A Stochastic Theory of Black-Box Software Testing

Chapter · June 2006

DOI: 10.1007/11780274_30 · Source: DBLP

1 author:

Karl Meinke
KTH Royal Institute of Technology
74 PUBLICATIONS **647** CITATIONS

SEE PROFILE

A Stochastic Theory of Black-Box Software Testing

Karl Meinke

School of Computer Science and Communication,
Royal Institute of Technology, 100-44 Stockholm, Sweden

Abstract. We introduce a mathematical framework for black-box software testing of functional correctness, based on concepts from stochastic process theory. This framework supports the analysis of two important aspects of testing, namely: (i) coverage, probabilistic correctness and reliability modelling, and (ii) test case generation. Our model corrects some technical flaws found in previous models of probabilistic correctness found in the literature. It also provides insight into the design of new testing strategies, which can be more efficient than random testing.

1 Introduction

Structural or glass-box testing of the functional correctness of software systems has been theoretically studied at least since the early 1950s (see for example [Moore 1956]). Although many useful structural test strategies have been developed, (see for example the survey [Lee and Yannakakis 1996]), theoretical studies clearly indicate the limitations of structural testing. In particular the complexity of structural testing techniques often grows exponentially with the size of the system.

To overcome this limitation, software engineers use black-box testing methods for large systems (see e.g. [Beizer 1995]). We will assume that a functional requirement on a system S can be modelled as a pair (p, q) consisting of precondition p on the input data and a postcondition q on the output data of S. The simplest strategy for black-box testing is random testing, in which input vectors satisfying p are randomly generated, and the output of each execution is compared with the postcondition q as a test oracle.

Efforts to improve on random testing, for example by careful manual design of test cases based on system knowledge and programming expertise, face the problem of proving their cost effectiveness. In fact to date, a general theoretical framework in which different black-box test strategies can be compared seems to be lacking. Constructing such a theory is a challenging problem for the theoretician. Not least because when structural features of a system S are hidden, much less remains on which to build a mathematical model. Essentially we have only the pre and postconditions p and q and the semantics or black-box behaviour of S.

K. Futatsugi et al. (Eds.): Goguen Festschrift, LNCS 4060, pp. 578–595, 2006.

In this paper we introduce a mathematical foundation for black-box testing of functional correctness based on the theory of stochastic processes. Our approach is sufficiently general to deal with both: (i) coverage, test termination and reliability modelling, and (ii) efficient test case generation. These two issues seem to be central to any improvement in software testing technology.

A system under test is invisible under the black-box methodology. Thus we can view its output as a source of stochastic behaviour under a sequence of tests. Our approach exploits the analogy between:

(i) a *black-box test* which can be viewed as an input/output measurement made on an unseen and randomly given program, and
(ii) a *random variable* which is a measurable function between two σ-fields.

Thus we can study how different strategies for black-box testing involve different assumptions about the finite-dimensional distributions (FDDs) of such random variables. While exact calculations of probabilities are generally difficult, it seems possible to identify heuristics which simplify and approximate these calculations. These can form the basis for new test strategies and tools.

The organisation of this paper is as follows. In Section 2 we formalise the concept of test success for a program S with respect to a functional correctness requirement $\{p\}S\{q\}$. In Section 3, we introduce the necessary concepts from measure theory and the theory of stochastic processes which are used to formalise probabilistic statements about testing. Our model corrects some technical flaws found in previous models of probabilistic correctness in the literature. In Section 4, we consider the coverage problem and termination criteria for testing. In Section 5 we consider efficient test case generation. Section 6 considers open problems and future research.

2 Logical Foundations of Functional Black-Box Testing

In this section we formalise functional black-box testing within the traditional framework of program correctness. The principle concept to be defined is that of a *successful black-box test* for functional correctness.

To simplify our exposition, we consider requirements specifications and computation over the ordered ring $\mathbb{Z}$ of integers. It should be clear that our approach can be generalised to any countable many-sorted data type signature Σ, (see for example [Loeckx et al. 1996]) and any minimal Σ algebra A.

The first-order language or signature Σ^{ring} for an ordered ring of integers consists of two constant symbols 0, 1, three binary function symbols $+$, $*$, $-$ and two binary relation symbols $=$, $\leq$. The ordered ring $\mathbb{Z}$ of integers is the first-order structure with domain $\mathbb{Z}$ where the constant, function and relation symbols are interpreted by the usual arithmetic constants, functions and relations.

Let X be a set of variables. The set $T(\Sigma^{ring}, X)$ of all *terms* is defined inductively in the usual way, and $T(\Sigma^{ring}) = T(\Sigma^{ring}, \varnothing)$ denotes the subset of all variable free or *ground terms*. If $\alpha : X \to \mathbb{Z}$ is any assignment then

$\overline{\alpha} : T(\Sigma,\ X) \to \mathbb{Z}$ denotes the *term evaluation mapping*. For any ground term t we may write $t_{\mathbb{Z}}$ for $\overline{\alpha}(t)$.

We assume the usual definition of the set $L(\Sigma^{ring},\ X)$ of all *first-order formulas* over Σ^{ring} and X as the smallest set containing all atomic formulas (equations and inequalities) which is closed under the propositional connectives $\wedge$, $\neg$ and the quantifier $\forall$. The expression $\phi \vee \psi$ denotes $\neg(\neg\phi \wedge \neg\psi)$ while $\phi \to \psi$ denotes $\neg\phi \ \vee \ \psi$. The set $L_{\omega_1,\,\omega}(\Sigma^{ring},\ X)$ of all *infinitary first-order formulas* over Σ^{ring} and X extends $L(\Sigma^{ring},\ X)$ with closure under countably infinite conjunctions

$$\bigwedge_{i \in I} \phi_i,$$

for I a countable set. (See e.g. [Barwise 1968].) This infinitary language plays a technical role in Section 3 to translate first-order formulas into probabilistic statements about testing.

We assume the usual definitions of a free variable in a formula or term, and the substitution of a free variable by a term.

Next we recall how first-order formulas are used to define pre and postconditions for a program within the framework of the Floyd-Hoare theory of program correctness. For an overview of this theory see e.g. [de Bakker 1980].

Definition 1. *For any set X of variable symbols. define $X' = \{x' \mid x \in X\}$. A variable $x \in X$ is termed a prevariable while $x' \in X'$ is termed the corresponding postvariable.*

A precondition is a formula $p \in L(\Sigma^{ring},\ X)$ with only prevariables, while a postcondition is a formula $q \in L(\Sigma^{ring},\ X \cup X')$ that may have both pre and postvariables.

Let $\Omega(X)$ be an arbitrary programming language in which each program $\omega \in \Omega(X)$ has an *interface* $i(\omega) = \overline{x}$ where $\overline{x} = x_1,\ \ldots,\ x_k \in X^k$ is a finite sequence of length $k \geq 1$ of integer variables. The interface variables x_i are considered to function as both input and output variables for ω. If ω has the interface $\overline{x}$ we may simply write $\omega[\overline{x}]$. Note that, consistent with black-box testing, we assume no internal structure or syntax for $\Omega(X)$ programs. We will assume semantically that each program $\omega[\overline{x}] \in \Omega(X)$ has a simple *deterministic transformational action* on the initial state of $x_1,\ \ldots,\ x_k$. In the sequel, for any set B, we let $B_\perp = B \cup \{\perp\}$ where $\perp$ denotes an *undefined value*. We let $f : A \to B_\perp$ denote a *partial function* between sets A and B that may be undefined on any value $a \in A$, in which case we write $f(a) = \perp$. If $f(a)$ is defined and equal to $b \in B$ we write $f(a) = b$. We use $[A \to B_\perp]$ to denote the set of all partial functions from A to B.

Definition 2. *Let $\Omega(X)$ be a programming language. By a semantic mapping for $\Omega(X)$ we mean a mapping of programs into partial functions,*

$$[\![\ .\ ]\!] : \bigcup_{k \geq 1} \ [\ \mathbb{Z}^k \to \mathbb{Z}^k_\perp\ ]$$

where for any program $\omega \in \Omega(X)$ *with interface* $i(\omega) = x_1, \ldots, x_k$

$$[\![\omega]\!] : \mathbb{Z}^k \to \mathbb{Z}_\perp^k$$

is a partial recursive function.

Intuitively, for any input $a = a_1, \ldots a_k \in \mathbb{Z}^k$, either ω fails to terminate when the interface variables $x_1, \ldots, x_k$ are initialised to $a_1, \ldots a_k$ respectively, and $[\![\omega]\!](a) = \perp$, or else ω terminates under this initialisation, and $[\![\omega]\!](a) = b_1, \ldots b_k$, where b_i is the state of the interface variable x_i after termination.

Let us collect together the definitions introduced so far to formalise the concepts of test success and failure. Recall the usual satisfaction relation $\mathbb{Z}, \alpha \models \phi$ in $\mathbb{Z}$ for a formula ϕ (finitary or infinitary) under an assignment $\alpha : X \to \mathbb{Z}$ in the Σ^{ring} structure $\mathbb{Z}$.

Definition 3. *Let* $\omega \in \Omega(X)$ *be a program with interface* $i(\omega) = x_1, \ldots, x_k \in X^k$.

(i) A functional specification $\{p\}\omega\{q\}$ *is a triple, where* $p \in L(\Sigma^{ring}, X)$ *is a precondition and* $q \in L(\Sigma^{ring}, X \cup X')$ *is a postcondition.*

(ii) A specification $\{p\}\omega\{q\}$ *is said to be true in* $\mathbb{Z}$ *under assignments* $a : X \to \mathbb{Z}$ *and* $b : X' \to \mathbb{Z}$ *if, and only if,* $\mathbb{Z}, a \models p$ *and if* $[\![\omega]\!](a(x_1), \ldots a(x_k)) = b(x'_1), \ldots, b(x'_k)$ *then* $\mathbb{Z}, a \cup b \models q$. *If* $\{p\}\omega\{q\}$ *is true in* $\mathbb{Z}$ *under* a *and* b *we write*

$$\mathbb{Z}, a \cup b \models \{p\}\omega\{q\}.$$

We say that $\{p\}\omega\{q\}$ *is valid in* $\mathbb{Z}$ *and write* $\mathbb{Z} \models \{p\}\omega\{q\}$, *if, and only if, for every* $a : X \to \mathbb{Z}$ *if there exists* $b : X' \to \mathbb{Z}$ *such that* $[\![\omega]\!](a(x_1), \ldots a(x_k)) = b(x'_1), \ldots, b(x'_k)$ *then* $\mathbb{Z}, a \cup b \models \{p\}\omega\{q\}$.

(iii) Let p *be a precondition and* q *be a postcondition. For any* $a : X \to \mathbb{Z}$ *we say that* ω *fails the test* a *of* $\{p\}\omega\{q\}$ *if, and only if, there exists* $b : X' \to \mathbb{Z}$ *such that* $\mathbb{Z}, a \models p$ *and* $[\![\omega]\!](a(x_1), \ldots a(x_k)) = b(x'_1), \ldots, b(x'_k)$ *and*

$$\mathbb{Z}, a \cup b \not\models q.$$

We say that ω *passes the test* a *of* $\{p\}\omega\{q\}$ *if* ω *does not fail* a.

Intuitively ω fails the test $a : X \to \mathbb{Z}$ of $\{p\}\omega\{q\}$ if a satisfies the precondition p, and ω terminates on the input a but the resulting output assignment $b : X' \to \mathbb{Z}$ does not satisfy the postcondition q. This definition is consistent with the *partial correctness interpretation* of validity for $\{p\}\omega\{q\}$ used in Definition 3.(ii) (c.f. [de Bakker 1980]). Partial correctness (rather than the alternative *total correctness interpretation*) is appropriate if we require a failed test case to produce an observably incorrect value in finite time rather than an unobservable infinite loop. In particular, for this choice of Definitions 3.(ii) and 3.(iii) we have

$$\mathbb{Z} \not\models \{p\}\omega\{q\} \;\Leftrightarrow\; \omega \text{ fails some test } a \text{ of } \{p\}\omega\{q\}.$$

Thus we have formalised program testing as the search for counterexamples to program correctness (under the partial correctness interpretation).

3 A Stochastic Calculus for Program Correctness

Within the framework of program correctness outlined in Section 2, we wish to approach functional black-box testing in a quantitative way. The following question seems central. *Given a specification* $\{p\}\omega\{q\}$ *suppose that* ω *has passed* n *tests* $a_1, \ldots, a_n$ *of* p *and* q*: for any new test* a_{n+1} *what is the probability that* ω *will fail* a_{n+1}*?* Intuitively, we expect this failure probability to monotonically decrease as a function of n. For fixed n we also expect the failure probability to depend on the values chosen for $a_1, \ldots, a_n$. An optimal testing strategy would be to choose a_{n+1} with the maximal probability that ω fails a_{n+1}, for each $n \geq 0$. In this section we will introduce a probabilistic model of testing that can be used to answer this question.

Recall from probability theory the concept of a *σ–algebra* or *σ–field* (Ω, F) of events, where Ω is a non-empty set of elements termed *outcomes* and $F \subseteq \wp(\Omega)$ is a collection of sets, known as *events*, which is closed under countable unions and intersections. (See e.g. [Kallenberg 1997].) Importantly for us, $(A, \wp(A))$ is a σ-field, for any countable set A, including any set of partial recursive functions.

Definition 4. *Let* $\overline{x} = x_1, \ldots, x_k \in X^k$*, for* $k \geq 1$*, be an interface. By a sample space of programs over* $\overline{x}$ *we mean a pair*

$$\mathbf{\Omega}[\overline{x}] = (\ \Omega[\overline{x}],\ eval\ ),$$

where $\Omega[\overline{x}] \subseteq \Omega(X)$ *is a subset of programs* $\omega[\overline{x}]$ *all having the same interface* $\overline{x}$*, and* $eval : \Omega[\overline{x}] \times \mathbb{Z}^k \to \mathbb{Z}^k_\perp$ *is the program evaluation mapping given by*

$$eval(\ \omega,\ a_1,\ \ldots,\ a_k\ ) = [\![\ \omega\ ]\!](\ a_1,\ \ldots,\ a_k\ ).$$

We say that $\Omega[\overline{x}]$ *is extensional if, and only if, for all programs* $\omega,\ \omega' \in \Omega[\overline{x}]$

$$(\ \forall a \in \mathbb{Z}^k\ \ eval(\omega,\ a) = eval(\omega',\ a)\ )\ \ \rightarrow\ \ \omega = \omega'.$$

We consider extensional sample spaces of programs only. It is important for distribution modeling that all programs $\omega \in \Omega[\overline{x}]$ have the same interface $\overline{x}$.

Recall that a *probability measure* $\mathbb{P}$ on σ-field (Ω, F) is a function $\mathbb{P} : F \to [0, 1]$ satisfying: $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$, and for any collection $e_1, e_2, \ldots \in F$ of events which are pairwise disjoint, i.e. $i \neq j \Rightarrow e_i \cap e_j = \emptyset$,

$$\mathbb{P}(\ \bigcup_{i=1}^{\infty} e_i\ ) = \sum_{i=1}^{\infty} \mathbb{P}(\ e_i\ ).$$

The triple $(\Omega, F, \mathbb{P})$ is termed a *probability space*.

Let (Ω, F) and (Ω', F') be σ-fields. A function $f : \Omega \to \Omega'$ is said to be *measurable*, if, and only if, for each event $e \in F'$,

$$f^{-1}(e) \in F.$$

Let $P = (\Omega,\ F,\ \mathbb{P})$ be a probability space. A *random variable* $X : \Omega \to \Omega'$ is a measurable function. If $X : \Omega \to \Omega'$ is any random variable then X induces a probability function $\mathbb{P}_X : F' \to [0,\ 1]$ defined on any $e \in F'$ by

$$\mathbb{P}_X(e) = \mathbb{P}(\ X^{-1}(e)\ ).$$

Then $(\Omega',\ F',\ \mathbb{P}_X)$ is a probability space. Thus one important role of a random variable that we will exploit in Definition 9 is to transfer a probability measure from F-events onto F'-events.

We may take more than one measurement on the outcome of any random experiment. We can consider any finite number or even an infinite number of measurements. This leads us naturally to the important concept of a stochastic process. Let I be any non-empty set. A *stochastic process* S over $(\Omega',\ F')$ is an I-indexed family of random variables $S = \langle S_i : \Omega \to \Omega' \mid i \in I \rangle$. For each $\omega \in \Omega$, the function $S(\omega) : I \to \Omega'$ defined by

$$S(\omega)(i) = S_i(\omega)$$

is termed a *path* of the process S. (See e.g. [Grimmet et al. 1982].)

Definition 5. *Let* $\overline{x} = x_1,\ \ldots,\ x_k \in X^k$*, for* $k \geq 1$*, be an interface and let* $\mathbf{\Omega}[\overline{x}]$ *be a sample space of programs. Define*

$$I_{\overline{x}} = \{x_1,\ \ldots,\ x_k\} \times \mathbb{Z}^k$$

to be an indexing set for a family of random variables. For each index

$$(x_i,\ a_1,\ \ldots,\ a_k) \in I_{\overline{x}},$$

define the random variable $S_{(x_i,\ a_1,\ \ldots,\ a_k)} : \Omega[\overline{x}] \to \mathbb{Z}_\perp$ *by*

$$S_{(x_i,\ a_1,\ \ldots,\ a_k)}(\omega) = \begin{cases} \perp & \text{if } \mathit{eval}(\ \omega,\ a_1,\ \ldots,\ a_k\ ) = \perp, \\ \mathit{eval}(\ \omega,\ a_1,\ \ldots,\ a_k\ )_i, & \text{otherwise.} \end{cases}$$

Thus $S_{(x_i,\ a_1,\ \ldots,\ a_k)}(\omega)$ gives the output obtained for the interface variable x_i by executing program ω on the input $a_1,\ \ldots,\ a_k$.

A *path* $S(\omega) : \{x_1,\ \ldots,\ x_k\} \times \mathbb{Z}^k \to \mathbb{Z}_\perp$ for the stochastic process $S = \langle\ S_i \mid i \in I_{\overline{x}}\ \rangle$ gives the entire input/output behaviour of the program ω.

Definition 5 exploits the analogy between: (i) a *black-box test* which can be viewed as an input/output measurement made on an unseen and therefore essentially randomly given program, and (ii) a *random variable* which is a measurable function between two σ-fields. A test sequence is just a sequence of such input/output measurements made on the same unseen program. Therefore, it is natural to model all possible tests on the same program as a path of a stochastic process. Hence we arrive at the model given by Definition 5.

Modelling programs as stochastic processes in this way now makes it possible to derive probabilistic statements about test success and failure.

In the remainder of this section, we let $(\Omega[\overline{x}],\ F,\ \mathbb{P})$ denote an arbitrary probability space, where $\mathbf{\Omega}[\overline{x}] = (\Omega[\overline{x}],\ eval)$ is a countable extensional sample space of programs, and $F = \wp(\Omega[\overline{x}])$. In order to assign probabilities to pre and postconditions on any program $\omega \in \Omega[\overline{x}]$ we need to be able to represent these as events (i.e. sets of programs) within the σ-field F. For this we begin by showing how first-order terms can be analysed in terms of the random variables introduced in Definition 5.

Now $\mathbb{Z}$ is a minimal Σ^{ring} structure. So by definition, for any integer $i \in \mathbb{Z}$ there exists a canonical numeral $\overline{i} \in T(\Sigma^{ring})$ which denotes i in $\mathbb{Z}$, i.e. $\overline{i}_{\mathbb{Z}} = i$.

Definition 6. *For any assignment $a : X \to \mathbb{Z}$, we define the translation mapping*

$$a^\sharp : T(\Sigma^{ring},\ X \cup \{x'_1,\ \ldots,\ x'_k\}) \to T(\Sigma^{ring},\ I_{\overline{x}})$$

by induction on terms.
(i) $a^\sharp(0) = 0$ and $a^\sharp(1) = 1$.
(ii) For any variable $x \in X$, $a^\sharp(x) = \overline{a(x)}$,
(iii) For any postvariable $x' \in \{x'_1,\ \ldots,\ x'_k\}$,

$$a^\sharp(x') = (\ x,\ a(x_1),\ \ldots,\ a(x_k)\ ).$$

(iv) For any terms $t_1,\ t_2 \in T(\Sigma^{ring},\ X \cup X')$, and for any function symbol $op \in \{+,\ *,\ -\}$,

$$a^\sharp(\ t_1\ op\ t_2\ ) = (\ a^\sharp(t_1)\ op\ a^\sharp(t_2)\ ).$$

In essence $a^\sharp$ replaces each logical variable and prevariable with the name of its value under a. Also $a^\sharp$ replaces each postvariable with the index of its corresponding random variable under a.

In order to translate first-order formulas into events we extend $a^\sharp$ to all first-order formulas by mapping these into the quantifier-free fragment of $L_{\omega_1,\ \omega}(\Sigma,\ I_{\overline{x}})$, in which even bound variables have dissappeared.

Definition 7. *For any assignment $a : X \to A$, we define the translation mapping*

$$a^\sharp : L(\Sigma^{ring},\ X \cup \{x'_1,\ \ldots,\ x'_k\}) \to L_{\omega_1,\ \omega}(\Sigma^{ring},\ I_{\overline{x}})$$

by induction on formulas.
(i) For any terms $t_1,\ t_2 \in T(\ \Sigma^{ring},\ X \cup \{x'_1,\ \ldots,\ x'_k\}\ )$ and any relation symbol $R \in \{\leq,\ =\}$,

$$a^\sharp(\ t_1\ R\ t_2\ ) = (\ a^\sharp(t_1)\ R\ a^\sharp(t_2)\ ).$$

(ii) For any formulas $\phi_1,\ \phi_2 \in L(\ \Sigma^{ring},\ X \cup \{x'_1,\ \ldots,\ x'_k\}\ )$,

$$a^\sharp(\ \phi_1 \wedge \phi_2\ ) = (\ a^\sharp(\phi_1) \wedge a^\sharp(\phi_2)\ )$$

$$a^\sharp(\ \neg\phi_1\ ) = \neg(\ a^\sharp(\phi_1)\ )$$

(iii) For any variable $x \in X \cup \{x'_1, \ldots, x'_k\}$, *and any formula* $\phi \in L(\Sigma^{ring}, X \cup \{x'_1, \ldots, x'_k\})$,

$$a^\sharp(\forall x\ \phi) = \bigwedge_{i \in \mathbb{Z}} a[z \to i]^\sharp(\phi[x/z])$$

where $z \in X - \{x_1, \ldots, x_n\}$ *is the least (non-interface) variable (under a fixed enumeration) such that* z *is not free in* ϕ *and* $a[z \to i] : X \to A$ *agrees with* a *everywhere except on* z, *where* $a[z \to i](z) = i$.

Now we can easily associate an event consisting of a set of programs with every quantifier free formula $\phi \in L_{\omega_1, \omega}(\Sigma^{ring}, I_{\overline{x}})$ as follows.

Definition 8. *Define the event set* $\mathfrak{F}(\phi) \subseteq \Omega[\overline{x}]$ *for each quantifier free formula* $\phi \in L_{\omega_1, \omega}(\Sigma^{ring}, I_{\overline{x}})$ *by induction on formulas.*
(i) For any terms $t_1[i_1, \ldots, i_m], t_2[i_1, \ldots, i_m] \in T(\Sigma^{ring}, I_{\overline{x}})$, *and any relation symbol* $R \in \{\leq, =\}$,

$$\mathfrak{F}(t_1\ R\ t_2) =$$

$$\langle S_{i_1}, \ldots, S_{i_m} \rangle^{-1}(\{ b \in \mathbb{Z}^m \mid \mathbb{Z}, b \models t_1\ R\ t_2 \}).$$

(ii) For any quantifier free formulas $\phi_1, \phi_2 \in L_{\omega_1, \omega}(\Sigma^{ring}, I_{\overline{x}})$,

$$\mathfrak{F}(\phi_1 \wedge \phi_2) = \mathfrak{F}(\phi_1) \cap \mathfrak{F}(\phi_2)$$

$$\mathfrak{F}(\neg\phi_1) = \Omega[\overline{x}] - \mathfrak{F}(\phi_1)$$

(iii) For any countable family of quantifier free formulas $\langle \phi_i \in L_{\omega_1, \omega}(\Sigma^{ring}, I_{\overline{x}}) \mid i \in I \rangle$,

$$\mathfrak{F}(\bigwedge_{i \in I} \phi_i) = \bigcap_{i \in I} \mathfrak{F}(\phi_i).$$

Notice in Definition 8.(i) we assume that $\{ b \in \mathbb{Z}^m \mid \mathbb{Z}, b \models t_1\ R\ t_2 \}$ is indeed an event on $\mathbb{Z}^m$. In the case that we take the discrete σ-field $\wp(\mathbb{Z}^m)$ this requirement is trivially satisfied.

Using Definition 8 we can now translate the probability distribution $\mathbb{P}$ on programs into probability values for correctness statements. Thus we come to the central definitions of this section.

Definition 9. *Let* $\phi \in L(\Sigma^{ring}, X \cup \{x'_1, \ldots, x'_k\})$ *be any formula.*

(i) We define the probability that ϕ *is satisfiable under* $a : X \to \mathbb{Z}$ *by*

$$\mathbb{P}(Sat_a(\phi)) = \mathbb{P}(\mathfrak{F}(a^\sharp(\phi))).$$

(ii) We define the probability that ϕ *is satisfiable by*

$$\mathbb{P}(Sat(\phi)) = \mathbb{P}(\bigcup_{a : X \to \mathbb{Z}} \mathfrak{F}(a^\sharp(\phi))).$$

(i) We define the probability that ϕ is valid by

$$\mathbb{P}(\ \mathbb{Z} \models \phi\ ) = \mathbb{P}(\ \bigcap_{a:X\to\mathbb{Z}} \mathfrak{F}(\ a^\sharp(\phi)\ )\ ).$$

Definition 9.(i) provides a rigorous mathematical answer to the initial question this section. It is helpful to illustrate this definition with some simple examples.

Example 1. Let $x \in X$ be a single variable interface.
(i) Consider the specification

$$\{\ \}\omega\{x' = m * x + c\},$$

which asserts that $\omega[x]$ computes a linear function $f(x) = mx + c$ of its input variable x. For any input assignment of $a \in \mathbb{Z}$ to x

$$\mathfrak{F}(\ a^\sharp(x' = m * x + c)\ ) =$$

$$\mathfrak{F}(\ (\ x,\ a\ ) = m * a + c\ ) =$$
$$\langle\ S_{(x,\ a)}\ \rangle^{-1}\{b \in \mathbb{Z}\ \mid\ \mathbb{Z},\ b \models (\ x,\ a\ ) = m * a + c\} =$$
$$\{\omega \in \Omega[x]\ \mid\ eval(\omega,\ a) = m * a + c\}.$$

Thus

$$\mathbb{P}(\ Sat_a(\ \neg\ x' = m * x + c\ )\ ) =$$
$$\mathbb{P}(\ \{\omega \in \Omega[x]\ \mid\ eval(\omega,\ a)\ \neq m * a + c\}\ )$$

is the probability that a randomly chosen single variable program $\omega[x] \in \Omega[x]$ fails the postcondition $x' = m * x + c$ on the test input $a \in \mathbb{Z}$.
(ii) More generally, the probability that a program $\omega[x]$ will fail a test $a_{k+1} \in \mathbb{Z}$ of p and q given that ω has already passed k tests $a_1,\ \ldots,\ a_k \in \mathbb{Z}$ with output $b_1,\ \ldots,\ b_k \in \mathbb{Z}$ is the conditional probability

$$\mathbb{P}(\ Sat_{a_{k+1}}(\ p \wedge \neg q\ )\ |\ \{\omega \in \Omega : eval(\omega,\ a_i) = b_i\ \text{ for }\ 1 \leq i \leq k\}\ ).$$

Definition 9 satisfies several intuitive properties.

Proposition 1. *Let $\phi,\ \psi \in L(\ \Sigma^{ring},\ X \cup \{x'_1,\ \ldots,\ x'_k\}\ )$ be any formulas.*

(i) If $\mathbb{Z} \models \phi$ then $\mathbb{P}(\ \mathbb{Z} \models \phi\ ) = 1$.

(ii) If $\mathbb{Z} \models \neg\phi$ then $\mathbb{P}(\ \mathbb{Z} \models \phi\ ) = 0$.

(iii) If $\mathbb{Z} \models \phi \to \psi$ then $\mathbb{P}(\ \mathbb{Z} \models \phi\ ) \leq \mathbb{P}(\ \mathbb{Z} \models \psi\ )$.

(iv) $\mathbb{P}(\ \mathbb{Z} \models \phi\ ) = 1 - \mathbb{P}(\ Sat(\neg\phi)\ )$.

(v) If $\mathbb{Z} \models \phi \leftrightarrow \psi$ then $\mathbb{P}(\ \mathbb{Z} \models \phi\ ) = \mathbb{P}(\ \mathbb{Z} \models \psi\ )$.

Proof. Follows easily from Definition 9.

By Proposition 1.(v) the probability that a correctness formula is valid is independent of its syntactic structure and depends only on its semantics.

Although these properties are intuitive, they are not satisfied by any of the reliability models of [Hamlet 1987], [Miller et al. 1992] or [Thayer et al. 1978]. For example, these models all assign a probability $p < 1$ of satisfying a tautology. We believe this points to a significant conceptual flaw in existing models of probabilistic correctness in the literature.

4 Test Coverage and Software Reliability

Given that n tests of a program $\omega[\overline{x}]$ are unsuccessful in finding an error in ω, what is the probability that ω satisfies a specification $\{p\}\omega\{q\}$? If it is possible to calculate or even estimate this probability value, then we have a clearly defined stopping criterion for black-box testing: we may terminate when a desired probability of correctness has been achieved. Thus the concept of probability of correctness gives a formal model of black-box test coverage, where by coverage we mean the extent of testing. We shall apply the theoretical model introduced in Section 3 to consider the problem of estimating the probability of correctness.

An obvious technical problem is to find a distribution $\mathbb{P}$ on programs which is realistic. However, to begin to study coverage and the testing termination problem we can use very simple heuristical probability distributions, and examine how calculations can be made.

For simplicity, we consider programs with a single integer variable interface $x \in X$. Let

$$\mathbf{\Omega}[x] = (\ \Omega[x],\ \mathit{eval}\ ).$$

Also, for simplicity, we assume that $\Omega[x]$ is a subrecursive language, i.e. each program $\omega[x] \in \Omega[x]$ terminates on all inputs. Thus $\mathit{eval} : \Omega[x] \times \mathbb{Z} \to \mathbb{Z}$ is also a total function, which allows us to work with totally defined random walk models of paths.

To calculate the probability of satisfying a formula ϕ, we need a probability distribution $\mathbb{P} : \wp(\ \Omega[x]\ ) \to [0,\ 1]$ Recalling Definition 5, one approach is to consider the associated family of random variables

$$S = \langle\ S_{(x,\ i)} : \Omega[x] \to \mathbb{Z}\ \mid\ i \in \mathbb{Z}\ \rangle.$$

A simple model of the FDDs of these random variables is to assume a *random walk hypothesis*. Writing S_i for $S_{(x,\ i)}$ we can relate S_n and S_{n+1} by a formula $S_{n+1} = S_n + X_n$ where

$$\langle\ X_i : \Omega[x] \to \mathbb{Z}\ \mid\ i \in \mathbb{Z}\ \rangle$$

is another family of random variables. A simple relationship is to define an *exponential distribution* on the X_i by

$$\mathbb{P}(X_i = +n) = \mathbb{P}(X_i = -n) = \frac{1}{3}\left(\frac{1}{2}\right)^n$$

for all $n \in \mathbb{N}$.

An exponential random walk over a finite interval can model any total function over that interval. This distribution captures the simple intuition that fast growing functions are increasingly unlikely. Furthermore it is easily analysed, and for simple formulas, we can estimate the probability of satisfiability. After n test passes on the inputs $a_1 \leq a_2 \leq \ldots \leq a_n \in \mathbb{Z}$ we have a sequence of $n-1$ intervals $[a_i,\ a_{i+1}]$. We can consider the probability of satisfying a formula $p \wedge \neg q$ (where p is a precondition and q is a postcondition) over each of these intervals separately. To perform such an analysis, we first note that the interval $[a_i,\ a_{i+1}]$ can be renormalised to the interval $[0,\ a_{i+1} - a_i]$ without affecting the probability values. (An exponential random walk is homogeneous along the x-axis.) Let us consider probabilities for the individual paths over an interval $[0,\ a]$.

The probability of a path following an exponential random walk is a function of its length and its volatility.

Definition 10. *Let $y = y_0,\ y_1,\ \ldots,\ y_n \in \mathbb{Z}^{n+1}$ be a path of length $n \geq 1$. We define the volatility $\lambda(y) \in \mathbb{Z}$ of y by*

$$\lambda(y) = \sum_{i=1}^{n} \mid y_i - y_{i-1} \mid.$$

Proposition 2. *Let $y = y_0,\ y_1,\ \ldots,\ y_n \in \mathbb{Z}^{n+1}$ be a path of length $n \geq 1$. Then*

$$\mathbb{P}(\ S_i = y_i \ \textit{for}\ \ i = 1,\ \ldots,\ n \ \mid\ S_0 = y_0\ ) = \left(\frac{1}{3}\right)^n \left(\frac{1}{2}\right)^{\lambda(y)}.$$

Proof. By induction on n.

Let us consider monotone paths.

Definition 11. *Let $y = y_0,\ y_1,\ \ldots,\ y_n \in \mathbb{Z}^{n+1}$ be a path of length $n \geq 1$. We say that y is monotone if $y_0 \leq y_1 \leq \ldots \leq y_n$ or $y_0 \geq y_1 \geq \ldots \geq y_n$.*

Proposition 3. *Let $y = y_0,\ y_1,\ \ldots,\ y_n \in \mathbb{Z}^{n+1}$ be a path of length $n \geq 1$.*
(i) If y is monotone then $\lambda(y) = |y_n - y_0|$.
(ii) If y is non-monotone then $\lambda(y) > |y_n - y_0|$.

Proof. By induction on the length of paths.

Thus it is easy to calculate the probability of a monotone path.

Corollary 1. *Let $y = y_0,\ y_1,\ \ldots,\ y_n \in \mathbb{Z}^{n+1}$ be a monotone path of length n. Then*

$$\mathbb{P}(\ S_i = y_i \ \textit{for}\ \ i = 1,\ \ldots,\ n \ \mid\ S_0 = y_0\ ) = \left(\frac{1}{3}\right)^n \left(\frac{1}{2}\right)^{|y_n - y_0|}.$$

Proof. Immediate from Propositions 2 and 3.

So the probability of a monotone path under the exponential distribution is independent of the steps taken, and depends only on the start and end points. In fact, by Proposition 3, this property characterises the monotone paths. Furthermore, any non-monotone path y from y_0 to y_n has exponentially lower probability than any monotone path from y_0 to y_n by a factor $\left(\frac{1}{2}\right)^{\lambda(y)-|y_n-y_0|}$.

Let $C^R(\ n,\ r\ )$ be the number of ways of selecting a collection of r objects from a total of n objects with repetitions. Then

$$C^R(\ n,\ r\ ) = C(\ n+r-1,\ r\ )$$

where $C(\ n,\ r\ )$ is the binomial coefficient defined by

$$C(\ n,\ r\ ) = \frac{n(n-1)\ldots(n-r+1)}{r!}.$$

Recall the well known *upper negation identity* (see e.g. [Graham et al. 1989])

$$C(\ n,\ r\ ) = (-1)^r C(\ r-n-1,\ r\ ),$$

from which we can infer $C^R(\ n,\ r\ ) = (-1)^r C(\ -n,\ r\ )$.

Theorem 1. *For any $n \geq 1$ and $y_0,\ y_n \in \mathbb{Z}$,*

$$\mathbb{P}(\ S_n = y_n \mid S_0 = y_0\ ) =$$

$$\left(\frac{1}{3}\right)^n \left(\frac{1}{2}\right)^{|y_n-y_0|} \left(C^R(\ n,\ |y_n-y_0|\ ) + \right.$$

$$\left. \sum_{i>0} \sum_{k=1}^{min(i,\ n-1)} C(\ n,\ k\ )\ C^R(\ k,\ i-k\ )\ C^R(\ n-k,\ |y_n-y_0|+i\ ) \left(\frac{1}{2}\right)^i \right).$$

Proof. Apply Proposition 2 and sum over all volatility values.

To illustrate the approach, we estimate the probability of a single variable program $\omega[x]$ failing a test of a simple linear equational specification

$$\{\ \}\omega\{x' = m * x + c\},$$

within an interval $[0, n]$. (Recall Example 1.) We may assume that ω passes both tests of the endpoints 0 and n.

Theorem 2. *(i) For any $n > 1$ and any $c \in \mathbb{Z}$,*

$$\mathbb{P}(\ S_i \neq c \ \textit{for some}\ 0 < i < n \mid S_0 = c,\ S_n = c\ ) \approx \left(\frac{n-1}{n+1}\right).$$

(ii) For any $n > 1$ and any $m,\ c \in \mathbb{Z}$, where $m \neq 0$,

$$\mathbb{P}(\ S_i \neq mi + c \ \textit{for some}\ 0 < i < n \mid S_0 = c,\ S_n = mn + c\ )$$

$$\approx 1 - \frac{1}{C^R(\ n,\ |nm|\ )}.$$

Proof. (i) Follows from Proposition 2 and Theorem 1 by considering non-monotone paths with highest probability satisfying $S_i \neq c$ for some $0 < i < n$.

(ii) Clearly, there is only one monotone path $y_0, \ \ldots, \ y_n$ of length n, satisfying $y = mx + c$, namely

$$y = y_0 = c, \ y_1 = m + c, \ \ldots, \ y_n = nm + c.$$

So for all monotone paths from c to $nm + c$ excluding y, using Corollary 1 and Theorem 1,

$$\mathbb{P}(\ S_n = nm + c, \ S_i \neq mi + c \ \textit{for some} \ 0 < i < n \ \mid \ S_0 = c\ ) =$$

$$(\ \ C^R(\ n, \ |nm|\ ) - 1\ \ )\left(\frac{1}{3}\right)^n \left(\frac{1}{2}\right)^{|nm|}.$$

Hence the result follows.

By a similar analysis of other types of correctness formulas, it becomes clear that closed form solutions to reliability estimation problems become intractable for anything other than simple kinds of formulas. For practical testing problems, Monte Carlo simulation (see e.g. [Bouleau 1994]) seems to be a necessary tool to estimate reliability after n tests, even for such a simple distribution as the exponential random walk.

Clearly, the results of this section depend on specific properties of the exponential random walk. This distribution model represents a naive but mathematically tractable model of reality. An open question for future research is to find more realistic models or program distributions. Of course, more accurate models would lead to slightly different results than those presented here.

5 Test Case Generation (TCG)

In section 4 we considered the problem of stopping the testing process with some quantitative conclusion about the reliability of a program after n tests have been passed. In this section we consider how to apply our stochastic model to the actual testing phase that precedes termination. How can we use the stochastic approach to efficiently generate test cases that can effectively uncover errors? We have already seen in Section 4 that calculations of correctness probabilities may be computationally expensive. However, for certain kinds of probability distributions an approach to TCG can be developed from outside probability theory, using classical *function approximation theory*, with the advantage of efficient speed.

For clarity of exposition, we will again deal with the case of a program interface consisting a single input/output variable $x \in X$. Furthermore, we will generalise from probability measures to arbitrary finite measures at this point. (Recall that every probability measure is a measure, but not vice-versa.)

Definition 12. *Let* $M : [\,[m,n] \to D\,] \to \mathbb{R}^+$ *be a measure for* $D \subseteq \mathbb{Z}$. *(If the codomain of* M *is* $[0,1]$ *then* M *is an FDD.) Then* M *is elective if, and only if, for any* $(a_1,\ b_1),\ \ldots,\ (a_k,\ b_k) \in [m,n] \times D$ *the set*

$$F_{(a_1,\ b_1),\ \ldots,\ (a_k,\ b_k)} = \{\ g : [m,n] \to D \mid g(a_i) = b_i \ \text{ for } \ 1 \leq i \leq k\ \}$$

has a unique maximum member under M, *i.e. there exists* $f \in F_{(a_1,\ b_1),\ \ldots,\ (a_k,\ b_k)}$ *such that for all* $g \in F_{(a_1,\ b_1),\ \ldots,\ (a_k,\ b_k)}$

$$g \neq f \ \ \Rightarrow \ \ M(g) < M(f).$$

To understand Definition 12, suppose that $b_1,\ \ldots,\ b_k \in D$ are the results of executing a program $\omega[x]$ on the test inputs $a_1,\ \ldots,\ a_k \in [m,n]$ respectively. Then an elective measure M gives for the input/output pairs

$$(a_1,\ b_1),\ \ldots,\ (a_k,\ b_k)$$

a unique "most likely candidate" for a function f extending these pairs to the entire interval $[m,n]$. This candidate function f, which is the maximum member $f \in F_{(a_1,\ b_1),\ \ldots,\ (a_k,\ b_k)}$ under M, represents a "best" guess of what the partially known system under test might look like in its entirety.

An elective measure $M : [\,[m,n] \to D\,] \to \mathbb{R}^+$ gives rise to an iterative test case generation procedure in the following way. Given k executed test cases $a_1,\ \ldots,\ a_k \in [m,\ n]$ for a program $\omega[x]$ with results $b_1,\ \ldots,\ b_k \in D$, we can consider the unique elected function $f_k \in F_{(a_1,\ b_1),\ \ldots,\ (a_k,\ b_k)}$ as a model of $\omega[x]$. By analysing f_k we may be able to locate a new test case $a_{k+1} \in [m,n]$ such that a_{k+1} satisfies a precondition p but $f_k(a_{k+1})$ does not satisfy a postcondition q. Then a_{k+1} is a promising new test case to execute on $\omega[x]$. If no such a_{k+1} exists we can use some other choice criteria (e.g. random) for the $k+1$-th test, and hope for a more promising test case later as the sequence of elected functions $f_k : k \geq 1$ converges to the actual input/output behaviour of $\omega[x]$.

The fundamental technical problem for this approach to TCG is to find a suitable elective measure M. One pragmatic solution to this problem is introduced in [Meinke 2004] using function approximation theory. Specifically, an *interpolant* (usually a local interpolant) of $(a_1,\ b_1),\ \ldots,\ (a_k,\ b_k) \in [m,n] \times D$ is chosen as the elected function. In [Meinke 2004] piecewise polynomials were investigated as local interpolants. Many other classes of approximating functions are known in the literature such as splines, wavelets, radial basis functions, etc. Thus the technique gives a rich source of algorithms.

Our main result in this section is to show that for a large class of approximation methods, the function approximation approach (which is non-probabilistic, and fast to the extent that interpolants can be efficiently computed and evaluated) is equivalent to the measure theoretic approach.

Definition 13. *Let* $D \subseteq \mathbb{Z}$ *be any subset. An interpolation scheme is a mapping* $I : \wp([m,n] \times D) \to [\,[m,n] \to D\,]$ *such that for all* $1 \leq i \leq k$

$$I(\ \{(a_1,\ b_1),\ \ldots,\ (a_k,\ b_k)\}\ )(a_i) = b_i.$$

We will show that a large class of interpolation schemes, including polynomial interpolation, actually give rise to elective measures, and even elective probability measures. Thus the approximation approach can be seen as a special case of the stochastic approach, where the FDDs are implicit, but can be efficiently computed.

Definition 14. *Let* $\mu : 2^{2^{[m,n]}} \to \mathbb{R}^+$ *be a finite measure (not necessarily a probability measure). Let*

$$I : \wp([m,n] \times D) \to [\ [m,n] \to D\ ]$$

be an interpolation scheme. Define the measure

$$\mu^I : [\ [m,n] \to D\ ] \to \mathbb{R}^+$$

by

$$\mu^I(\ f\ ) = \mu(\ \{\ \{a_{i_1},\ \ldots,\ a_{i_k}\ \} \subseteq [m,n]\ |$$
$$I[\ \{\ (a_{i_1},\ f(a_{i_1})),\ \ldots,\ (a_{i_1},\ f(a_{i_1}))\ \}\ ] = f\ \}\ ).$$

Proposition 4. *If* $D \subseteq \mathbb{Z}$ *is finite then* μ *can be defined so that* μ^I *is a probability measure for any interpolation scheme* I*.*

Proof. By construction.

Definition 15. *Let* $M : [\ [m,n] \to D\ ] \to \mathbb{R}^+$ *be an elective measure. Define the interpolation scheme* $I^M : \wp([m,n] \times D) \to [\ [m,n] \to D\ ]$ *by*

$$I^M[\ \{\ (a_0,\ b_0),\ \ldots,\ (a_k,\ b_k)\ \}\ ] = f,$$

where $f \in \{\ g : [m,n] \to D \mid g(x_i) = y_i\ \text{for}\ 1 \leq i \leq k\ \}$ *is the unique element for which* $M(f)$ *is maximum.*

Definition 16. *Let* $I : \wp([m,n] \times D) \to [\ [m,n] \to D\ ]$ *be an interpolation scheme.*
(i) I *is monotone if, and only if,* $I[\ \{\ (a_0,\ b_0),\ \ldots,\ (a_k,\ b_k)\ \}\ ] = f$ *and*
$\{\ (a'_0,\ b'_0),\ \ldots,\ (a'_j,\ b'_j)\ \} \subseteq f$ *and* $\{\ a_0,\ \ldots,\ a_k\ \} \subseteq \{\ a'_0,\ \ldots,\ a'_j\ \}$ *imply*

$$I[\ \{\ (a'_0,\ b'_0),\ \ldots,\ (a'_j,\ b'_j)\ \}\ ] = f.$$

(ii) I *is permutable if, and only if, for any* $\{\ a_1,\ \ldots,\ a_k\ \} \subseteq [m,n]$ *and* $f : [m,n] \to D$ *if* $I[\ \{\ (a_1,\ f(a_1)),\ \ldots,\ (a_k,\ f(a_k))\ \}\ ] = f$ *then for any*
$\{\ a'_1,\ \ldots,\ a'_k\ \} \subseteq [m,n]$

$$I[\ \{\ (a'_1,\ f(a'_1)),\ \ldots,\ (a'_k,\ f(a'_k))\ \}\ ] = f.$$

Example 2. Polynomial approximation is monotone and permutable.

Theorem 3. *Let* $I : \wp([m,n] \times D) \to [\ [m,n] \to D\ ]$ *be a monotone permutable interpolation scheme. Then* μ^I *is elective and* $I^{\mu^I} = I$.

Proof. Consider any $\{(a_1,\ b_1),\ \ldots,\ (a_n,\ b_n)\} \subseteq [m,n] \times D$ and suppose

$$I[\ \{(a_1,\ b_1),\ \ldots,\ (a_n,\ b_n)\}\ ] = f,$$

then we need to show that

$$I^{\mu^I}[\ \{(a_1,\ b_1),\ \ldots,\ (a_n,\ b_n)\}\ ] = f.$$

It suffices to show that for any $g : [m,n] \to D$ such that $g(a_i) = b_i$ for $1 \leq i \leq n$ and $g \neq f$, $\mu^I(g) < \mu^I(f)$, i.e. μ^I is elective. Consider any such g and any $\{a'_1,\ \ldots,\ a'_k\} \subseteq [m,n] \times D$ and suppose that

$$I[\ \{(a'_1,\ g(a'_1)),\ \ldots,\ (a'_k,\ g(a'_k))\}\ ] = g.$$

Since I is permutable we must have $k \geq n$.

Since I is an interpolation scheme

$$I[\ \{(a_1,\ f(a_1)),\ \ldots,\ (a_n,\ f(a_n))\}\ ] = f.$$

Then since I is permutable

$$I[\ \{(a'_1,\ f(a'_1)),\ \ldots,\ (a'_n,\ f(a'_n))\}\ ] = f,$$

Finally since I is monotone and $k \geq n$,

$$I[\ \{(a'_1,\ f(a'_1)),\ \ldots,\ (a'_k,\ f(a'_k))\}\ ] = f.$$

Therefore

$$\{\{a_1,\ \ldots,\ a_k\} \subseteq [m,n] \mid I[\ \{(a_1,\ g(a_1)),\ \ldots,\ (a_k,\ g(a_k))\}\ ] = g\} \subseteq$$

$$\{\{a_1,\ \ldots,\ a_k\} \subseteq [m,n] \mid I[\ \{(a_1,\ f(a_1)),\ \ldots,\ (a_k,\ f(a_k))\}\ ] = f\}.$$

Thus since μ is a measure, $\mu^I(f) > \mu^I(g)$, i.e. μ^I is elective.

6 Conclusions

In this paper we have introduced a stochastic model for black-box testing of the functional correctness of programs. This model allows us to derive a probability value for the validity of a correctness formula of the form $\{p\}\omega\{q\}$ conditional on the results of any finite set of black-box tests on ω. It corrects technical problems with similar models occuring previously in the literature. Our model provides a solution to the difficult problem of measuring coverage in black-box testing. It also suggests new approaches to the test case generation process itself.

Further research is necessary to establish accurate models of the probabilistic distribution of programs. Furthermore, we may generalise our model to consider how program distributions are influenced by the choice of the programming problem to be solved (the precondition p and postcondition q). This would give a theoretical model of the *competent programmer hypothesis* of [Budd 1980]. This also requires consideration of the difficult problem of non-termination. For example, it may be necessary to introduce a non-functional time requirement into specifications, in order to abort a test that can never terminate. Research into other abstract data types and concrete data structures also presents an important problem in this area.

Much of this research was carried out during a sabbatical visit to the Department of Computer Science Engineering at the University of California at San Diego (UCSD) during 2003. We gratefully acknowledge the support of the Department, and in particular the helpful comments and advice received from Joseph Goguen and the members of the Meaning and Computation group. We also acknowledge the financial support of TFR grant 2000-447.

References

[de Bakker 1980] J.W. de Bakker, *Mathematical Theory of Program Correctness*, Prentice-Hall, 1980.

[Barwise 1968] J. Barwise (ed), *The Syntax and Semantics of Infinitary Languages*, Lecture Notes in Mathematics 72, Springer-Verlag, Berlin, 1968.

[Bauer 1981] H. Bauer, *Probability Theory and Elements of Measure Theory*, Academic Press, London, 1981.

[Beizer 1995] B. Beizer, *Black-Box Testing*, John Wiley, 1995.

[Bouleau 1994] N. Bouleau, D. Lepingle, *Numerical Methods for Stochastic Processes*, John Wiley, New York, 1994.

[Budd 1980] Budd, T.A. DeMillo, R.A. Lipton, R.J. Sayward, F.G. Theoretical and Empirical Studies on Using Program Mutation to Test the Functional Correctness of Programs, Proc. 7th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, 220-223, 1980.

[Graham et al. 1989] R.L. Graham, D.E. Knuth and O. Patashnik, *Concrete Mathematics*, Addison-Wesley, Reading Mass., 1989.

[Grimmet et al. 1982] G. Grimmet, D. Stirzaker, *Probability and Random Processes*, Oxford University Press, 1982.

[Hamlet 1987] Hamlet, R.G. Probable Correctness Theory, Inf. Proc. Letters 25, 17-25, 1987.

[Kallenberg 1997] O. Kallenberg, *Foundations of Modern Probability*, Springer Verlag, 1997.

[Lee and Yannakakis 1996] D. Lee, M. Yannakakis, Principles and Methods of Testing Finite State Machines - a Survey, Proc. IEEE, **84** (8), 1090-1123, 1996.

[Loeckx et al. 1996] J. Loeckx, H-D. Ehrich, M. Wolf, *Specification of Abstract Data Types*, Wiley Teubner, Chichester 1996.

[Meinke 2004] K. Meinke, Automated Black-Box Testing of Functional Correctness using Function Approximation, pp 143-153 in: G. Rothermel (ed) *Proc. ACM SIGSOFT Int. Symp. on Software Testing and Analysis, ISSTA 2004*, Software Engineering Notes 29 (4), ACM Press, 2004.

[Miller et al. 1992] Miller, K.W. Morell, L.J. Noonan, R.E. Park, S.K. Nicol, D.M. Murrill, B.W. Voas, J.M.: Estimating the Probability of Failure when Testing Reveals no Failures, IEEE Trans. Soft. Eng. 18 (1), 33-43, 1992.

[Moore 1956] E.F. Moore, Gedanken-experiments on Sequential Machines, Princeton Univ. Press, Ann. Math. Studies, 34, 129-153, Princeton NJ, 1956.

[Thayer et al. 1978] Thayer, T.A. Lipow, M. Nelson, E.C.: Software Reliability, North Holland, New York, 1978.

[Weiss and Weyuker 1988] Weiss, S.N. Weyuker, E.J.: An Extended Domain-Based Model of Software Reliability, IEEE Trans. Soft. Eng. 14 (10), 1512-1524, 1988.