fst}(\text{prob_neg_bino}(\text{suc } n) p s) = \text{suc } x\} = \binom{x}{n} p^{\text{suc } n} (1 - p)^{x - \text{suc } n}) \\
& \implies \sum_{i=0}^{x-n-1} (\lambda i. p(1-p)^i * \mathbb{P}\{\text{fst}(\text{prob_neg_bino}(\text{suc } n) p s) = x - i\}) \\
& = \binom{x}{\text{suc } n} p^{\text{suc}(\text{suc } n)} (1-p)^{x-\text{suc } n}
\end{aligned} \tag{3.8}$$

Rewriting with the induction assumption now simplifies the subgoal as follows:

$$\begin{aligned}
& \sum_{i=0}^{x-n-1} (\lambda i. p(1-p)^i * \binom{x-i-1}{n} p^{n+1} (1-p)^{x-(i+\text{suc } n)}) \\
& = \binom{x}{\text{suc } n} p^{\text{suc}(\text{suc } n)} (1-p)^{x-\text{suc } n}
\end{aligned} \tag{3.9}$$

The probability terms have been thus removed from subgoals and the remaining proof is based on real theoretic reasoning. By rearrangement and canceling the terms we can simplify the above expression to reach the following subgoal:

$$\sum_{i=0}^{x-(n+1)} (\lambda i. \binom{x-i-1}{n}) = \binom{x}{\text{suc } n} \tag{3.10}$$

To verify this relationship, we proceed by applying induction on the variable x . The base case is verified based on the definition of the binomial function and some simple arithmetic reasoning. We get the following subgoal for the step case:

$$\sum_{i=0}^{x-n} (\lambda i. \binom{x-i}{n}) = \binom{x+1}{\text{suc } n} \tag{3.11}$$

Now, using properties of real summation along with some simple arithmetic reasoning, we reach the following subgoal:

$$\sum_{i=0}^{x-n} (\lambda i. \binom{x-i-1}{n-1}) + \sum_{i=0}^{x-(n+1)} (\lambda i. \binom{x-i-1}{n}) = \binom{x+1}{\text{suc } n} \quad (3.12)$$

The assumption of the step case allows us to simplify the subgoal to the following expression:

$$\binom{x}{n} + \binom{x}{\text{suc } n} = \binom{\text{suc } x}{\text{suc } n} \quad (3.13)$$

The above subgoal is the variant of the famous Pascal's triangle property [9, 12], which we formally verified in HOL4 as part of the reported formalization to discharge the final subgoal. This also concludes the formal verification of Theorem 1.

3.2.2 Expectation

Based on Definition 3, the Negative Binomial random variable is essentially a sum of n Geometric random variables. Therefore, its expectation can be verified based on the Linearity of expectation property, given in Section 2, and the expectation of the Geometric random variable. The expectation property of the Negative Binomial random variable can be expressed as the following higher-order-logic theorem.

Theorem 2. *Expectation of Negative Binomial Random Variable*

$$\vdash \forall n p. \ 0 \leq p \wedge p < 1 \Rightarrow \\ \text{expec } (\lambda s. \ \text{prob_neg_bino } (\text{suc } n) \ p \ s) = \frac{\text{suc } n}{p}$$

The first step is to simplify the above theorem using the Definition 3

$$\text{expec}(\lambda s. \text{sum_rv_lst}(\text{geo_ram_lst } n \ p \ s)) = \frac{\text{suc } n}{p} \quad (3.14)$$

Next, we use induction on variable n and the subgoal for the base case for $n = 0$ is as follows

$$\text{expec}(\lambda s.\text{sum_rv_lst}(\text{geo_ram_lst } 0 \text{ p } s)) = \frac{1}{p} \quad (3.15)$$

which can be verified based on Definitions 1 and 2 and the formal definition of the function `expec` [19]. The step case for our induction step generates the following subgoal after simplification from Definitions 1 and 2.

$$\begin{aligned} \forall n p. (0 \leq p) \wedge (p < 1) \wedge (\forall n p. 0 \leq p \wedge p < 1 \Rightarrow \text{expec}(\lambda s.\text{prob_neg_bino}(\text{suc } n) \text{ p } s) = \frac{\text{suc } n}{p}) \\ \implies \text{expec}(\lambda s.\text{sum_rv_lst}(\text{geo_ram_lst } n \text{ p } s) + (\text{prob_geometric_p } p \text{ } s)) = \frac{\text{suc}(\text{suc } n)}{p} \end{aligned} \quad (3.16)$$

Now, using the linearity of expectation property, given in Section 2, we can further simplify our subgoal as follows:

$$\begin{aligned} \text{expec}(\lambda s.\text{sum_rv_lst}(\text{geo_ram_lst } n \text{ p } s)) \\ + \text{expec}(\lambda s.(\text{prob_geometric_p } p \text{ } s)) = \frac{\text{suc}(\text{suc } n)}{p} \end{aligned} \quad (3.17)$$

The expectation of the Geometric random variable is $\frac{1}{p}$, as given in Lemma 1, and the expectation of first variable in the subgoal is given in the induction assumption, as shown in subgoal 3.16. This completes the proof of Theorem 2 and thus verifies that the expectation of the Negative Binomial random variable, given in Definition 1, is equivalent to its theoretical expectation relationship verified using paper-and-pencil proof methods.

The formal verification of the PMF and expectation properties of our formal definition of the Negative Binomial random variable ensures the correctness of our formalization. Moreover, these proofs can be used to reason about any probabilistic or statistical property associated with the Negative Binomial random variable in the sound environment of HOL4 since any probability or statistical property of a random variable can be expressed in terms of its PMF and expectation, respectively. This aspect, in turn, facilitates formal analysis

of systems involving Negative Binomial random variable as will be illustrated in the next section. Besides these major benefits, the formalization led to the formal verification of many classical mathematical results, such as the Pascal's triangle property and some useful probability laws, for the first time and these results can be very useful in other formal reasoning based research. Our proof script is available at [40] for download. It took 600 man-hours and is composed of approximately 2500 lines of HOL code.

Chapter 4

Application

4.1 Software Reliability Application

Due to the extensive usage of software in safety-critical domains, like transportation and financial institutions, its correctness has become imperative these days. One way to ensure software correctness is to check its outputs for all possible inputs exhaustively. This approach may be feasible for small programs but for most of the real-world software the possible combinations of inputs is so large that exhaustive testing is not feasible given the available computational resources. Thus, testing under a subset of all the possible input combinations is practically done. This gives rise to a perpetual question faced by all software developers and that is about the amount of testing required as testing requires significant amount of resources and time but releasing an unreliable software is also very costly.

Negative Binomial distribution plays a vital role in this context [39]. It can be used to model the total of test cases required to catch n failures given the failure rate p of a program. This way, we can predict the actual amount of resources required for testing a software with a given failure rate. The particular properties of interest are the probability that the test cases would be less than a given number, which corresponds to the available resources and the average number of test cases for catching a specific number of errors.

The reported formalization of Negative Binomial random variable can be used to reason about the above mentioned interesting software reliability characteristics. The first step in this regard would be to model the total number of test cases required to catch n failures for a software with a failure rate p as follows:

Definition 4:

$$\vdash \forall n p. \text{ num_test_cases } n p = \text{ prob_neg_bino } np$$

The first property of interest is the probability of the number of test cases being less than some given value. It can be formalized as follows:

$$\vdash \forall x n p. 0 \leq p \wedge p < 1 \implies$$

$$\mathbb{P} \{s \mid \text{fst}(\text{num_test_cases } \text{succ } n p s) = \text{succ } x\} = \binom{x}{n} p^{\text{succ } n} (1 - p)^{x-n}$$

We proved this property by using Theorem 1, the formally verified standard probability theory result $Pr(X \leq x) = \sum_{i=0}^x Pr(X = i)$ and the fact that the individual equality based events are mutually exclusive of one another.

The next property of interest in the context of reliability analysis of software is the average number of test runs. The corresponding theorem can be stated as follows:

Expectation of number of test cases

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

$$\text{expec } (\lambda s. \text{ num_test_cases } (\text{succ } n) p s) = \frac{\text{succ } n}{p}$$

and can be verified in a straightforward way using the formal verified expectation of the Negative Binomial random variable expression, given in Theorem 2. It can be clearly observed from the expectation relation that the lower the failure rate of the given software is, the more test runs are required to catch n errors.

4.2 Concluding Remarks

In this chapter, we presented the real-world applications of our proposed framework. In the first application, we presented the application of negative binomial in software reliability we formalized expected of test cases required for checking reliability of software and also formalized probability of the number of test cases being less than some given value. The

above theorems have been verified within the sound core of a theorem prover and thus provide accurate results, which is not possible in any other computer based analysis method. Moreover the theorems are universally quantified for all the variables involved and are not restricted to some specific values. The interactive reasoning required to verify the above theorems was merely just a few lines and thus far less than what was required to verify Theorems 1 and 2. This is the main strength and usefulness of the formalization presented in this thesis, i.e, our results can be built upon to formally reason about real-world problems involving the Negative Binomial random variable.

Our formal analysis of above mentioned case studies is fairly general and can be extended to other safety-critical fractional order systems. For example, our analysis can be utilized to conduct formal analysis of a such as transportation [26] (accident proneness [38, 24, 32]), medicine (disease prevalence in a population [34], [35]), agriculture [3] and software (reliability assessment [39]). . Another interesting application of our proposed frame work is the prediction of production yield of Integrated circuits, In production of Integrated cricuits (I.C.s) esitimation of yield of working devices is economically important. Our framework can be used for formally analyzing of defect distribution. Number of defects in integrated circuits is a random variable which can be model as negative binomial random variable. we can estimate the expected number of defects in integrated circuits using expectation of negative binomial distribution.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis presents a formalization the Negative Binomial random variable along with the verification of its PMF and expectation properties in the higher-order logic theorem prover HOL4. This formalization can be used to formally reason about systems that involve the Negative Binomial distributions. The Negative Binomial random variable is widely used in many safety and financial critical applications including transportation, medical, software, agriculture, quality control and insurance. Thus the precise nature of the proposed analysis can be very useful in all these domains. We illustrated the applicability of the formalized Negative Binomial random variable in analyzing real-world problems by analyzing a software reliability problem. A simple example was picked to facilitate understanding and the ideas are generic enough to be extended to more complicated scenarios and systems as well.

5.2 Future Work

The Negative Binomial random variable provides very interesting relationships to many other random variables including Beta, Gamma and Poisson. To the best of our knowledge, the higher-order-logic formalization of these random variables is not available and we plan to utilize our formalization of the Negative Binomial random variable to formalize and verify these random variables. This would further extend the scope of the higher-order-logic theorem proving based probabilistic analysis framework to systems involving these new random variables.

References

- [1] The HOL System Description. <http://hol.sourceforge.net/documentation.html>, 2011.
- [2] B. Akbarpour and L. C. Paulson. Metitarski: An Automatic Prover for the Elementary Functions. In *AISC/MKM/CalcuIemus*, pages 217–231, 2008.
- [3] NEAL ALEXANDER. Sptial Modelling of Individual Level Parasite cout using Negative Binomial distribution. *Biostatistics (2000)*, pages 453–463, 2000.
- [4] C. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen. Model Checking Algorithms for Continuous time Markov Chains. *IEEE Trans. on Software Engineering*, 29(4):524–541, 2003.
- [5] C. Baier and J. Katoen. *Principles of Model Checking*. MIT Press, 2008.
- [6] P.P. Boca, J.P. Bowen, and J.I. Siddiqi. *Formal Methods: State of the Art and New Directions*. Springer, 2009.
- [7] Ch. Li. Documentation Based Testing Tool for Software Module Reliability Estimation. Master’s thesis, Master of Engineering Thesis,McMaster University, 1986.
- [8] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD Thesis, Stanford University, Stanford, USA, 1997.
- [9] John McBrewster Frederic P. Miller, Agnes F. Vandome. *Binomial Coefficient*. VDM Verlag Dr. Mueller, 2010.
- [10] J. Galambos. *Advanced Probability Theory*. Marcel Dekker Inc., 1995.
- [11] William G.Faris. *Lectures on Elementary Probability*. 2002.
- [12] Catherine A . Gorini. *Master Math: Probability*. Course Technology PTR, 2011.

- [13] J. Harrison. *Theorem Proving with the Real Numbers*. Springer-Verlag, 1998.
- [14] J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.
- [15] J. Harrison. A List of Theorem Provers. <http://www.cl.cam.ac.uk/users/jrh/ar.html>, 2011.
- [16] J. Harrison, K. Slind, and R. Arthan. HOL. In *The Seventeen Provers of the World*, volume 3600 of *LNCS*, pages 11–19. Springer, 2006.
- [17] O. Hasan, N. Abbasi, B. Akbarpour, S. Tahar, and R. Akbarpour. Formal Reasoning about Expectation Properties for Continuous Random Variables. In *Formal Methods*, volume 5850 of *LNCS*, pages 435–450. Springer, 2009.
- [18] O. Hasan and S. Tahar. Formalization of the Continuous Probability Distributions. In *Conference on Automated Deduction*, volume 4603 of *LNAI*, pages 3–18. Springer, 2007.
- [19] O. Hasan and S. Tahar. Using Theorem Proving to Verify Expectation and Variance for Discrete Random Variables. *Journal of Automated Reasoning*, 41(3–4):295–323, 2008.
- [20] 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.
- [21] 17. J. H?lzl and A. Heller. Three Chapters of Measure Theory in Isabelle/HOL. In *Interactive Theorem Proving*, volume 6172 of *LNCS*, pages 135–151. Springer, 2011.
- [22] J. Hurd. *Formal Verification of Probabilistic Algorithms*. PhD Thesis, University of Cambridge, Cambridge, UK, 2002.

- [23] B. Jeannet, P.D. Argenio, and K. Larsen. Rapture: A Tool for Verifying Markov Decision Processes. In *Tools Day, 13th Int. Conf. Concurrency Theory*, 2002.
- [24] K B KUNE. Accident liability. *British Journal of Industrial Medicine*, pages 336–340, 1985.
- [25] 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.
- [26] Wing-gung Wong Kwok suen, Wing-tat Hung. An algorithm for assessing the risk of traffic accident. *Journal of Safety Research*, 33:387–410, 2002.
- [27] D.J.C. MacKay. Introduction to Monte Carlo methods. In *Learning in Graphical Models, NATO Science Series*, pages 175–204. Kluwer Academic Press, 1998.
- [28] T. Mhamdi, O. Hasan, and S. Tahar. On the Formalization of the Lebesgue Integration Theory in HOL. In *Interactive Theorem Proving*, volume 6172 of *LNCS*, pages 387–402. Springer, 2011.
- [29] R. Milner. A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.
- [30] D. Parker. *Implementation of Symbolic Model Checking for Probabilistic System*. PhD Thesis, University of Birmingham, UK, 2001.
- [31] L. C. Paulson. *ML for the Working Programmer*. Cambridge University Press, 1996.
- [32] M. Poch and F. Mannerin. Negative Binomial Analysis of Intersection-Accident Frequencies. *Journal of Transportation Engineering*, 122 (2):105–113, 1996.
- [33] PRISM. www.cs.bham.ac.uk/~dxp/prism, 2007.
- [34] A. Pritchard and M. Tebbs. Estimating disease prevalence using inverse binomial pooled testing. *Journal Agriculture Biology and Environment*, pages 70–87, 2010.

- [35] A. Pritchard and M. Tebbs. Bayesian Inference For Disease Prevalence Using Negative Binomial Group Testing. *Biometrical Journal*, 53:40–56, 2011.
- [36] J. Rutten, M. Kwiatkowska, G. Normal, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.
- [37] K. Sen, M. Viswanathan, and G. Agha. VESTA: A Statistical Model-Checker and Analyzer for Probabilistic Systems. In *IEEE International Conference on the Quantitative Evaluation of Systems*, pages 251– 252, Washington, DC, USA, 2005.
- [38] LEROY J. SIMON. An Introduction To The Negative Binomial Distribution And Its Applications. page 8, 1962.
- [39] B. Singh, D.L. Parnas, and R Viveros. *Estimating Software Reliability Using Inverse Sampling*. CRL Report 351. McMaster University, Hamilton Ontario, Canada, 1997.
- [40] M. Wisal. Formal Verification of Negative Binomial Distribution - in HOL4 Proof Scrip. Technical Report, NUST, H-12, Islamabad, january 2012; <http://www.save.seecs.nust.edu.pk/students/wisal/nb.html>.