Abstract

Software testing is a process in which a software system’s dynamic behaviours are observed and analysed so that the system’s properties can be inferred from the information revealed by test executions. While the existing theories of software testing might be adequate in describing the testing of sequential systems, they are not capable to describe the testing of concurrent systems that can exhibit different behaviours on the same test case due to non-determinism and concurrency. In our previous work, we proposed a theoretical framework that provides a unified foundation to study all testing methods for both sequential and concurrent systems. The main results of the framework include a formal definition of the notion and a set of the desirable properties of an observation scheme. In this paper, we provide several constructions of observation schemes that have direct implications in current software testing practice. We also study the properties of the constructions.

1. Introduction

Software testing is an essential approach for software quality assurance. Over the past thirty years, a great number of testing methods has been proposed and investigated in the literature [1]. Given such a large variety of software testing methods, two important questions arise - how these testing methods are related to each other and when a particular testing method should be used. A rapid growing body of research results aiming to answer the above questions has emerged in the past decades. Work in this direction includes comparisons of testing methods by the subsumption relationships between test adequacy criteria [2–5], empirical and experimental comparisons of the fault detection abilities of testing methods [6–9], and theoretical assessments of testing methods against axioms of software test adequacy [10–17].

These research results are based more or less on a widely accepted notion of software test adequacy criteria, which can be defined as rules for measuring or determining the adequacy of a test set for testing a program against a specification [10]. Formally, a test adequacy criterion can be defined as a predicate $C : T \times P \times S \rightarrow \text{Bool}$ on the space $T$ of test sets, programs $P$ and specifications $S$ (see e.g. [10–13, 18–20]), or as a function $C : T \times P \times S \rightarrow [0, 1]$ mapping from test sets, programs and specifications to a set of numerical scales (see e.g. [14]).

However, the above theories with regard to testing adequacy are primarily for sequential systems. They are not adequate for concurrent systems due to the fact that the behaviour of a concurrent program cannot be uniquely determined by the input test cases due to non-determinism [21]. Neither the predicate nor the functional formulation mentioned above is appropriate. The dynamic behaviours of a concurrent system must be taken into consideration in defining testing adequacy criteria.

Several researchers proposed specific testing methods and adequacy criteria for concurrent systems. In [22], several structural coverage adequacy criteria were defined for concurrent programs. The adequacy criteria were not directly defined on the concurrent program under test, but on the sequences of concurrent states resulted from the execution of the program. Furthermore, the adequacy criteria were only defined on behaviours; the test cases were completely ignored. In [23], six adequacy criteria for a particular concurrency model, Petri nets, were proposed. In [24], a testing method for concurrent systems was presented and several specification-based adequacy criteria were defined.

In [25], we proposed the notion of observation schemes, which is an abstraction of the ways how dynamic behaviours are observed. We argued that the space of phenomena - i.e. observable behaviours by testing - constitute a complete partially ordered set, which is an algebraic structure commonly used in the denotational semantics of programming languages. An observation scheme consists of two components: a set of observable phenomena and the relationships between test sets and the phenomena observable during testing. Because in the testing of non-deterministic and concurrent systems a test may reveal a number of possible phenomena, we defined an observation scheme as a mapping that associates a test with its corresponding set of observable phenomena. This definition also covers the testing of sequential software, where each test is associated with a singleton set of phenomena.
phenomenon. As a characterisation of systematic and consistent methods of behaviour observations, a set of desirable properties of observation schemes were proposed and their interrelationships were proved. We also proposed and investigated a relationship between observation schemes, which is called extraction relation. The framework has been applied to the analysis of the existing testing methods as well as the development of new testing methods for Petri nets [26].

Based on the previous work, this paper presents several common constructions of observation schemes and shows their properties. Each construction is an abstraction of some methods of observations defined in existing testing methods. These constructions enable us to understand the properties of the existing testing methods from a high level of abstraction and to develop new testing methods of desired properties.

The paper is organized as follows. In section 2, we briefly review our previous work. We introduce the notion of observation schemes, discuss their desirable properties, and define the extraction relationship between them. In section 3, we define a number of constructions of observation schemes. For each construction, we examine their properties against the axioms discussed in section 2. Section 4 is the conclusion of the paper.

2. Overview of the previous results

2.1. The notion of observation schemes

In software testing practice, observations on a system’s dynamic behaviour can be made on a number of different aspects of the executions of the system. For example, in addition to the correctness of the output, it can also be: (1) the set of executed statements as in statement testing method; (2) the set of exercised branches as in branch testing method; (3) the set of executed paths as in path testing methods; (4) the sequences of communications occurred between processes as in the testing of communication protocols; (5) the sequences of synchronisation events as in [22]; (6) the set of dead mutants as in mutation testing, and so on. To make observed and recorded information meaningful, we require that observations be systematic and consistent. For example, it is not acceptable if we record executed statements at one time and communications between processes at another time during a single test execution.

We use the word scheme to denote a systematic and consistent way of observing and recording dynamic behaviours in software testing. We require that a scheme have a universe of observable phenomena of systems’ dynamic behaviours. This universe must have the following properties.

(A) Existence of a partial ordering on observable phenomena. Suppose that a software system is tested and an observation is made. By carrying out more tests on the same software system and use the same observation scheme, we should be able to obtain more information about the dynamic behaviour of the system. The final cumulative observation should contain more information than the intermediate observation. Therefore, there is an ordering between these two observations. However, observations made during two independent tests are not necessarily ordered. Therefore, all possible observations made during testing a software system form a partial order. For example, in statement testing, an observation of the dynamic behaviour of a system is a set of executed statements during testing. All such sets constitute a universe of phenomena and the partial ordering relation on the universe is the set inclusion. The more a program is tested, the larger the set of statements is executed.

(B) Existence of a summation operation. A summation operation is needed to add up a number of individual observations made during testing a software system. The result of such a summation should be the least upper bound of the individual observations. For statement testing, set union on the executed statements is the summation operation on the space of observable phenomena.

(C) Existence of the least element. Nothing about the dynamic behaviour of a system can be observed if the system is not executed. To denote such a situation, the universe of phenomena is required to contain an element denoting “no information”. This element must be the least element in the universe of phenomena, because it must contain less information than any other observable phenomenon. For example, in statement testing, the least element is the empty set, which means no statement is executed. Of course, any other set of statements observed in a testing includes the empty set.

A universe of the above properties constitutes an algebraic structure called Complete Partially Ordered set (CPO set). Such algebraic structures have been well studied in programming language semantics under the title of domain theory and denotational semantics. Readers are referred to [27] for a concise treatment of the domain theory and their uses in semantics of programming language. For the sake of readability, the following gives the mathematical notions and notations used in the paper.

Let $D$ be a non-empty set, $\leq$ be a binary relation on $D$, and $S \subseteq D$ be a subset of $D$.

- **Partial ordering**: the binary relation $\leq$ is a partial ordering, if $\leq$ is reflexive, transitive and anti-symmetric.
- **Upper bound**: $u$ is called an upper bound of $S$, if $s \leq u$ for all $s \in S$.
- **Consistent subset**: $S$ is a consistent subset if for all $s_1$, $s_2 \in S$, there is $s \in D$ such that $s_1 \leq s$ and $s_2 \leq s$. We say that $s_1$ is consistent with $s_2$ and write $s_1 \parallel s_2$.
- **Directed subset**: $S$ is a directed subset, if for all $s_1$, $s_2 \in S$, there exists $s \in S$ such that $s_1 \leq s$ and $s_2 \leq s$. 


Complete partially ordered set (CPO): \( <D, \leq> \) is called a complete partially ordered set, if (1) \( D \) has a least element, written \( \bot \), for every directed subset \( S \subseteq D, S \) has a least upper bound, written as \( \sum S \), i.e. for any other upper bound \( u, \sum S \leq u \).

Let \( I \) be any index set, \( x_i \) be a variable that ranges over set \( X_i, \forall i \in I \). We write \( \exists x_i \in X_i, \text{Pred} \) as a short hand for \( \exists x_1 \in X_1, \exists x_2 \in X_2, \ldots, \text{Pred} \). \( \forall x_i \in X_i, \text{Pred} \) is defined similarly.

The bag \( X \) is used to denote the set of all multiple sets on a set \( X \). The traditional set operations are used to denote their multiple set variants as well. \( \tilde{Y} \) is the set obtained by removing duplicated elements in a multiple set \( Y \).

Let \( \varphi \) be a mapping from \( X \) to \( Y \). For all subsets \( A \subseteq X \), we define \( \varphi(A) = \{ \varphi(a) | a \in A \} \).

In the sequel, we use the following symbols.

\( D \) and its variants for a concurrent system, and \( P \) for the set of all concurrent systems;
\( \sigma \) and its variants for a phenomenon observable during testing a concurrent system, and \( \Gamma \) for a set of phenomena.

An observation scheme is thus a mapping from a test set to a set of elements in the universe of phenomena. Each phenomenon is a possible observation when the software system is executed on the test set. We now formally define the notion of observation schemes as a mathematical structure.

Definition 1. (Observation Scheme)

A scheme of behaviour observation and recording, or simply an observation scheme, is a mapping from concurrent systems \( p \) to ordered pairs \( \langle B_p, \mu_p \rangle \), where \( B_p = \langle B_p, \leq_p \rangle \) is a CPO set, called the universe of phenomena on \( p \). \( \mu_p \) is called the recording function, which is a mapping from a test set \( T \) to a non-empty consistent subset of \( B_p \).

In formally, each element in \( B_p \) is a phenomenon observable from testing a concurrent system \( p \). \( \sigma_1 \leq_p \sigma_2 \) means that phenomenon \( \sigma_2 \) contains more information than phenomenon \( \sigma_1 \). The least element \( \bot_p \) denotes the least information that one can observe using the scheme. It can be regarded as ‘nothing is observed’. \( \mu_p(T) \) is the set of all possible phenomena observable by testing \( p \) on test set \( T \). In other words, \( \sigma \in \mu_p(T) \) means that \( \sigma \) is a phenomenon that is observable by an execution of \( p \) on test set \( T \).

The following examples illustrate the notion of observation schemes. More examples can be found later in the paper.

Example 1. (Input/Output observation scheme)

Let \( IO_p = \{ x, y \} \) for all \( x \in D_p \) and \( y \in p(x) \), where \( y \in p(x) \) means that \( y \) is a possible output of concurrent system \( p \) when executed on input data \( x \). The universe of observable phenomena is defined to be the power set of \( IO_p \), and the partial ordering be set inclusion. The recording function \( \mu_p(T) \) is defined to be the collection of sets of input/output pairs observable from testing \( p \) on \( T \).

For instance, assume that \( D_p = \{0,1\} \), \( p(1) = \{1\} \), and \( p(0) = \{0,1\} \) due to non-determinism. Let test data \( t=0 \), and test set \( T = \{t\} \), then \( \mu_p(T) = \{\{<0,0>\}, \{<0,1>\}\} \), i.e. one may observe either \( \{<0,0>\} \) or \( \{<0,1>\} \) by executing \( p \) on input 0 once. Let test set \( T = \{t\} \), then \( \mu_p(T) = \{\{<0,0>\}, \{<0,1>\}, \{<0,0>, <0,1>\}\} \). i.e. one of the following three different phenomena can be observed by executing \( p \) twice on the same input 0:
1) \( \{<0,0>\} \) - \( p \) outputs 0 in two executions on input 0;
2) \( \{<0,1>\} \) - \( p \) outputs 1 in two executions on input 0;
3) \( \{<0,0>, <0,1>\} \) - \( p \) outputs 0 in one execution, and 1 in another execution on the same input 0.

Example 2. (Dead mutant observation scheme)

Consider the observation scheme for mutation testing [28–30]. Let \( \Phi \) be a set of mutation operations. The application of \( \Phi \) to a program \( p \) produces a set of mutants of \( p \). Let \( \Phi(p) \) be the set of such mutants that are not equivalent to \( p \). Define the universe of phenomena to be the power set of \( \Phi(p) \). The partial ordering is defined to be the set inclusion relation. For all test sets \( T \), the recording function \( \mu_p(T) \) is defined to be the collection of sets of mutants. Each element in \( \mu_p(T) \) is a set of mutants that can be killed by one test of \( p \) on \( T \).

Example 3. (Output diversity observation scheme)

The observation scheme in this example records the number of different outputs on each input data. A phenomenon observable from testing a concurrent system on a set of test cases consists of a set of records. Each record has two parts \( <t, n> \), where \( t \) is a valid input, \( n \) is the number of different outputs on the input data observed from the testing. Formally, an element of the universe of phenomena is a set in the form of \( \{<t_i, n_i> | t_i \in D_p, n_i \geq 0, \forall i \in I\} \). The partial ordering relation on phenomena is defined as follows:

\( \sigma \leq \sigma' \Leftrightarrow \forall <t, n> \in \sigma \exists <t', n'> \in \sigma'. (t = t' \land n \leq n') \).

The least upper bound of \( \sigma_1 \) and \( \sigma_2 \) is a set, written as \( \sigma_1 + \sigma_2 \), which contains elements in the form of \( <t, n> \), and we have that

\( a \in t, \max(n_1, n_2) >\in \sigma_1 + \sigma_2 \).
if $\exists n_1, n_2 > 0,\langle t, n_1 \rangle \in \sigma_1 \land \langle t, n_2 \rangle \in \sigma_2$;
(b) $\langle t, n_1 \rangle \in \sigma_1 + \sigma_2$,
if $\exists n_2, (\langle t, n_1 \rangle \in \sigma_1) \land \neg \exists n_2 > 0,\langle t, n_2 \rangle \in \sigma_2$;
(c) $\langle t, n_2 \rangle \in \sigma_1 + \sigma_2$,
if $\exists n_2, (\langle t, n_2 \rangle \in \sigma_2) \land \neg \exists n_2 > 0,\langle t, n_1 \rangle \in \sigma_1$.

2.2. Properties of schemes

Having defined the notion of observation schemes, we discuss what is a good observation scheme in this section. We first propose some desirable properties of observation schemes and then study the interrelationships between the properties.

2.2.1. Well-founded schemes

The first group of properties to be discussed here is related to whether non-trivial phenomena can be observed from non-empty tests. Notice that, testing a concurrent system on the empty test set means that the concurrent system is not executed at all. Therefore, nothing can be observed. Considering the phenomena observable from empty testing as the least elements in the universe of observable phenomena, such an intuition can be formalised in different ways as follows.

Lower least element property: Any phenomenon $\sigma$ observable by the empty testing is contained in a phenomenon observed from some non-empty test. Formally, $\forall \sigma \in \mu_p(\emptyset) \exists \sigma' \in \mu_p(T). (\sigma' \supseteq \sigma)$.

Middle least element property: Testing on any test set can reveal all phenomena observable from the empty testing. Formally, $\exists \sigma \in \mu_p(T) \forall \sigma' \in \mu_p(\emptyset). (\sigma \supseteq \sigma')$.

Upper least element property: Any phenomenon observable from testing on any test set contains information of some phenomenon observable from empty testing. Formally, $\forall \sigma \in \mu_p(T) \exists \sigma' \in \mu_p(\emptyset). (\sigma \supseteq \sigma')$.

All of these properties state that the empty test set is the weakest test set in terms of the observable phenomena. The following observability requires that the weakest phenomenon observable by the empty testing be the least element of the CPO set.

Observability: If a concurrent system is tested on valid inputs, some non-trivial phenomenon of the system’s behaviour can always be observed, but nothing can be observed from the empty testing. Formally, observability is defined by the following formula.

$\forall p \in P.( T \cap D_p \neq \emptyset \Rightarrow \bot_p \in \mu_p(T)) \land$
$\forall p \in P.( T \cap D_p = \emptyset \Rightarrow \mu_p(T) = \bot_p )$.

Another property related to non-trivial phenomena is the following domain limited property.

Domain limited property: Only valid inputs affect the phenomena observable from a testing. Formally, $\forall p \in P.( \mu_p(T) = \mu_p(T \cap D_p))$.

2.2.2. Incremental schemes

The testing process is often incremental. The second group of properties about observation schemes states the relationship between the phenomena observable from testing on a test set and its subsets.

Extendibility: Every phenomenon observable from executing a concurrent system on a test set is a part of a phenomenon observable from its superset. Formally, $\forall p \in P.( \sigma \in \mu_p(T) \land T \subseteq T' \Rightarrow \exists \sigma' \in \mu_p(T'). (\sigma \subseteq_p \sigma')$.

Tractability: Every phenomenon observable from executing a concurrent system on a test set $T$ contains a phenomenon observable from a subset $T'$ of $T$. Formally, $\forall p \in P.( \sigma \in \mu_p(T) \land T \supseteq T' \Rightarrow \exists \sigma' \in \mu_p(T'). (\sigma \geq_p \sigma')$.

Repeatability: Every phenomenon observable from executing a concurrent system $p$ on a test set $T$ can be observed from executing $p$ on the same test cases in the test set $T$ for more times. Formally, $\forall p \in P.( \sigma \in \mu_p(T) \land T' \in \text{bag}(T) \land T' \supseteq T \Rightarrow \sigma \in \mu_p(T') )$.

2.2.3. Compositional schemes

A test task can often be divided into several subtasks and performed separately. Such a testing strategy can be considered as testing on several subsets of test cases, in which observations are made independently. The question is what is the relationship between the observations made on subsets and the observation that can be made if the testing is not divided into subsets. The third group of properties about schemes is concerned with the relationships between the phenomena observable from testing on different test sets.

The consistency requires that a phenomenon observable from one test set be consistent with any phenomenon observable from another test set if both are obtained from the same program. As in denotational semantics, two phenomena are defined to be consistent, if they have a common upper bound in the CPO set.

Consistency: For any given concurrent system, any two phenomena that are observed from two tests of the system must be consistent. A set $\Gamma$ of phenomena is said to be consistent with a set $\Gamma'$ of phenomena, written as $\Gamma \uparrow \Gamma'$, if for all $\sigma \in \Gamma$ and all $\sigma' \in \Gamma'$, $\sigma \cap \sigma'$.

$\forall p \in P.( \mu_p(T) \uparrow \mu_p(T') )$.

It should be noticed that a concurrent system might well produce different outputs on one test case in two test executions. Whether such phenomena are considered as consistent depends on the definition of the universe of
phenomena. For example, the definition of the input/output observation scheme given in Example 1 considers such phenomena as consistent. However, they are considered as inconsistent if the definition of $IO_p$ in Example 1 is replaced with $IO'_p = \{ < t, f(t) > | t \in D_p \}$, where $f$ is a function on $D_p$. In fact, using $IO'$ to define the universe of phenomena excludes the possibility of non-determinism. Hence, different outputs on the same input indicate inconsistency.

Suppose that a testing consists of executions of a concurrent system on several subsets of test cases. The phenomenon observed from the whole testing should contain all the information obtained from the executions on the subsets. This property is called completeness.

**Completeness:** Every phenomenon observable from testing a concurrent system on a subset is contained in a phenomena observable from testing on the superset. Formally,

$$\forall p \in P \forall \sigma \in \mu_p(T_i) \exists \sum \sigma = \sum \mu_p \left( \bigcup_{i=1}^{n} T_i \right) \sigma \geq \sigma_i$$.

Once a testing task is divided into a number of subtasks and a number of phenomena are observed, an overall results of the testing must be derived. The following property requires that the summation of the phenomena observed from individual subtasks be a phenomenon observable if the testing is not divided into subtasks.

**Composability:** The phenomena observable by executing a concurrent system $p$ on a number of test sets can be put together to form a phenomenon that is observable from executing $p$ on the union of the test sets. Formally,

$$\forall p \in P \forall \sum \sigma \in \mu_p \left( \bigcup_{i=1}^{n} T_i \right) \sigma \geq \sum \mu_p \left( \bigcup_{i=1}^{n} T_i \right) \sigma$$.

Composability means that the summation operation is safe in the sense that the summation of the phenomena observable from a number of test sets is still observable if the testing is not divided into subtasks. The decomposability defined below states another kind of safety. It means that dividing a testing task into smaller subtasks will not loss the possibility of observing a phenomenon. Unfortunately, some schemes that look sound at first sight do not have decomposability as we can see later.

**Decomposability:** For all test sets $T$ and its partitions into subsets, every phenomenon observable from executing a concurrent system $p$ on the test set $T$ can be decomposed into the summation of the phenomena observable from the subsets of the partition. Formally, let $T = \bigcup_{i=1}^{n} T_i$,

$$\forall p \in P \forall \sigma \in \mu_p(T) \exists \sum \sigma = \sum \mu_p \left( \bigcup_{i=1}^{n} T_i \right) \sigma$$.

For example, the input/output observation scheme defined in Example 1 satisfies all the properties defined above. The dead mutant observation scheme defined in Example 2 has repeatability, consistency, completeness, composability, extendibility, tractability, and decomposability. Its observability and domain limited property depend on the mutation operations. For some mutation operations, a non-empty test set may kill no mutants. It may also have no domain limited property because an invalid input for the original program may be valid for a mutant, hence it kills the mutant. The output diversity observation scheme defined in Example 3 has observability, repeatability, consistency, completeness, composability, extendability, tractability and domain limited property. However, it does not have decomposability.

Figure 1 summarises the implication relationships between properties of schemes proved in [25].

### 2.3. Extraction relation between schemes

An important question in the study of observation schemes is what is the relationship between the observable phenomena when using different schemes. From a phenomenon observed under a given scheme, one can often derive what is observable under another schemes.
For example, we can derive the set of executed statements from the set of executed paths. The following extraction relation formally defines such relationships between observation schemes.

Let \( A: p \rightarrow< A_p, \mu^A_p > \) and \( B: p \rightarrow< B_p, \mu^B_p > \) be two schemes.

**Definition 2. (Extraction Relation between Schemes)**

Scheme \( A \) is an extraction of scheme \( B \), written \( A \prec B \), if for all \( p \in P \), there is a homomorphism \( \varphi_p \) from \( < B_p, \leq_{B_p} > \) to \( < A_p, \leq_{A_p} > \), such that (1) \( \varphi_p(\sigma) = \perp_{A_p} \) if and only if \( \sigma = \perp_{B_p} \) and (2) for all test sets \( T \), \( \mu^A_p(T) = \varphi_p(\mu^B_p(T)) \).

Informally, scheme \( A \) is an extraction of scheme \( B \) means that scheme \( B \) observes and records more detailed information about dynamic behaviours than scheme \( A \) does. The phenomena that scheme \( A \) observes can be extracted from the phenomena that \( B \) observes.

The extraction relation is a partial ordering on observation schemes.

**Definition 3. (Equivalence Relation on Schemes)**

Scheme \( A \) and scheme \( B \) are said to be equivalent, written \( A \sim B \), if there is an isomorphism \( \varphi \) from \( A \) to \( B \) such that for all test sets \( T \), \( \varphi_p(\mu^A_p(T)) = \mu^B_p(T) \).

The \( \sim \) relation is an equivalence relation on observation schemes. It also preserves the properties of schemes discussed in the previous section [25].

### 3. Constructions of schemes

This section provides a number of constructions of observation schemes and investigates their properties. Each construction can be considered as an abstract model of observation schemes of common features. The constructions studied in the section enable us to understand the strengths and weaknesses of testing methods from a very high level of abstraction and to develop testing methods according to the desired features.

The proofs of the properties of the constructions are omitted for the sake of space. Readers are referred to [25] for details.

**3.1. Set construction**

In statement testing, software testers observe and record the subset of statements in the software source code that are executed, see e.g. [31, 32]. In this observation scheme, the execution of a statement is an atomic event to be observed, and the universe of phenomena consists of all the sets of such events. The partial ordering on phenomena is just set inclusion. Such a construction of scheme is common to many testing methods. The following is a formal definition of this construction.

**Definition 4. (Regular Set Scheme)**

Scheme \( B: p \rightarrow< B_p, \mu^B_p > \) is said to be a regular set scheme (or simply regular scheme) with base \( U_{p \in P} \), if for all \( p \in P \), the elements in the CPO \( < B_p, \subseteq_p > \) are subsets of \( U_p \) and the partial ordering \( \subseteq_p \) is the set inclusion relation \( \subsetneq \). Moreover, the following conditions hold for the mapping \( \mu_p \):

1. \( U_p = \bigcup_{t \in D_p} (\bigcup_{\mu_p((t))} \).
2. \( \mu_p(\emptyset) = \emptyset \).
3. \( T \cap D_p \neq \emptyset \Rightarrow \emptyset \notin \mu_p(T) \).
4. \( \mu_p(T) = \mu_p(T \cap D_p) \).
5. \( \mu_p \left( \bigcup_{i \in I} T_i \right) = \bigcup_{i \in I} \mu_p(T_i), i \in I \).

**Lemma 1.**

Let \( B: p \rightarrow< B_p, \mu^B_p > \) be an observation scheme. If \( B \) is a regular scheme, then we have that

1. \( B \) has observability;
2. \( B \) has domain limited property;
3. \( B \) has composability;
4. \( B \) has decomposability.

**Theorem 1. (Extraction Theorem for Regular Schemes)**

Let \( B: p \rightarrow< B_p, \mu^B_p > \) be a regular scheme. Let \( A: p \rightarrow< A_p, \mu^A_p > \). Assume that for all \( p \in P \), there is a set \( U^A_p \) such that \( < A_p, \subseteq_p > \) is a CPO on subsets of \( U^A_p \) with set inclusion relation \( \subsetneq \). If for all \( p \in P \), there is a surjection \( f_p \) from \( U^A_p \) to \( U^B_p \) such that:

(a) \( \sigma_A \in A_p \Leftrightarrow \exists \sigma_B \in B_p : (\sigma_A = \{ f_p(x) | x \in \sigma_B \}) \), or in short, \( A_p \parallel f_p(B_p) \), and
(b) for all test sets \( T \), \( \mu^A_p(T) = \{ f_p(\sigma) | \sigma \in \mu^B_p(T) \} \),

then, we have that:

1. \( A \) is a regular scheme with base \( U^A \), and
2. \( A \) is an extraction of \( B \).

We say that \( A \) is the regular scheme extracted from \( B \) by the extraction mapping \( f_p \).

**3.2. Partially ordered set construction**

In the set construction, there is no ordering relationship between the basic events to be observed. However, in some testing methods such as path testing, the basic events are ordered by a partial ordering.

Let \( X \) be a non-empty set and \( \leq \) be a partial ordering on \( X \). A subset \( S \subseteq X \) is said to be downward closed if for all \( x \in S, y \leq x \Rightarrow y \in S \). Let \( p \in P \). Given a partially ordered set (also called poset) \( < A_p, \leq_p > \), we define the universe \( B_p \) of phenomena to be the set of downward closed subsets of \( A_p \). The binary relation \( \leq_{B,p} \) on phenomena is defined as follows:

\[ \sigma_1 \leq_{B,p} \sigma_2 \Leftrightarrow \forall x \in \sigma_1, \exists y \in \sigma_2, (x \leq_p y) . \]
It is easy to prove that \( \leq_{B,p} \) is a partial ordering. Moreover, if the poset \( \langle A_p, \leq_{A_p} \rangle \) has a least element \( \bot_p \), the poset \( \langle B_p, \leq_{B,p} \rangle \) is a CPO with the least element \( \{ \bot_p \} \). The least upper bound of \( \sigma_1 \) and \( \sigma_2 \) is \( \sigma_1 \cup \sigma_2 \).

**Definition 5. (Poset Scheme)**

An observation scheme \( B = \langle B, \mu \rangle \) is said to be a partially ordered set scheme (or poset scheme) with base \( \langle A_p, \leq_{A_p} \rangle, p \in P \), if its universe of phenomena is defined as above and the recording function has the following properties:
1. \( \mu_p(\emptyset) = \{ \bot_p \} \).
2. \( T \cap D_p \neq \emptyset \Rightarrow \{ \bot_p \} \in \mu_p(T) \).
3. \( \mu_p(T) = \mu_p(T \cap D_p) \).
4. \( \mu_p(\bigcup_{i \in I} T_i) = \bigcup_{i \in I} \mu_p(T_i), i \in I \).

**Lemma 2.**

A poset scheme has observability, domain limited property, decomposability, and composability.

**Example 4. (Observation scheme for path testing [31–33])**

Let \( p \) be any given program. A path in \( p \) is a sequence of statements in \( p \) executed in the order. Let \( A_p \) be the set of paths in \( p \), and the partial ordering \( \leq_p \) be the sub-path relation. Let \( s \) be a set of paths in \( p \). The downward closure of \( s \) is the set of sub-paths covered by \( s \), written as \( \overline{s} \). Let \( T \) be a test set. Define:
\[
\mu_p(T) = \{ \overline{s_{r,p}} \mid s_{r,p} \text{ is a set of execution paths in } p \text{ that may be executed on } T \}.
\]

It is easy to see that the function defined above satisfies the conditions (1)–(4) in the definition of the poset scheme. Therefore, by Lemma 2, it has observability, domain limited property, composability and decomposability.

Similar to Example 4, we can define observation schemes that observe the sequences of a type of events happened during test executions of a system, such as the sequences of communication and synchronisation events. Such schemes have the same property as the scheme for path testing.

### 3.3. Product construction

Given two schemes \( A \) and \( B \), we can define a new scheme from them by including both of the information observed by the schemes. The following defines the product scheme of \( A \) and \( B \).

**Definition 6. (Product Construction)**

Let \( A \colon p \mapsto \langle A_p, \mu_p^A \rangle \) and \( B \colon p \mapsto \langle B_p, \mu_p^B \rangle \). The scheme \( C \colon p \mapsto \langle C_p, \mu_p^C \rangle \) is said to be the product of \( A \) and \( B \), written \( C = A \times B \), if for all \( p \in P \),

1. \( C_p = \langle \langle \sigma_{i_A}, \sigma_{i_B} \rangle \mid \sigma_{i_A} \in A_p, \sigma_{i_B} \in B_p \rangle, \leq_{C,p} \rangle \), where
   \[
   \langle \sigma_{i_A}, \sigma_{i_B} \rangle \leq_{C,p} \langle \sigma'_{i_A}, \sigma'_{i_B} \rangle \iff (\sigma_{i_A} \leq_{A,p} \sigma'_{i_A}) \land (\sigma_{i_B} \leq_{B,p} \sigma'_{i_B}) \]
   \[
   \lor (\sigma_{i_A} <_{A,p} \sigma'_{i_A}) \land (\sigma_{i_B} <_{B,p} \sigma'_{i_B});
   
   \]
   \[
   (2) \text{ for all test sets } T, \mu_p^C(T) = \mu_p^A(T) \times \mu_p^B(T).
   \]

**Theorem 2.**

Let \( A \colon p \mapsto \langle A_p, \mu_p^A \rangle \) and \( B \colon p \mapsto \langle B_p, \mu_p^B \rangle \) be two schemes. We have that:
1. if both \( A \) and \( B \) have observability, so does \( A \times B \);
2. if both \( A \) and \( B \) have domain limited property, so does \( A \times B \);
3. if both \( A \) and \( B \) have consistency, so does \( A \times B \);
4. if both \( A \) and \( B \) have completeness, so does \( A \times B \);
5. if both \( A \) and \( B \) have extendibility, so does \( A \times B \);
6. if both \( A \) and \( B \) have tractability, so does \( A \times B \);
7. if both \( A \) and \( B \) have repeatability, so does \( A \times B \);
8. if both \( A \) and \( B \) have composability, so does \( A \times B \);
9. if both \( A \) and \( B \) have decomposability, so does \( A \times B \).

The following lemma states that product construction has associativity.

**Lemma 3.** For all schemes \( A, B \) and \( C \), we have that \( A \times (B \times C) = (A \times B) \times C \).

**Example 5. (Typed dead mutant observation scheme)**

In Example 2, an observation scheme is defined for mutation testing. In software testing tools, mutation operators are often divided into a number of classes to generate different types of mutants, see e.g. [34]. Dead mutants of different types are then recorded separately to provide more detailed information. To define the observation scheme for this, let \( \Phi_1, \Phi_2, ..., \Phi_n \) be sets of mutation operators. For each \( \Phi_i, i=1, 2, ..., n \), define a dead mutant observation scheme \( M_i \) as in Example 2. Then, we define the typed dead mutant observation scheme \( M_{\text{Typed}} = M_1 \times M_2 \times ... \times M_n \).

### 3.4. Statistical constructions

An observation scheme in the set construction or partially ordered set construction observes and records whether certain types of events happen during testing process. Another type of observations often used in software testing is the statistics of the number or frequency of certain events happened in testing. This subsection discusses such type of observation schemes.

Let \( B \colon p \mapsto \langle B_p, \mu_p^B \rangle \) be an observation scheme. \( N \) be any given set of numbers. Then, \( \langle N, \leq \rangle \) is a totally ordered set under the less than or equal to relation \( \leq \) on numbers. We can define a scheme \( A \colon p \mapsto \langle A_p, \mu_p^A \rangle \) as follows.

**Definition 7. (Statistical Construction)**

A scheme \( A \colon p \mapsto \langle A_p, \mu_p^A \rangle \) is said to be a statistical observation scheme based on \( B \colon p \mapsto \langle B_p, \mu_p^B \rangle \), if there
exists a set $N$ of numbers and a collection of mappings $s_{p\in P} : B_p \rightarrow N$ such that

1. For all $p \in P$, $A_p \equiv N$, and $\leq_{A_p}$ is the less than or equal to relation on $N$;
2. For all $p \in P$, the mapping $s_p$ from $B_p$ to the set $N$ preserves the orders in $B_p$, i.e.,
   $$\sigma \leq_{B_p} \sigma' \Rightarrow s_p(\sigma) \leq s_p(\sigma')$$;
3. For all test sets $T$, $\mu^s_p(T) = \{s_p(\sigma) | \sigma \in \mu_p(T)\}$.

Informally, the observable phenomena in a statistical construction are numerical values ordered as numbers. The mapping $s_p$ can be considered as the measurement of sizes of the phenomena observed by the base scheme. This measurement of the size must be consistent with the ordering on the phenomena in the base scheme. In other words, the more information is contained in a phenomenon observed by the base scheme, the larger the size of the phenomenon should be. For example, statement coverage is a statistical construction based on the statement testing scheme.

**Example 6. (Statement coverage)**

Let $B : p \rightarrow <B_p, \mu_p^{\text{sp}}>$ be the regular scheme for statement testing. Define $s_p(\sigma) = ||\sigma|| / n_p$, where $n_p$ is the number of statements in program $p$, $||\sigma||$ is the size of the set $\sigma$. We thus define a statistical observation scheme for statement coverage. The phenomena observed by the scheme are the percentage of statements executed during testing.

**Example 7. (Mutation score)**

In mutation testing, mutation score is defined by the following equation and used as an adequacy degree of a test set [28, 29].

$$\text{Mutation Score} = \frac{\text{number of dead mutants}}{\text{number of non-equivalent mutants}}$$

We define the mutation score as a statistical observation scheme based on the dead mutant observation scheme with the mapping $s_p(\sigma) = ||\sigma|| / m_p$, where $||\sigma||$ is the size of the set $\sigma$ and $m_p$ is the number of non-equivalent mutants of $p$ generated by the set of mutation operators.

Notice that the statement coverage scheme defined above is not decomposable, although the observation scheme for statement testing is regular, which has decomposability according to Lemma 1. Similarly, the mutation score scheme does not have decomposability while the dead mutation scheme has decomposability. The examples show that the space of statistical information observable from testing separately on several smaller test sets may be smaller than the space observable from a large test set.

In software testing, statistics can be also made on the phenomena observed from testing on each test case. The following defines the general construction of such schemes.

**Definition 8. (Case-Wise Statistical Construction)**

A scheme $A : p \rightarrow \langle A_p, \mu_p^{\text{sp}} \rangle$ is said to be a case-wise statistical observation scheme based on $B : p \rightarrow <B_p, \mu_p^{B_p}>$, if there exists a set $N$ of numbers and a collection of mappings $s_{p\in P} : B_p \rightarrow N$ such that

1. For all $p \in P$, $A_p \equiv N$, where $D_p \rightarrow N$ is the set of partial functions from $D_p$ to $N$, and $\leq_{A_p}$ is defined by the following equation:
   $$\sigma_1 \leq_{A_p} \sigma_2 \iff \forall i \in D_p, (\sigma_1(i) = \text{undefined} \lor \sigma_1(i) \leq \sigma_2(i))$$,
   where $\leq$ is the less than or equal to relation on $N$;
2. For all $p \in P$, the mapping $s_p$ from $B_p$ to the set $N$ preserves the order in $B_p$, i.e.,
   $$\sigma \leq_{B_p} \sigma' \Rightarrow s_p(\sigma) \leq s_p(\sigma')$$;
3. For all test sets $T = \{n_i | i \in I\}$ where $i \neq j \Rightarrow t_i \neq t_j$, we have that $\sigma_t \in \mu^s_p(T)$, iff
   (a) $\forall i \in I, \exists \sigma_i \in \mu^s_p(\{n_i\}), (\sigma_t(i) = s_p(\sigma_i))$, and
   (b) $t \in T \Rightarrow \sigma_t(i) = \text{undefined}$.

Informally, a phenomenon in the universe $A_p$ consists of a sequence of records. Each record represents the size of the phenomenon observed using the base scheme from the execution(s) of the concurrent system $p$ on one test case. As in the statistical construction, the size function $s_p$ must be consistent with the partial ordering relation defined on the base scheme.

Notice that if the base scheme of a case-wise statistical scheme has composability and decomposability, the case-wise statistical information can be derived from the phenomena observed by using the base scheme. However, the case-wise observation scheme may have properties different from its base scheme. The following is such an example.

**Example 8.**

The output diversity observation scheme defined in Example 3 is the case-wise statistical observation scheme based on the input/output observation scheme with the mapping $s_p$ being the set size function.

4. Concluding remarks

In this paper, we reviewed our previous results on a theoretical framework for studying testing methods for both sequential and concurrent systems. The main results of the framework include a formal definition of the notion and a set of the desirable properties of an observation scheme. In this paper, we provide several constructions of observation schemes that have direct implications in current software testing practice. We also study the properties of the constructions. Each construction is an abstraction of the ways that observations are defined in testing methods. These constructions enable us to understand the properties of the existing testing methods from a higher level of abstraction and to develop new
testing methods of desired properties.

Currently, we are investigating constructions reflecting software architectures and supporting integration testing methods.

Acknowledgements

This work is jointly funded by the National Science Foundation of the USA under grant INT-9731620 and the National Science Foundation of China under grant 69811120643. X. He was also partially supported by the Office of Naval Research of the USA under grant N00014-98-1-0591.

References


