A Theory of Behaviour Observation in Software Testing

Hong Zhu

School of Computing and Mathematics, Oxford Brookes University,
Gipsy Lane, Headington, Oxford, OX3 0BP, UK, Email: hzhu@brookes.ac.uk

Xudong He

School of Computer Science, Florida International University,
University Park, Miami, FL 33199, Email: hex@cs.fiu.edu

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 software testing theories 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. This paper presents a theory of behaviour observation in software testing. We first introduce and formally define the notion of observation scheme that characterises systematic and consistent methods of behaviour observations and recordings. We propose a set of desirable properties for observation schemes and study the relationships among the properties. We provide several constructions of observation schemes that have direct implications in current software testing practice. We then explore the relationships between different observation schemes and examine these observation schemes with regard to the desirable properties. Finally, we apply the theory to a concrete computation model for concurrent systems, high-level Petri nets, and demonstrate how to use observation scheme to define test adequacy criteria.

Keywords: software testing, behaviour observation, concurrent systems, Petri nets, complete partially ordered set.

1. Introduction

Software testing is a process in which a software system's dynamic behaviours are observed, recorded and analysed so that properties of the system can be inferred from the information revealed by test executions. The past decades have seen a rapid growth in the study of testing methods, see e.g. [ZHM97] for a survey. The most well known testing methods include structural testing, functional testing, and fault-based testing. These testing methods not only differ from each other in the way that test cases are selected or generated, but also in the way that the dynamic behaviour of the software under test are observed and analysed. For example, in statement testing, executions of the statements in program source code are observed and recorded for analysing the percentage of statements tested. In path testing methods, observations are made and recorded to analyse various types of paths executed during testing. In mutation testing, what observed is the liveness of mutants, i.e. whether or not a mutant produces a result different from that of the software under test. These behaviour observation and recording methods are usually implemented through software instrumentation. It is no doubt that how to observe system's behaviour during the testing process is a very important issue of testing methods. No matter how dynamic behaviour is observed and recorded, it should be made consistently and systematically.

Unfortunately, few existing theories have addressed the important issue of behaviour

observation in software testing. Instead, existing theories of software testing assume explicitly or implicitly that a software system can only demonstrate one dynamic behaviour on one test set and the test cases uniquely determine the dynamic behaviour of the program under test. For example, test criteria were defined as a function or predicate on test sets, e.g. [GG75, BuA82, Wey86, ChS87, DaW88 Wey88, PZ91, PZ93, FW93, ZH93]. This assumption is sound for testing sequential software. However, testing a concurrent system on the same test case twice may reveal different behaviours due to non-determinism. The uniqueness assumption on software's dynamic behaviour hence significantly limits the application of existing software testing theories. This paper addresses the complex of non-determinism and concurrency in software testing while sequential software systems are treated as special cases.

This paper proposes a theory that enables us to study the properties of various ways of behaviour observations and the relationships between them. It is organised as follows. Section 2 introduces the notion of observation schemes and the preliminary concepts and notations used in the paper. Section 3 discusses various properties of observation schemes. Section 4 studies the extraction relation between observation schemes. Section 5 proposes a number of schemes constructions as abstractions of testing methods. Section 6 applies the theory to the testing of Petri net. A number of observation schemes are defined and analysed. Section 7concludesthe paper with a discussion of related and future work.

2. The Notion of Observation Schemes

The recorded observations of system's dynamic behaviour during a testing can be (1) the set of executed statements, (2) the set of exercised branches, (3) the set of traversed paths, (4) the sequences of communications occurred between processes, or (5) the set of dead mutants, and so on. To make the behaviour observation and recording meaningful, we require that the observation and recording 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 testing 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 phenomena on systems' dynamic behaviours that are observable from testing. This universe must have the structure of a complete partially ordered set for the following reasons.

(1) *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 in general not ordered. Therefore, all possible observations made during testing a software system form a partial order.

(2) *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 is the least upper bound of the individual observations.

(3) *Existence of the least element*. No information about the dynamic behaviour of a system can be observed if the system is not executed. To identify such a situation, the universe of phenomena is required to contain the least element denoting "no information".

For example, let an observation of the dynamic behaviour of a system be the 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 least element is the empty set. The summation operation is set union.

Before we formally define the notion of schemes and investigate their properties, we first give some mathematical concepts and notations. Readers are referred to [GS90] for details about domain theory.

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 $s_1 \uparrow s_2$.
- *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 $\perp$; (2) 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, $i \in I$. We write $\underset{i \in I}{\exists} x_i \in X_i.Pred$ as a short hand for $\exists x_1 \in X_1.\exists x_2 \in X_2 \ldots .Pred$. $\underset{i \in I}{\forall} x_i \in X_i.Pred$ is defined similarly.
- **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. $\overleftarrow{Y}$ is the set obtained by removing duplicated elements in a multiple set Y.
- Let φ be a mapping from X to Y. For all subsets $A \subseteq X$, we define $\varphi(A)=\{\varphi(a) \mid a \in A\}$.

We use the following symbols in this paper:

- p and its variants for a concurrent system, and P for the set of all concurrent systems;
- s and its variants for a specification, and S for the set of all specifications;
- D for the set of input data for all concurrent systems, and D_p for the set of valid inputs of p;
- t and its variants for a test case, i.e. an input datum for testing a concurrent system.
- T and its variants for a test set. Generally, T is a multiple set (or bag) so that multiple executions on the same test case can be described.

- σ and its variants for a phenomenon observable during testing a concurrent system, and Γ for a set of phenomena.

We now formally define a scheme as follows.

Definition 1. (Observation Scheme)

A *scheme* of behaviour observation and recording, or simply an *observation scheme*, is an ordered pair $<B, \mu>$, where $B=\{<B_p, \leq_p> \mid p\in P\}$, and $\mu=\{\mu_p \mid p\in P\}$. B is called the *universe of phenomena*. For all concurrent systems $p\in P$, $<B_p, \leq_p>$ is a complete partially ordered set. B_p is called the *universe of phenomena* on p. μ is called the *recording function*. For all concurrent systems $p\in P$, the recording function μ_p maps a test set T to a non-empty consistent subset of B_p.

Informally, each element in B_p is a *phenomenon* observable from testing a concurrent system p. $\sigma_1 \leq_p \sigma_2$ means that phenomenon σ_1 is a part of phenomenon σ_2. The least element $\perp_p$ of B_p denotes that 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 σ is a phenomenon that is observable by an execution of p on test set T.

Example 1. (Input/Output observation scheme)

Let $IO_p = \{<x,y> \mid x\in D_p$ 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 $T_1=\{t\}$, then $\mu_p(T_1)=\{\{<0,0>\}, \{<0,1>\}\}$, i.e. one may observe either $\{<0,0>\}$ or $\{<0,1>\}$ by executing p on input 0 once. Let test set $T_2=\{2t\}$, then $\mu_p(T_2)=\{\{<0,0>\}, \{<0,1>\}, \{<0,0>, <0,1>\}\}$, 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 output 0 in two executions on input 0;

(2) $\{<0,1>\}$ - p output 1 in two executions on input 0;

(3) $\{<0,0>, <0,1>\}$ - p output 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 [DLS78, Bud81, How82]. Let Φ be a set of mutation operations. The application of Φ 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 testing 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 $\{<t_i,n_i> \mid t_i \in D_p,\ n_i>0,\ i \in I\}$. The partial ordering relation on phenomena is defined as follows:

$$\sigma \le \sigma' \Leftrightarrow \forall <t,n> \in \sigma . \exists <t',n'> \in \sigma' . (t = t' \wedge n \le n').$$

The least upper bound of σ and σ' is, then,

$$\sigma_1 + \sigma_2 = \{<t,n> \mid <t,n> = \begin{cases} <t, \max(n_1,n_2)>, & if\ \ \exists n_1,n_2 > 0.(<t,n_1> \in \sigma_1 \wedge <t,n_2> \in \sigma_2) \\ <t,n_1>, & if\ \ \exists n_1 > 0.(<t,n_1> \in \sigma_1) \wedge \neg \exists n_2 > 0.(<t,n_2> \in \sigma_2) \\ <t,n_2>, & if\ \ \exists n_2 > 0.(<t,n_2> \in \sigma_2) \wedge \neg \exists n_1 > 0.(<t,n_1> \in \sigma_1) \end{cases} \}$$

3. Properties of Schemes

The definition of scheme provided in the previous section is still insufficient to characterise the notion of consistent and systematic ways of behaviour observation. For example, given a set of mutation operators, a set of mutants of a given program *p* can be generated by applying the mutation operators to *p*. Suppose that the phenomena observed during a mutation testing of *p* is the set of mutants still alive after test. Therefore, one might define the universe of phenomena to be the sets of live mutants and the partial ordering on phenomena to be the set inclusion relation. Our intuition suggests that if an observation σ_1 is made by a test execution, one should be able to have an observation σ_2 such that $\sigma_1 \le \sigma_2$ when carrying out more tests. However, using the above definition it is just the opposite, because the more one tests a program, the less mutants may be alive, i.e. $\sigma_1 \ge \sigma_2$.

Therefore, we further require that an observation scheme satisfy additional properties. The first property to be discussed here is related to whether non-trivial phenomena can be observed from non-empty tests.

Definition 2. (Observability)

An observation scheme is said to have *observability*, if a concurrent system is tested on valid inputs, some phenomenon of the system's behaviour can always be observed, but nothing can be observed from a testing using only invalid input. Formally,

(1) $\forall p \in P.(\ T \cap D_p \neq \varnothing \Rightarrow \perp_p \notin \mu_p(T))$, and

(2) $\forall p \in P.(\ T \cap D_p = \varnothing \Rightarrow \mu_p(T) = \{\perp_p\})$.

The following are some properties weaker than observability. All of them state that the empty test set, which means no testing has been done, is the weakest one in terms of the observable phenomena.

Definition 3. (Lower, Middle and Upper Least Element)

An observation scheme is said to have the *lower least element* property, if for all test set T, $\forall\sigma\in\mu_p(\varnothing).\exists\sigma'\in\mu_p(T).(\sigma'\geq_p\sigma)$. It is said to have the *middle least element* property, if for all test sets *T*, $\exists\sigma\in\mu_p(T).\forall\sigma'\in\mu_p(\varnothing).(\sigma\geq_p\sigma')$. It is said to have the *upper least element* property, if for all test set *T*, $\forall\sigma\in\mu_p(T).\exists\sigma'\in\mu_p(\varnothing).(\sigma\geq_p\sigma')$.

Lemma 1.

(1) The middle least element property implies the lower least element property.

(2) Observability implies the lower least element property, the middle least element and upper lower element property.

Proof. Statement (1) is straightforward by definition. Statement (2) can be proved by condition (2) of observability.

Another property related to when a non-trivial phenomenon can be observed is the domain limited property defined below.

Definition 4. (Domain Limited Property)

An observation scheme is said to have *domain limited property*, if only valid inputs effect the observation. Formally, $\forall p\in P.(\mu_p(T)=\mu_p(T\cap D_p))$.

The second group of properties about schemes is concerned with the relationship between the phenomena observable from testings 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 testing the same program.

Definition 5. (Consistency)

A set Γ of phenomena is said to be *consistent* with a set Γ' of phenomena, written as $\Gamma\uparrow\Gamma'$, if for all $\sigma\in\Gamma$ and all $\sigma'\in\Gamma'$, $\sigma\uparrow\sigma'$. An observation scheme is said to have *consistency*, if for all systems *p*, all test sets *T* and *T'*, the set of phenomena observable from executing *p* on *T* is consistent with the set of phenomena observable from executing *p* on *T'*. Formally, $\forall p\in P.(\mu_p(T)\uparrow\mu_p(T'))$.

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.

A property stronger than consistency is the completeness property.

Definition 6. (Completeness)

An observation scheme is said to have *completeness*, if every phenomenon $\sigma_{i\in I}$ observable

from executing a concurrent system p on a test set $T_{i\in I}$ is contained in the phenomena σ observed from executing p on $\bigcup_{i\in I} T_i$. Formally, $\forall\, p\in P.(\ \forall_{i\in I}\sigma_i \in \mu_p(T_i).\ (\exists\ \sigma\in\ \mu_p.(\bigcup_{i\in I} T_i)\ (\sigma\ \geq_p \sigma_i)))$.

Lemma 2.

Let $T=\{t_i \mid i\in I\}$. If an observation scheme is complete, then we have that

$$\forall\, p\in P.(\ \forall_{i\in I}\sigma_i\in\mu_p(\{t_i\}).\ (\exists\ \sigma\in\ \mu_p(T).(\sigma\geq_p\sigma_i))).$$

Proof. Let $T_i=\{t_i\}$ for $i\in I$ in the definition of completeness.

Lemma 3.

Completeness implies the middle least element property. That is, if an observation scheme has completeness, then for all test sets T, there is $\sigma\in\ \mu_p(T)$ such that, for all $\sigma'\in\ \mu_p(\varnothing)$, $\sigma\geq\sigma'$.

Proof. For all test sets T, we have that $T = T\cup\ \bigcup_{i\in\mu_p(\varnothing)} T_i$, where $T_i=\varnothing$. The statement then follows the definition of completeness immediately.

The third group of properties about schemes states the relationship between phenomena observable from testing on a test set and its subset.

Definition 7. (Extendibility)

An observation scheme is said to have *extendibility*, if every phenomenon observable from executing a concurrent system p on a test set T is a part of a phenomenon observable from executing p on a superset T' of T. Formally,

$$\forall\, p\in P.(\sigma\in\mu_p(T)\wedge T\subseteq T'\Rightarrow\exists\sigma'\in\mu_p(T').(\sigma\leq_p\sigma')).$$

Lemma 4. Extendibility implies the lower least element property.

Proof. Let $T=\varnothing$ in the definition of extendibility.

Lemma 5. Extendibility implies consistency.

Proof. Let T_1 and T_2 be any given test sets. Then, $T = T_1\cup T_2 \supseteq T_1$. Similarly, $T\supseteq T_2$. By extendibility, for all $\sigma_1\in\mu_p(T_1)$ there is $\sigma\in\mu_p(T)$ such that $\sigma\geq\sigma_1$, and for all $\sigma_2\in\mu_p(T_2)$ there is $\sigma'\in\mu_p(T)$ such that $\sigma'\geq\sigma_2$. By the definition of observation schemes, $\mu_p(T)$ is a consistency set, therefore, there is σ^* such that $\sigma^*\geq\sigma$ and $\sigma^*\geq\sigma'$. By the transitivity of $\geq$ relation, we have that $\sigma^*\geq\sigma_1$ and $\sigma^*\geq\sigma_2$. Hence, $\sigma_1\uparrow\sigma_2$. The statement follows from the definition of consistency.

Lemma 6. Completeness implies extendibility.

Proof. Assume that $T\subseteq T'$ and $\sigma\in\mu_p(T)$. Let $T'=T\cup T''$ and $\sigma''\in\mu_p(T'')$. By the definition of completeness, $\sigma+\sigma''\in\mu_p(T')$. The statement follows from the fact that $\sigma+\sigma''\geq\sigma$.

Definition 8. (Tractability)

An observation scheme is said to have *tractability*, if every phenomenon observable from executing a concurrent system *p* on a test set *T* contains a phenomenon observable from executing *p* on a subset *T*' of *T*. Formally,

$$\forall\, p \in P.(\sigma \in \mu_p(T) \wedge T \supseteq T' \Rightarrow \exists \sigma' \in \mu_p(T').(\sigma \geq_p \sigma')) .$$

Lemma 7. Tractability implies the upper least element property, that is, for all test sets *T*, $\forall \sigma \in \mu_p(T).\exists \sigma' \in \mu_p(\varnothing).(\sigma \geq_p \sigma')$ **.**

Proof. Let $\varnothing$ be the *T*' in the definition of tractability.

Definition 9. (Repeatability)

An observation scheme is said to have *repeatability,* if 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,

$$\forall\, p \in P.(\sigma \in \mu_p(T) \wedge T' \in \mathbf{bag}(\tilde{T}) \wedge T' \supseteq T \Rightarrow \sigma \in \mu_p(T')).$$

The last group of properties about schemes states the relationships between the phenomena observable from testing on a number of subsets and the phenomena observable from the union of the subsets. These properties are related to questions like whether a testing task can be divided into a number of subtasks.

Definition 10. (Composability)

An observation scheme is said to have *composability,* if the phenomena observable by executing a concurrent system *p* on a number of test sets $T_{i \in I}$ can be put together to form a phenomenon that is observable from executing *p* on the union of the test sets $T_{i \in I}$. Formally,

$$\forall\, p \in P.(\underset{i \in I}{\forall} \sigma_i \in \mu_p(T_i).(\sum_{i \in I} \sigma_i \in \mu_p(\bigcup_{i \in I} T_i))).$$

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 subsets. The decomposability formally defined below states another kind of safety. It means that dividing a testing task into small subtasks will not loss the possibility of observing a phenomenon. Unfortunately, some schemes that look sound do not have decomposability as we will see later.

Definition 11. (Decomposability)

An observation scheme is said to have *decomposability,* if for all test sets *T*, $T = \bigcup_{i \in I} T_i$ implies that 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 executing *p* on test sets $T_{i \in I}$. Formally, let $T = \bigcup_{i \in I} T_i$,

$$\forall\, p \in P.(\sigma \in \mu_p(T) \Rightarrow \underset{i \in I}{\exists} \sigma_i \in \mu_p(T_i).(\sigma = \sum_{i \in I} \sigma_i)).$$

Lemma 8.

(1) Decomposability implies that

$$\forall\, p\in P.\forall T=\{t_i \mid i\in I\}.(\sigma\in\mu_p(T) \Rightarrow \underset{i\in I}{\exists}\ \sigma_i\in\mu_p(\{t_i\})\ .(\sigma=\sum_{i\in I}\sigma_i\ )),\ (*)$$

(2) If an observation scheme has composability, then (*) implies decomposability.

Proof.

Statement (1) is obvious from the definition of decomposability. To prove (2), let $T=\bigcup_{i\in I}T_i$ and $T_i=\{t_{i,j} \mid j\in J_i\}$. Let $\sigma\in\ \mu_p(T)$. By (*), we have that

$$\underset{i\in I,\, j\in J_i}{\exists}\ \sigma_{i,j}\in\mu_p(\{t_{i,j}\})\ .(\sigma=\sum_{i\in I,\, j\in J_i}\sigma_{i,j}\ )).$$

By composability, we have that for all $i\in I$, $\sum_{j\in J_i}\sigma_{i,j}\in\mu_p(T_i)$, Let $\sigma_i=\sum_{j\in J_i}\sigma_{i,j}$. Then, we have that $\sigma=\sum_{i\in I}\sigma_i$.

The following lemma proves that composability and decomposability are very strong properties.

Lemma 9.

(1) Composability implies completeness;

(2) Decomposability implies tractability;

(3) Composability and decomposability imply repeatability.

Proof.

(1) The statement follows from the fact that for all $i\in I$, $\sum_{i\in I}\sigma_i\geq\sigma_i$.

(2) Assume $T\supseteq T'$ *and* $\sigma\in\mu_p(T)$. Let $T=T'\cup T''$. By the definition of decomposability, $\exists\sigma_1\in\mu_p(T').\exists\sigma_2\in\mu_p(T'').(\sigma=\sigma_1+\sigma_2)$. The statement follows from the fact that $\sigma_1+\sigma_2\geq\sigma_1$.

(3) Assume that $T=\{t_i \mid i\in I\}$, $T'\supseteq T$ and $T'\in\mathbf{bag}(\bar{T})$. Let $\sigma\in\mu_p(T)$. By decomposability, there are $\sigma_i\in\mu_p(\{t_i\})$ for all $i\in I$ such that $\sigma=\sum_{i\in I}\sigma_i$. Since $T'\supseteq T$ and $T'\in\mathbf{bag}(\bar{T})$, there is T^* such that (a) $T'=T\cup T^*$, and (b) for all $t\in T^*$, there is $i\in I$ *such that* $t=t_i\in T$. Let $\sigma_t=\sigma_i$. By composability, $\left(\sigma+\sum_{t\in T^*}\sigma_t\right)\in\mu_p(T')$. The statement follows from the fact that $\sigma+\sum_{t\in T^*}\sigma_t=\sum_{i\in I}\sigma_i+\sum_{t\in T^*}\sigma_t$, and for all σ_i, $\sigma_i+\sigma_i=\sigma_i$.

The relationships between properties of schemes are summarised in Figure 1.

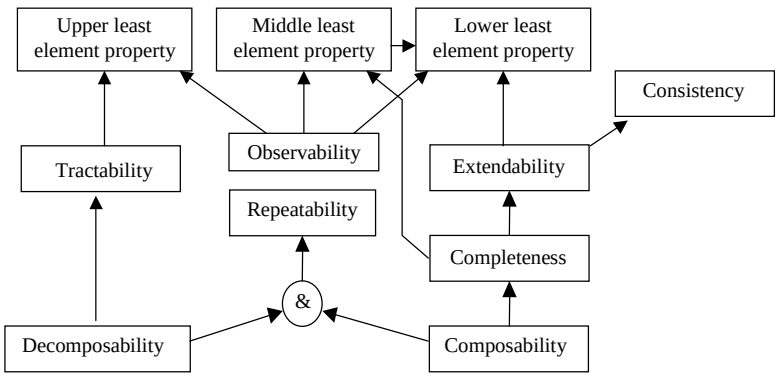

Figure 1. Relationships between properties

The following theorem gives the structure of phenomena observable from finite test sets using a scheme with composability and decomposability.

Theorem 1. (Structure Theorem)

If a scheme B=<*B*, μ> has composability and decomposability, we have

(1) $\mu_p(\{nt\}) = \{\sum \Gamma \mid \Gamma \subseteq \mu_p(\{t\}) \wedge 0 < \|\Gamma\| \le n\}$, for all n>0;

(2) $\mu_p(\{n_1t_1, n_2t_2, \cdots, n_kt_k\}) = \{\sum \Gamma \mid \Gamma = \bigcup_{i=1}^{k} \Gamma_i \wedge \Gamma_i \subseteq \mu_p(\{t_i\}) \wedge 0 < \|\Gamma_i\| \le n_i \wedge 1 \le i \le k\}$.

Proof.

(1) By applying induction on *n*.

- For *n*=1, the right hand side of equation (1) is:

$$\{\sum \Gamma \mid \Gamma \subseteq \mu_p(\{t\}) \wedge \|\Gamma\| = 1\} = \{\sigma \mid \sigma \in \mu_p(\{t\})\} = \mu_p(\{t\})$$

Therefore, the statement is true when n=1.

- Assume that the statement is true for *n*=*N*>0, i.e.

$$\mu_p(\{Nt\}) = \{\sum \Gamma \mid \Gamma \subseteq \mu_p(\{t\}) \wedge 0 < \|\Gamma\| \le N\}. \qquad (*)$$

- For *n*=*N*+1, by the definition of composability, the following equation follows from (*).

$$\mu_p(\{(N+1)t\}) \supseteq \{\sigma + \sigma' \mid \sigma \in \mu_p(\{Nt\}) \wedge \sigma' \in \mu_p(\{t\})\}$$

$$= \{\sum \Gamma \mid \Gamma \subseteq \mu_p(\{t\}) \wedge 0 < \|\Gamma\| \le N+1\}.$$

By the definition of decomposability, we have that

$$\mu_p(\{(N+1)t\}) \subseteq \{\sum_{i=1}^{N+1} \sigma_i | \sigma_i \in \mu_p(\{t\}) \ \wedge \ 1 \le i \le N+1\}$$

$$= \{\sum \Gamma \mid \Gamma \subseteq \mu_p(\{t\}) \wedge 0 < \|\Gamma\| \le N+1\}.$$

Therefore, the statement is also true for $n=N+1$.

According to the principle of mathematical induction, the statement is true for all n>0.

(2) The proof is similar to the proof of statement (1) by applying induction on *k*.

Example 4.

(1) The input/output observation scheme defined in Example 1 satisfies all the properties defined in definitions 2 ~ 11.

(2) The dead mutant observation scheme defined in Example 2 has repeatability, consistency, completeness, composability, extendibility, tractability, and decomposability. Its observability and domain limited property depends 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.

(3) 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.

Proof. The proofs of the statements are straightforward. Details are omitted for the sake of space.

4. Extraction Relation Between Schemes

Given a phenomenon observable from testing of a concurrent system, we can often extract information from the observation. For example, we can extract the set of executed statements from the set of executed paths. This section formally defines such extraction relation between schemes and studies its properties.

Let $\mathsf{A}=\langle A, \mu^A\rangle$ and $\mathsf{B}=\langle B, \mu^B\rangle$ be two schemes.

Definition 12. (Extraction Relation between Schemes)

Scheme A is an *extraction* of scheme B, written $\mathsf{A} \lhd \mathsf{B}$, if for all $p \in P$, there is a homomorphism φ_p from $\langle B_p, \le_{B,p}\rangle$ to $\langle A_p, \le_{A,p}\rangle$, such that (1) $\varphi_p(\sigma)=\perp_{A,p}$ if and only if $\sigma=\perp_{B,p}$, and (2) for all test sets T, $\mu_p^A(T) = \varphi_p(\mu_p^B(T))$.

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.

Definition 13. (Equivalence Relation on Schemes)

Scheme A and scheme B are said to be *equivalent*, written A~B, if there is an isomorphism φ from A to B such that for all test sets T, $\varphi_p(\mu_p^A(T)) = \mu_p^B(T)$.

Obviously, we have that:

Lemma 10. For all schemes A and B, A~B implies that A◁B, and B◁A.

Lemma 11. For all schemes A, B and C, (a) A~A; (b) A~B⇒B~A; (c) A~B∧B~C⇒A~C.

It is also easy to see that the extraction relation is a partial ordering on schemes when equivalent schemes are considered as the same one.

Lemma 12.

For all schemes A, B, and C, we have that:

(1) *Reflexivity*: A◁A;

(2) *Transitivity*: A◁B and B◁C imply that A◁C;

(3) *Anti-symmetry*: A◁B and B◁A imply that A~B.

Proof. Straightforward, by the definitions.

The following theorem proves that extraction mappings preserve the properties of schemes discussed in the previous section.

Theorem 2.

Assume that scheme A=<A, μ^A> is an *extraction* of scheme B=<B, μ^B>. We have that:

(1) if B has observability, so does A;

(2) if B has domain limited property, so does A;

(3) if B has consistency, so does A.

(4) if B has completeness, so does A;

(5) if B has extendibility, so does A;

(6) if B has tractability, so does A;

(7) if B has repeatability, so does A;

(8) if B has composability, so does A;

(9) if B has decomposability, so does A;

Proof. Let φ_p, $p \in P$, be the extraction mappings from B to A.

(1) Let T be any given test set.

(a) Assume that $T \cap D_p \neq \varnothing$. By the observability of B, $\perp_{B,p} \notin \mu_p^B(T)$. Since $\mu_p^A(T) = \varphi_p(\mu_p^B(T))$, and $\varphi_p(\sigma) = \perp_{A,p} \Leftrightarrow \sigma = \perp_{B,p}$, we have that $\perp_{A,p} \notin \mu_p^A(T)$. Therefore, $T \cap D_p \neq \varnothing \Rightarrow \perp_{A,p} \notin \mu_p^A(T)$.

(b) Now assume that $T \cap D_p = \varnothing$. By the observability of B, $\mu_p^B(T) = \{\perp_{B,p}\}$. By the definition of extraction, $\mu_p^A(T) = \varphi_p(\mu_p^B(T)) = \varphi_p(\{\perp_{B,p}\}) = \{\perp_{A,p}\}$. That is, $T \cap D_p = \varnothing \Rightarrow \mu_p^A(T) = \{\perp_{A,p}\}$. Therefore, statement (1) is true.

(2) Let T be any given test set. We have that

$$\mu_p^A(T) = \varphi_p(\mu_p^B(T)) \qquad \text{(by the definition of extraction)}$$

$$= \varphi_p(\mu_p^B(T \cap D_p)) \qquad \text{(by the domain limited property of B)}$$

$$= \mu_p^A(T \cap D_p) \qquad \text{(by the definition of extraction)}$$

(3) Let T and T' be any given test sets. Let $\sigma \in \mu_p^A(T)$ and $\sigma' \in \mu_p^A(T')$. By the definition of extraction, there are $\sigma_B \in \mu_p^B(T)$ and $\sigma'_B \in \mu_p^B(T')$ such that $\sigma = \varphi_p(\sigma_B)$ and $\sigma' = \varphi_p(\sigma'_B)$. By the consistency of B, we have that $\mu_p^B(T) \uparrow \mu_p^B(T')$. Hence, there exists σ_B^* such that $\sigma_B^* \geq_B \sigma_B$ and $\sigma_B^* \geq_B \sigma'_B$. Because φ_p is a homomorphism, we have that $\varphi_p(\sigma_B^*) \geq_A \varphi_p(\sigma_B)$ and $\varphi_p(\sigma_B^*) \geq_A \varphi_p(\sigma'_B)$. Therefore, $\mu_p^A(T) \uparrow \mu_p^A(T')$.

(4) Let $T_{i \in I}$ be a collection of test sets and $\sigma_i \in \mu_p^A(T_i)$, $i \in I$. Because $\mu_p^A(T_i) = \varphi_p(\mu_p^B(T_i)), i \in I$, there are $\sigma_i^B \in \mu_p^B(T_i)$ such that $\sigma_i = \varphi_p(\sigma_i^B)$ for $i \in I$. By the completeness of B, there exists $\sigma_B \in \mu_p^B(\bigcup_{i \in I} T_i)$ such that $\sigma_B \geq_B \sigma_i^B$ for all $i \in I$. Since φ_p is a homomorphism, $\varphi_p(\sigma_B) \geq_A \varphi_p(\sigma_i^B) = \sigma_i$. By the definition of extraction, $\varphi_p(\sigma_B) \in \varphi_p(\mu_p^B(\bigcup_{i \in I} T_i)) = \mu_p^A(\bigcup_{i \in I} T_i)$. Therefore, A is complete.

(5) Let T, T' be test sets and $T \subseteq T'$. Let $\sigma \in \mu_p^A(T)$. By the definition of extraction, there exists $\sigma_B \in \mu_p^B(T)$ such that $\sigma = \varphi_p(\sigma_B)$. By the extendibility of B, there is $\sigma'_B \in \mu_p^B(T')$ such that $\sigma'_B \geq_B \sigma_B$. By the definition of extraction, $\varphi_p(\sigma'_B) \in \mu_p^A(T')$. Since φ_p is a homomorphism, $\varphi_p(\sigma'_B) \geq_A \varphi_p(\sigma_B) = \sigma$. Therefore, A also has extendibility.

(6) Let T, T' be test sets and $T \supseteq T'$. Let $\sigma \in \mu_p^A(T)$. By the definition of extraction, there exists $\sigma_B \in \mu_p^B(T)$ such that $\sigma = \varphi_p(\sigma_B)$. By the tractability of B, there is $\sigma'_B \in \mu_p^B(T')$ such that $\sigma_B \geq_B \sigma'_B$. Since φ_p is a homomorphism, $\varphi_p(\sigma_B) \geq_A \varphi_p(\sigma'_B)$. Therefore, A also has tractability.

(7) Let T be any given test set, and $T' \in \mathbf{bag}(\bar{T})$ and $T \subseteq T'$. Let $\sigma \in \mu_p^A(T)$. By the definition of extraction, there is $\sigma_B \in \mu_p^B(T)$ such that $\sigma = \varphi_p(\sigma_B)$. By the definition of

repeatability, $\sigma_B \in \mu_p^B(T')$. By the definition of extraction, we have that $\sigma = \varphi_p(\sigma_B) \in \varphi_p(\mu_p^B(T')) = \mu_p^A(T')$. Therefore, A also has repeatability.

(8) Let $T_{i\in I}$ be a collection of test sets and $\sigma_i \in \mu_p^A(T_i)$, $i \in I$. Because $\mu_p^A(T_i) = \varphi_p(\mu_p^B(T_i)), i \in I$, there are $\sigma_i^B \in \mu_p^B(T_i)$ such that $\sigma_i = \varphi_p(\sigma_i^B)$. The composability of B implies that $\sum_{i\in I} \sigma_i^B \in \mu_p^B(\bigcup_{i\in I} T_i)$. Because φ_p is a homophormism, $\varphi_p(\sum_{i\in I} \sigma_i^B) = \sum_{i\in I} \varphi_p(\sigma_i^B)$. Therefore, $\sum_{i\in I} \varphi_p(\sigma_i^B) \in \varphi_p(\mu_p^B(\bigcup_{i\in I} T_i)) = \mu_p^A(\bigcup_{i\in I} T_i)$. That is, A also has composability.

(9) Let $T_{i\in I}$ be a collection of test sets. Let $\sigma \in \mu_p^A(\bigcup_{i\in I} T_i)$. By the definition of extraction, there is $\sigma_B \in \mu_p^B(\bigcup_{i\in I} T_i)$ such that $\sigma = \varphi_p(\sigma_B)$. The decomposability of B implies that there exist $\sigma_i^B \in \mu_p^B(T_i)$ for $i \in I$, such that $\sigma_B = \sum_{i\in I} \sigma_i^B$. Since φ_p is a homomorphism, we have that $\varphi_p(\sum_{i\in I} \sigma_i^B) = \sum_{i\in I} \varphi_p(\sigma_i^B)$. Hence, $\sigma = \sum_{i\in I} \varphi_p(\sigma_i^B)$. By the definition of extraction, $\varphi_p(\sigma_i^B) \in \mu_i^A(T_i)$ for all $i \in I$. Therefore, A also has decomposability.

5. Constructions of Schemes

This section provides a number of constructions of observation schemes and investigates their properties.

5.1 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. [Mye79, Bei90]. 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 14. (Regular Set Scheme)

Scheme B=<*B*, μ> 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, $\leq_p$> are subsets of U_p and the partial ordering $\leq_p$ is the set inclusion relation $\subseteq$. Moreover, the following conditions hold for the mapping μ_p:

(1) $U_p = \bigcup_{t\in D_p} (\bigcup \mu_p(\{t\}))$,

(2) $\mu_p(\varnothing) = \{\varnothing\}$,

(3) $T \cap D_p \neq \varnothing \Rightarrow \varnothing \notin \mu_p(T)$,

(4) $\mu_p(T) = \mu_p(T \cap D_p)$,

(5) $\mu_p(\bigcup_{i\in I} T_i) = \{\bigcup_{i\in I} \sigma_i \,|\, \sigma_i \in \mu_p(T_i), i \in I\}$.

Lemma 13.

Let B=<*B*, μ> 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.

Proof.

(1) Because $\varnothing$ is the least element $\perp_p$ in the universe of phenomena, condition (2) and (3) in the definition of regular scheme imply the observability;

(2) Condition (4) is the domain limited property;

(3) Composability and decomposability follow from condition (5) immediately.

Theorem 3. (Extraction Theorem for Regular Schemes)

Let B=<*B*, μ^B> be a regular scheme. Let A=<*A*, μ^A>. Assume that for all $p \in P$, there is a set U_p^A such that <A_p, $\leq_p$> is a CPO on subsets of U_p^A with set inclusion relation $\subseteq$. If for all $p \in P$, there is a surjection f_p from U_p^B to U_p^A 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 $A_p = f_p(B_p)$, and

(b) for all test sets *T*, $\mu_p^A(T) = \{f_p(\sigma) | \sigma \in \mu_p^B(T)\}$,

then we have that:

(1) A is a regular scheme with base U_p^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.

Proof.

(1) To prove statement (1), we first prove that A is a scheme, then prove that A is a regular

scheme with base U_p^A.

(a) To prove that A is a scheme, we only need to prove that for all test sets T, $\mu_p^A(T)$ is consistent. This is proved as follows. Let σ_A and σ'_A in $\mu_p^A(T)$. Then, there is σ_B and σ'_B in $\mu_p^B(T)$ such that $\sigma_A = f_p(\sigma_B)$ and $\sigma'_A = f_p(\sigma'_B)$. The consistency of $\mu_p^B(T)$ implies that there is σ^*_B such that $\sigma_B^* \geq \sigma_B$ and $\sigma_B^* \geq \sigma'_B$. By condition (a), we have that $f_p(\sigma_B^*) \in A_p$. It is easy to see by set theory that $f_p(\sigma_B^*) \geq f_p(\sigma_B)$ and $f_p(\sigma_B^*) \geq f_p(\sigma'_B)$. Therefore, $\mu_p^A(T)$ is consistent.

(c) To prove that A is a regular scheme with base U_p^A, we check that A satisfies conditions (1)~(5) in Definition 14. It follows the definition of A and the regularity of B. Details are omitted here for the sake of space.

(2) It is easy to see that the mapping $\varphi_p(\sigma) = \{f_p(x) | x \in \sigma\}$ is a homomorphism from $<B_p, \leq_{B,p}>$ to $<A_p, \leq_{A,p}>$. The statement follows directly from the definitions.

5.2 Partially Ordered Set Construction

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 a 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 $< A_p, \leq_p>$ has a least element $\perp_p$, the poset $<B_p, \leq_{B,p} >$ is a CPO with the least element $\{\perp_p\}$. The least upper bound of σ_1 and σ_2 is $\sigma_1 \cup \sigma_2$.

Definition 15. (Poset Scheme)

An observation scheme B=$<B, \mu>$ is said to be a *partially ordered set scheme* (or *poset scheme*) with base $<A_p, \leq_p>$, $p \in P$, if its universe of phenomena is defined as above and the recording function has the following properties:

(1) $\mu_p(\varnothing) = \{\{\perp_p\}\}$,

(2) $T \cap D_p \neq \varnothing \Rightarrow \{\perp_p\} \notin \mu_p(T)$,

(3) $\mu_p(T) = \mu_p(T \cap D_p)$,

(4) $\mu_p(\bigcup_{i \in I} T_i) = \{\bigcup_{i \in I} \sigma_i \,|\, \sigma_i \in \mu_p(T_i), i \in I\}$.

Lemma 14.

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

Proof. The observability follows from conditions (1) and (2) immediately. The domain limited property follows from condition (3). Composability and decomposability follow from the fact that the least upper bound of a directed subset of downward closed subsets is the union of the subsets. Details are omitted from here.

Example 5. (Observation scheme for path testing [How76, Mye79, Bei90])

Let p be any given program. A path in p is a sequence of statements in p executed in 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 $\breve{s}$. Let T be a test set. Define:

$\mu_p(T) = \{\breve{s}_{T,p} |\ s_{T,p}$ is a set of execution paths in p that may be executed when testing p 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 14, it has observability, domain limited property, composability and decomposability.

Similar to Example 5, 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.

5.3 Product Construction

Given two schemes A and B, we can define a new scheme from them. The following defines the product scheme of A and B.

Definition 16. (Product Construction)

Let A=<A, μ^A> and B=<B, μ^B>. The scheme C=<C, μ^C> is said to be the product of A and B, written C=A×B, if for all $p \in P$,

(1) $C_p = <\{<\sigma_A, \sigma_B> | \sigma_A \in A_p, \sigma_B \in B_p\}, \leq_{C,p}>$,

where $\left(<\sigma_A, \sigma_B> \leq_{C,p} <\sigma'_A, \sigma'_B>\right) \Leftrightarrow \left(\sigma_A \leq_{A,p} \sigma'_A\right) \wedge \left(\sigma_B \leq_{B,p} \sigma'_B\right)$;

(2) for all test sets T, $\mu_p^C(T) = \mu_p^A(T) \times \mu_p^B(T)$.

Theorem 4.

Let A=<A, μ^A> and B=<B, μ^B> be two schemes. We have that:

(1) if both A and B have observability, so does A×B;

(2) if both A and B have domain limited property, so does A×B;

(3) if both A and B have consistency, so does A×B.

(4) if both A and B have completeness, so does A×B;

(5) if both A and B have extendibility, so does A×B;

(6) if both A and B have tractability, so does A×B;

(7) if both A and B have repeatability, so does A×B;

(8) if both A and B have composability, so does A×B;

(9) if both A and B have decomposability, so does A×B.

Proof. Let scheme C=A×B.

(1) By the definition of product scheme, $\perp_{C,p} =< \perp_{A,p}, \perp_{B,p} >$. The following proves that C satisfies two conditions of observability, respectively.

(a) Assume that $T \cap D_p \neq \varnothing$. The observability of A and B implies that $\perp_{A,p} \notin \mu_p^A(T)$ and $\perp_{B,p} \notin \mu_p^B(T)$. By the definition of the product scheme, $\perp_{C,p} =< \perp_{A,p}, \perp_{B,p} > \notin \mu_p^A(T) \times \mu_p^B(T)$;

(b) Now assume that $T \cap D_p = \varnothing$. By the observability of A and B, $\mu_p^A(T)=\{\perp_{A,p}\}$ and $\mu_p^B(T) = \{\perp_{B,p}\}$. Therefore, $\mu_p^A(T) \times \mu_p^B(T) = \{< \perp_{A,p}, \perp_{B,p} >\} = \{\perp_{C,p}\}$.

(2) Let *T* be any given test set. Then,

$\mu_p^C(T) = \mu_p^A(T) \times \mu_p^B(T)$ (by the definition of the product scheme)

$= \mu_p^B(T \cap D_p) \times \mu_p^B(T \cap D_p)$ (by the domain limited property of A and B)

$= \mu_p^A(T \cap D_p)$ (by the definition of product scheme)

(3) Let *T* and *T'* be any given test sets. Let $\sigma_C \in \mu_p^C(T)$ and $\sigma'_C \in \mu_p^A(T')$. By the definition of the product scheme, there are $\sigma_A \in \mu_p^A(T)$, $\sigma'_A \in \mu_p^A(T')$, $\sigma_B \in \mu_p^B(T)$ and $\sigma'_B \in \mu_p^B(T')$ such that $\sigma_C =< \sigma_A, \sigma_B >$ and $\sigma'_C =< \sigma'_A, \sigma'_B >$. By the consistency of A and B, $\mu_p^A(T) \uparrow \mu_p^A(T')$ and $\mu_p^B(T) \uparrow \mu_p^B(T')$. Hence, there exists σ_A^* such that $\sigma_A^* \geq_A \sigma_A$ and $\sigma_A^* \geq_A \sigma'_A$, and there is σ_B^* such that $\sigma_B^* \geq_B \sigma_B$ and $\sigma_B^* \geq_B \sigma'_B$. Let $\sigma_C^* =< \sigma_A^*, \sigma_B^* >$. Then, $\sigma_C^* \geq_C \sigma_C$ and $\sigma_C^* \geq_C \sigma'_C$. Therefore, $\mu_p^C(T) \uparrow \mu_p^C(T')$.

(4) Let $T_{i \in I}$ be a collection of test sets and $\sigma_i^C \in \mu_p^C(T_i)$, $i \in I$. Because $\mu_p^C(T_i) = \mu_p^A(T_i) \times \mu_p^B(T_i)$ for all $i \in I$, there are $\sigma_i^A \in \mu_p^A(T_i)$ and $\sigma_i^B \in \mu_p^B(T_i)$ such that $\sigma_i^C =< \sigma_i^A, \sigma_i^B >$. By the completeness of A and B, there exists $\sigma_A \in \mu_p^A(\bigcup_{i \in I} T_i)$ such that $\sigma_A \geq_A \sigma_i^A$ for all $i \in I$, and there exists $\sigma_B \in \mu_p^B(\bigcup_{i \in I} T_i)$ such that $\sigma_B \geq_B \sigma_i^B$ for all $i \in I$. Let $\sigma_C =< \sigma_A, \sigma_B >$. Then, $\sigma_C \in \mu_p^A(\bigcup_{i \in I} T_i) \times \mu_p^B(\bigcup_{i \in I} T_i) = \mu_p^C(\bigcup_{i \in I} T_i)$ and $\sigma_C \geq_C \sigma_i^C$ for all $i \in I$. Therefore, C is complete.

(5) Let T, T' be test sets and $T \subseteq T'$. Let $\sigma_C \in \mu_p^C(T)$. By the definition of the product scheme, there exists $\sigma_A \in \mu_p^A(T)$ and $\sigma_B \in \mu_p^B(T)$ such that $\sigma_C =< \sigma_A, \sigma_B >$. By the extendibility of A and B, there is $\sigma'_A \in \mu_p^A(T')$ such that $\sigma'_A \geq_A \sigma_A$, and there is $\sigma'_B \in \mu_p^B(T')$ such that $\sigma'_B \geq_B \sigma_B$. Let $\sigma'_C =< \sigma'_A, \sigma'_B >$. By the definition of the product scheme, $\sigma'_C = < \sigma'_A \sigma'_B > \geq_C < \sigma_A, \sigma_B > = \sigma_C$ and $\sigma'_C \in \mu_p^C(T')$. Therefore, C also has extendibility.

(6) Let T, T' be test sets and $T \supseteq T'$. Let $\sigma_C \in \mu_p^C(T)$. By the definition of the product scheme, there exists $\sigma_A \in \mu_p^A(T)$ and $\sigma_B \in \mu_p^B(T)$ such that $\sigma_C =< \sigma_A, \sigma_B >$. By the tractability of A and B, there is $\sigma'_A \in \mu_p^A(T')$ and $\sigma'_B \in \mu_p^B(T')$ such that $\sigma_A \geq_A \sigma'_A$ and $\sigma_B \geq_B \sigma'_B$. Let $\sigma'_C =< \sigma'_A, \sigma'_B >$. By the definition of the product scheme, $\sigma'_C \in \mu_p^C(T')$ and $\sigma_C \geq \sigma'_C$. Therefore, C also has tractability.

(7) Let T be any given test set, and $T' \in \mathbf{bag}(\bar{T})$ and $T \subseteq T'$. Let $\sigma_C \in \mu_p^C(T)$. By the definition of the product scheme, there are $\sigma_A \in \mu_p^A(T)$ and $\sigma_B \in \mu_p^B(T)$ such that $\sigma_C =< \sigma_A, \sigma_B >$. By the definition of repeatability, $\sigma_A \in \mu_p^A(T')$ and $\sigma_B \in \mu_p^B(T')$. By the definition of the product scheme, $\sigma_C \in \mu_p^A(T') \times \mu_p^B(T') = \mu_p^C(T')$. Therefore, C also has repeatability.

(8) Let $T_{i \in I}$ be a collection of test sets and $\sigma_i \in \mu_p^C(T_i)$, $i \in I$. Because $\mu_p^C(T_i) = \mu_p^A(T_i) \times \mu_p^B(T_i)$ for all $i \in I$, there are $\sigma_i^A \in \mu_p^A(T_i)$ and $\sigma_i^B \in \mu_p^B(T_i)$ such that $\sigma_i =< \sigma_i^A, \sigma_i^B >$. The composability of A and B implies that $\sum_{i \in I} \sigma_i^A \in \mu_p^A(\bigcup_{i \in I} T_i)$ and $\sum_{i \in I} \sigma_i^B \in \mu_p^B(\bigcup_{i \in I} T_i)$. Therefore, $\sum_{i \in I} \sigma_i = \sum_{i \in I} < \sigma_i^A, \sigma_i^B > =< \sum_{i \in I} \sigma_i^A, \sum_{i \in I} \sigma_i^B > \in \mu_p^A(\bigcup_{i \in I} T_i) \times \mu_p^A(\bigcup_{i \in I} T_i) = \mu_p^C(\bigcup_{i \in I} T_i)$. That is, C also has composability.

(9) Let $T_{i \in I}$ be a collection of test sets. Let $\sigma_C \in \mu_p^C(\bigcup_{i \in I} T_i)$. By the definition of the product scheme, there are $\sigma_A \in \mu_p^A(\bigcup_{i \in I} T_i)$ and $\sigma_B \in \mu_p^B(\bigcup_{i \in I} T_i)$ such that $\sigma_C =< \sigma_{A,} \sigma_B >$. The decomposability of A and B implies that there exist $\sigma_i^A \in \mu_p^A(T_i)$ and $\sigma_i^B \in \mu_p^B(T_i)$, $i \in I$, such that $\sigma_A = \sum_{i \in I} \sigma_i^A$ and $\sigma_B = \sum_{i \in I} \sigma_i^B$. Because $\sum_{i \in I} < \sigma_i^A, \sigma_i^B > =< \sum_{i \in I} \sigma_i^A, \sum_{i \in I} \sigma_i^B >$ and $< \sigma_i^A, \sigma_i^B > \in \mu_p^C(T_i)$, C also has decomposability.

Lemma 15. For all schemes A, B and C, A×(B×C) ~ (A×B)×C.

Proof. Straightforward by the definition.

Example 6. (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. [KO91]. Dead mutants of different types are then recorded separately to provide more detailed information. To define the observation scheme for this, let Φ_1, Φ_2, …,

Φ_n be sets of mutation operators. For each Φ_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 $\mathsf{M}_{Typed}= \mathsf{M}_1 \times \mathsf{M}_2 \times \ldots \times \mathsf{M}_n$.

5.4 Statistical Constructions

Let $\mathsf{B}=<B, \mu^B>$ be an observation scheme. N be any given set of numbers. Then, $<N, \leq>$ is a totally ordered set under the less than or equal to relation $\leq$ on numbers. We can define a scheme $\mathsf{A}=<A, \mu^A>$ as follows.

Definition 17. (Statistical Construction)

A scheme $\mathsf{A}=<A, \mu^A>$ is said to be a *statistical* observation scheme based on $\mathsf{B}=<B, \mu^B>$, 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 $p \in P$, $A_p=N$, 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, $\mu_p^A(T) = \{s_p(\sigma) | \sigma \in \mu_p^B(T)\}$.

Example 7. (Statement coverage)

Let $\mathsf{B}=(B, \mu^B)$ 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 σ. 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 8. (Mutation score)

In mutation testing, mutation score is defined by the following equation and used as an adequacy degree of a test set [DLS78, Bud81].

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

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 σ 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 13. 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 18. (Case-Wise Statistical Construction)

A scheme A=<*A*, μ^A> is said to be a *case-wise statistical* observation scheme based on B=<*B*, μ^B>, 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 $p \in P$, $A_p = D_p \rightarrow N$, 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:

$$\sigma_1 \leq_{A,p} \sigma_2 \Leftrightarrow \forall t \in D_p.(\sigma_1(t) = \mathit{undefined} \vee \sigma_1(t) \leq \sigma_2(t)),$$

where ≤ 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 $T = \{n_i t_i | i \in I\}$ where $i \neq j \Rightarrow t_i \neq t_j$, $\sigma_A \in \mu_p^A(T)$ iff (a) $\forall i \in I.\exists \sigma_i \in \mu_p^B(\{n_i t_i\}).(\sigma_A(t_i) = s_p(\sigma_i))$, and (b) $t \notin T \Rightarrow \sigma_A(t) = \mathit{undefined}$.

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 different properties from its base scheme. The following is such an example.

Example 9.

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.

6. Behaviour Observations on Petri Nets

In this section, we apply the theory of behaviour observation to Petri nets - a well-known model of concurrent and distributed systems [Mur89].

6.1 Basic Notion of Petri Nets

In this subsection, we provide an overview of a class of high-level Petri nets called predicate transition nets (PrT nets in the sequel) [GL81]. A formal definition of predicate transition nets can be found in [He96].

A *PrT net* is a tuple (*Nt*, *Spec*, *Ins*), where

(1) *Nt* = (*Pl*, *Tr*, *Fl*) is the net structure, in which
  (i) *Pl* and *Tr* are non-empty finite sets satisfying $Pl \cap Tr = \varnothing$. *Pl* and *Tr* are the sets of *places* and *transitions* of *Nt* respectively;
  (ii) $Fl \subseteq (Pl \times Tr) \cup (Tr \times Pl)$ is a flow relation, called the *arcs* of *Nt*;
(2) *Spec* is the underlying specification, which defines the types, tokens, labels, and constraints of *Nt*;
(3) *Ins* = (φ, *L*, *R*, M_0) is a net inscription that associates a net element in *Nt* with its denotation in *Spec*:
  (i) φ is a mapping that associates each place *pl* in *Pl* with a valid type defined in *Spec*,
  (ii) *L* is a mapping that maps each arc in *Fl* to a valid label defined in *Spec*,
  (iii) *R* is a mapping that associates each transition *tr* in *Tr* with a first order logic formula defined in *Spec*,
  (iv) M_0 is a set of *initial markings*. Each initial marking assigns a multiple set of tokens

defined in *Spec* to each place *pl* in *Pl*.

A *marking* is a distribution of tokens in places. A transition is *enabled* if its pre-set places contain enough tokens and its constraint is satisfied with an *occurrence mode*, which is a substitution of relevant label variables with tokens. An enabled transition can *fire*. The firing of an enabled transition consumes the tokens in the pre-set places and produces tokens in the post-set places. We denote the firing of transition *tr* with occurrence mode α in marking *M* by $M \xrightarrow{tr/\alpha} M'$, where *M'* is the resulting marking. We call the pair tr/α a *firing*. Two transitions, including the same transition with two different occurrence modes, are in *conflict* if the firing of one of them disables the other. Two transitions not in conflict can fire concurrently. Conflicts are resolved non-deterministically. An *execution step* of a PrT net consists of the simultaneous firings of non-conflict enabled transitions, and is denoted by $M \xrightarrow{TrO} M'$, where *TrO* is a set of firings. An *execution* of a PrT net is a sequence of consecutive execution steps starting from an initial marking. The *dynamic semantics* of a Petri net is the set of all possible maximal execution sequences starting from initial markings.

Notice that, firstly, we have used an interleaved-set semantic model here and thus true concurrency can be studied. Alternatively, other semantic models such as branch structure and partial order can be used [Rei85]. Secondly, we have considered the dynamic semantics of a PrT net from a set of initial markings, instead of a single initial marking.

The following PrT net specifies the well-known dining philosophers' problem.

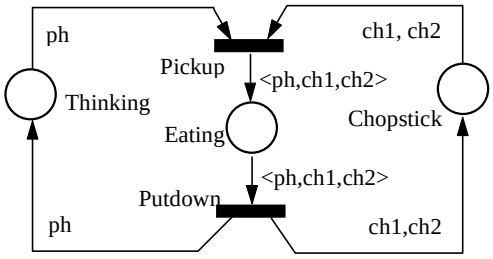

Figure 2 A PrT Net Specification of Dining Philosophers Problem

Figure 2 shows two philosopher states denoted by places *Thinking* and *Eating* respectively, two transitions *Pickup* and *Putdown*, and the available chopstick state defined by place *Chopstick*. The net inscription (φ, L, R, M_0) is as follows.

(1) *Place Types:*
$\varphi(\text{Thinking}) = \varphi(\text{Eating}) = 2^{\text{PHIL}}$, $\varphi(\text{Chopstick}) = 2^{\text{CHOP}}$,
where types PHIL and CHOP are induced from integers and defined in *Spec*.
(2) *Arc Labels:*
$L(\text{Thinking}, \text{Pickup}) = \text{ph}$, and the rest are obvious from Figure 2.
(3) *Transition Constraints:*
$R(\text{Pickup}) = (\text{ph} = \text{ch1}) \wedge (\text{ch2} = \text{ph} \oplus 1)$, $R(\text{Putdown}) = \textbf{true}$,
where $\oplus$ is modulus *k* addition.
(4) *Initial Marking:*
$M_0 = \{m_k \mid k=2, 3, \ldots,\}$, where m_k is defined as follows:
$m_k(\text{Thinking}) = \{1, 2, ..., k\}$, $m_k(\text{Eating}) = \varnothing$, $m_k(\text{Chopstick}) = \{1, 2, ..., k\}$.

The above specification allows concurrent executions such as multiple non-conflicting (non-neighboring) philosophers picking up chopsticks simultaneously, and some philosophers picking up chopsticks while others putting down chopsticks. The constraints associated with transitions *Pickup* and *Putdown* also ensure that a philosopher can only use two designated

chopsticks defined by the implicit adjacent relationships. Table 1 below gives the details of a partial execution of the PrT net in Figure 2.

Table 1. An Execution of the PrT Net in Figure 2

Markings m_i			Firings f_i	
Thinking	Eating	Chopstick	Fired Transition	Occurrence Mode
{1,2,3,4,5}	{ }	{1,2,3,4,5}	Pickup	{ph←1, ch1←1, ch2←2}
{2,3,4,5}	{<1,1,2>}	{3,4,5}	Putdown	{ph←1, ch1←1, ch2←2}
{1,2,3,4,5}	{ }	{1,2,3,4,5}	Pickup	{ph←2, ch1←2, ch2←3}
{1,3,4,5}	{<2,2,3>}	{1,4,5}	Pickup	{ph←4, ch1←4, ch2←5}
{1, 3, 5}	{<2,2,3>, <4,4,5>}	{1}	Putdown	{ph←2, ch1←2, ch2←3}
{1, 2, 3, 5}	{<4, 4, 5>}	{1,2,3}	Putdown	{ph←4, ch1←4, ch2←5}
{1,2,3,4,5}	{ }	{1,2,3,4,5}	Pickup	{ph←5, ch1←5, ch2←1}
{1,2,3,4}	{<5,5,1>}	{2,3,4}	Pickup	{ph←3, ch1←3, ch2←4}
{1,2,4}	{<5,5,1>, <3,3,4>}	{2}	Putdown	{ph←3, ch1←3, ch2←4}
{1,2,3,4}	{<5,5,1>}	{2,3,4}	Putdown	{ph←5, ch1←5, ch2←1}
{1,2,3,4,5}	{ }	{1,2,3,4,5}	…	…

6.2 Behaviour Observation Schemes on Petri Nets

Let p be a PrT net and M_0 be the set of initial markings of p. Thus, M_0 can be viewed as the domain of valid input for p. From the definition of PrT nets, an execution σ of p on a test case m_0 is a maximum sequence of execution steps starting from m_0, and is denoted by

$$\sigma : m_0 \xrightarrow{TrO_0} m_1 \xrightarrow{TrO_1} m_2 \cdots m_k \xrightarrow{TrO_k} \cdots$$

where $m_0 \in M_0$, m_i, i=1,2,…, are markings such that each m_i is obtained from m_{i-1} by firings TrO_{i-1}. For many concurrent systems, a maximum sequence of markings can be infinite, i.e. the execution does not terminate. However, in software testing practice, we cannot observe and record an infinite execution within a finite period of time. Therefore, we stop execution manually and observe and record a partial execution. We use σ^n to denote the partial execution (or prefix) consisting of the first n execution steps of σ, i.e. $\sigma^n : m_0 \xrightarrow{TrO_0} m_1 \xrightarrow{TrO_1} m_2 \cdots m_{n-1} \xrightarrow{TrO_n} m_n$. We denote the set of all finite partial executions from an initial marking m_0 of p as Σ^+_{p,m_0} and Σ^+_p as the set of finite partial executions from the set of initial markings M_0, respectively. We call the firing sequence TrO_0 TrO_1 … TrO_{n-1} extracted from a partial execution σ^n a *trace* and denote it by $Trace_p(\sigma^n)$. We use $Firing_p(\sigma^n)$ to denote the set of firings extracted from the trace $Trace_p(\sigma^n)$. Furthermore, we denote the transition sequence Tr_0 Tr_1 … Tr_{n-1} extracted from a trace by dropping all occurrence bindings as $Trans_p(\sigma^n)$, where each Tr_i is a multiple set.

It is worth noting that partial executions can have different numbers of execution steps, which are determined by the tester, hence form an additional dimension of non-determinism of testing. Without the loss of generality, we call both complete executions and partial executions as *test executions* in the sequel.

In the following sections, we define several concrete behaviour observation schemes based on the transition coverage criteria proposed in testing ER nets (a type of high-level Petri nets) in [MP90]. Six transition coverage criteria proposed in [MP90] are:

(C1) *Firing Sequence Adequacy*: an adequate testing must include all feasible firing sequences from the initial marking set;
(C2) *Firing Adequacy*: an adequate testing must cover all feasible firings;

(C3) *Transition Sequence Adequacy*: an adequate testing must cover all feasible transition sequences;

(C4) *Transition Adequacy*: in an adequate testing, every feasible transition must be fired at least once;

(C5) *N-Times Adequacy*: an *N*-Times adequate testing must cover those feasible transition sequences that contain any one transition no more than *N* times;

(C6) *N-Notable Adequacy*: an *N*-Notable adequate testing must cover those feasible transition sequences that contain any one transition no more than *N* times and no more than one *notable* sub-sequence, where a sub-sequence is said *notable*, if all sequences of the same length containing it result in the same final marking.

In [MP90], interleaving semantics was used such that each execution step involved only one firing of a transition, which is a special case of our execution step that can have simultaneous firings of multiple transitions. The subsuming relationships among the above adequacy criteria are shown in Figure 3.

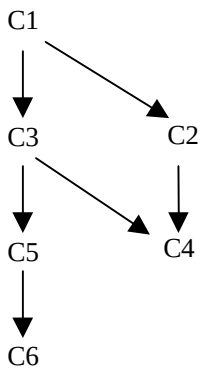

Figure 3. The Subsuming Relationships

Definition 19. (Firing Sequence Scheme)

For all $p \in P$, the universe of observation on p under the firing sequence scheme $\Omega_p^{FS} = < B_p^{FS}, \mu_p^{FS} >$ is defined as follows:

(1) $B_p^{FS} = 2^{Trace_p}$, where $Trace_p$ is the set of all traces from the set of initial markings,

(2) The partial ordering $\leq_p^{FS}$ on B_p^{FS} is the set inclusion;

(3) For all $m \in M_0$, $\mu_p^{FS}(\{m\}) = \{\{\sigma\} \mid \sigma \in Trace_{p,m}\}$, where $Trace_{p,m}$ is the set of all traces from the initial marking m;

(4) For all test sets T, $\mu_p^{FS}(\bigcup_{i \in I} T_i) = \left\{ \bigcup_{i \in I} \sigma_i \mid \sigma_i \in \mu_p^{FS}(T_i) \wedge i \in I \right\}$.

The domain D_p of a PrT net p is the set of initial markings M_0. From the definition of traces, it is easy to see the following properties of μ_p^{FS}.

(i) $\mu_p^{FS}(\varnothing) = \{\varnothing\}$,

(ii) $T \cap M_0 \neq \varnothing \Rightarrow \varnothing \notin \mu_p^{FS}(T)$,

(iii) $\mu_p^{FS}(T) = \mu_p^{FS}(T \cap M_0)$.

Thus the Firing Sequence Scheme is regular.

Definition 20. (Firing Sequence Coverage)

Let E be a collection of test executions of Petri net p. E is said to satisfy the *firing sequence coverage* criterion iff $Trace_p(E) = Trace_p$. The coverage measurement of E is defined by the formula $FSC(p,E) = \|Trace_p(E)\| / \|Trace_p\|$.

It is worth noting that most concurrent systems in practical use contain an infinite number of firing sequences. To satisfy such an adequacy criterion, we may need an infinite amount of computation resource.

Definition 21. (Firing Scheme)

For all $p \in P$, the universe of observation on p under the firing scheme $\Omega_p^{FT} = < B_p^{FT}, \mu_p^{FT} >$ is defined as follows.

(1) $B_p^{FT} = 2^{Firing_p}$, where $Firing_p$ is the set of all feasible firings in p,

(2) The partial ordering $\leq_p^{FT}$ on B_p^{FT} is the set inclusion;

(3) For all test sets T, $\mu_p^{FT}(T) = \left\{ \bigcup_{x \in u} Firing_p(x) \middle| u \in \mu_p^{FS}(T) \right\}$.

It is easy to see that the Firing Scheme is an extraction of the Firing Sequence Scheme such that the orders between firings are ignored.

The following defines the firing coverage criterion.

Definition 22. (Firing Coverage Criterion)

Let β be an observation under the firing scheme on Petri net p during a testing. The testing is said to be adequate according to the *firing coverage criterion* iff $\beta = Firings_p$. Moreover, for all $\beta \in B_p^{FT}$, the adequacy measurement is $FTC(p,\beta) = \|\beta\| / \|Firing_p\|$.

For example, the execution of the Petri net given in Table 1 satisfies the firing coverage criterion. All the transitions (i.e. *Pickup* and *Putdown*) and possible occurrences have appeared.

Definition 23. (Transition Sequence Scheme)

For all $p \in P$, the universe of observation on p under the transition sequence scheme $\Omega_p^{TS} = < B_p^{TS}, \mu_p^{TS} >$ is defined as follows.

(1) $B_p^{TS} = 2^{Trans_p}$, where $Trans_p$ is the set of transition sequences extracted from $Trace_p$ by ignoring occurrence modes in execution steps;

(2) The partial ordering $\leq_p^{TS}$ on B_p^{TS} is the set inclusion;

(3) For all test sets T, $\mu_p^{TS}(T) = \left\{ \bigcup_{x \in u} Trans_p(x) \middle| u \in \mu_p^{FT}(T) \right\}$, where $\mu_p^{FT}(T)$ is the recording function defined in the Firing Sequence Scheme.

It is easy to see that the Transition Sequence Scheme is an extraction of the Firing Sequence Scheme such that the bindings of occurrence modes to transitions are ignored.

Definition 24. (Transition Scheme)

For all $p \in P$, the universe of observation on p under the transition scheme $\Omega_p^{TR} = < B_p^{TR}, \mu_p^{TR} >$ is defined as follows.

(1) $B_p^{TR} = 2^{TR_p}$, where TR_p is the set of all feasible transition;

(2) The partial ordering $\leq_p^{TR}$ on B_p^{TR} is the set inclusion;

(3) For all test sets T, $\mu_p^{TR}(T) = \{t \mid t/\alpha \in \mu_p^{FT}(T)\}$, where $\mu_p^{FT}(T)$ is the recording function defined in the Firing Scheme.

It is easy to see that the Transition Scheme is an extraction of the Firing Scheme such that the bindings of occurrence modes to transitions are ignored. Alternatively, the Transition Scheme can be extracted from the Transition Sequence Scheme.

Definition 25. (N-Times Scheme)

For all $p \in P$, the universe of observation on p under the N-times scheme $\Omega_p^{NT} = < B_p^{NT}, \mu_p^{NT} >$ is defined as follows.

(1) $B_p^{NT} = 2^{NT_p}$, where NT_p is the set of prefixes of transition sequences extracted from $Trans_p$ that no transitions appear more than N times in the sequence;

(2) The partial ordering $\leq_p^{NT}$ on B_p^{NT} is the set inclusion;

(3) For all test sets T, $\mu_p^{NT}(T) = \left\{ \bigcup_{x \in u \cap NT_p} \{x\} \mid u \in \mu_p^{TS}(T) \right\}$, where $\mu_p^{TS}(T)$ is the recording function defined in the Transition Sequence Scheme.

It is easy to see that the N-Times Scheme is an extraction of the Transition Sequence Scheme such that only transition sequences containing no more than N appearances of any transition are included.

Definition 26. (N-Notable Scheme)

For all $p \in P$, the universe of observation on p under the N-Notable scheme $\Omega_p^{NN} = < B_p^{NN}, \mu_p^{NN} >$ is defined as follows.

(1) $B_p^{NN} = 2^{NN_p}$, where NN_p is the set of equivalent classes defined on notable sub-sequences from the set NT_p;

(2) The partial ordering $\leq_p^{NN}$ on B_p^{NN} is the set inclusion;

(3) For all test sets T, $\mu_p^{NN}(T) = \left\{ \bigcup_{x \in u} NN_p(x) \mid u \in \mu_p^{NT}(T) \right\}$, where $\mu_p^{NT}(T)$ is the recording function defined in the N-Times Scheme.

It is easy to see that the N-Notable Scheme is an extraction of the N-Times Scheme such that only one representative is needed for a set of equivalent transition sequences.

The extraction relationships among the above observation schemes are shown in Figure 4.

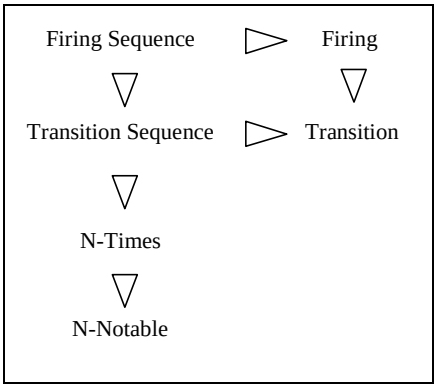

Figure 4. The Extraction Relationships

7. Discussion

7.1 Relation to Work on Theory of Software Testing

The research on software testing theory has focused on test adequacy criteria since Goodenough and Gerhart [GG75] pointed out that the central problem of software testing is test criteria. A great number of test criteria have been proposed and investigated in the literature [ZHM97]. Researches have been conducted to establish the relationships between test criteria and fault detecting ability and software reliability. In recent years, axiomatic approaches have been advanced to understand test criteria from a very high level of abstraction [BuA82, Wey88, PZ91, PZ93, ZH93, ZHM95, Zhu95, Zhu96a]. However, few existing theories of software testing take concurrency and non-determinism into account, instead the uniqueness of dynamic behaviour is assumed. Given the fact that a test set can generate a number of different dynamic behaviours of a non-deterministic system, we argued that test criteria should be defined as functions (or predicates) of the behaviour observed from test executions. By doing so we addressed the complexity of non-uniqueness of observations due to concurrency and non-determinism and extended the notion of test criteria. We also argued that the way to observe dynamic behaviour is a fundamental part of all testing methods. Such behabiour observations method are required to be consistent and systematic, if the testing method is well established. Yet, different observation methods have different properties.

The notion of observation scheme proposed in this paper formally characterises systematic and consistent methods of recording and observing software dynamic behaviour. The desirable properties and constructions of schemes proposed and investigated in this paper are analogue to the axiomatic study of test adequacy criteria proposed by Weyuker in [BuA82, Wey88], formalised by Parrish and Zweben in [PZ91, PZ93], and further developed in [ZH93, Zhu95, Zhu96a]. We are further investigating how to adapt and extend existing axioms of test adequacy criteria to our new notion of test adequacy criteria defined on observable phenomena. We are also searching for axioms that are suitable for testing concurrent systems but have not appeared in the study of testing sequential programs.

Relationships between testing methods have been investigated using the subsumption relation between adequacy criteria, e.g. [Nta88, FW88, CPR89]. Recently, Frankl and Weyuker [FW93a] proved that in general the subsumption relation does not guarantee better fault detecting ability, while Zhu [Zhu96b] proved that in posterior testing scenario the subsumption relation does mean better fault detecting. Other relations between testing methods are also defined as relations between test adequacy criteria, for example, various

relations proposed and investigated by Frankl and Weyuker [FW93b] and the framework proposed by Gourlay [Gou83]. The extraction relation between observation schemes provides an alternative approach for analysing relationships between testing methods. Many questions remain for further research, for example, the relationship between extraction relation and fault detecting ability, and the relationship between extraction and the subsumption relation and various Frankl and Weyuker's relations between test criteria [FW93a, FW93b].

The applicability of the theory proposed in this paper is demonstrated by the analysis of a number of testing methods proposed in [MP90] for testing Petri nets. The observation schemes underlying some well-known testing methods, such as mutation testing, statement and path testing, are discussed in the paper as examples. We are investigating the observation schemes underlying data flow testing methods [LaK83, Nta84, RaW85, FW88] and specification based testing methods [SC96, BGM91, Hie92, RiC85, Kem85]. We are also applying the theory to guide the design of a serious of testing methods for distributed concurrent systems specified in hierarchical predicate transition nets.

7.2 Relation to Work on Semantics of Programming Languages and Domain Theory

In this paper, we argued that the space of phenomena about software dynamic behaviour observable from software testing constitutes a CPO (complete partially ordered sets), if the the observations are systematic and consistent. The mathematical structure of CPO's has been investigated in the context of programming language semantics [Sto77]. As a result, the domain theory is established, which is an important branch of theoretical computer science, and forms the foundation of the denotational semantics of computer languages. The denotational semantics of programming languages considers the semantics of a program as a monotonic and continuous function on CPO. Various power domain constructions have been proposed and studied to define the denotational semantics of concurrent and non-deterministic programs [Plo76, Smy78]. In this paper, we have seen that the great variety of testing methods provides a spectrum of fresh concrete examples and general constructions of CPO's with a novel practical application of domain theory to software testing. In this paper, we focused on the issues related to software testing rather than the mathematical properties of such CPO's. From the viewpoint of domain theory, an observation scheme could be considered as a monotonic and continuous mapping from the space of test sets, which is a complete lattice, to a 'power domain' based on a CPO phenomena. The extendibility and tractability are monotonic conditions for schemes with respect to lower and upper power domain constructions, respectively. The composability and decomposability can then be considered as some kind of continuity conditions for schemes. However, an observation scheme differs from semantics in the way that given a program the semantics associates an output (or behaviour) with one input while a scheme relates an observable phenomenon to a set of test cases (i.e. inputs). Although it is demonstrated in the paper that the definition of observation schemes and various desirable properties of schemes are consistent with the operational semantics of Petri nets, the relationship between semantics of concurrent systems and the notion of scheme would be a very interesting topic for further research.

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.

References

[ZHM97] Zhu, H., Hall, P. and May, J., Software unit test coverage and adequacy, ACM

Computing Survey, Vol. 29, No. 4, Dec. 1997, pp366~427.

[GG75] Goodenough, J.B. & Gerhart, S.L., Toward a theory of test data selection, IEEE TSE, Vol.SE_3, June 1975.

[BuA82] Budd, T. A. & Angluin, D., Two notions of correctness and their relation to testing, Acta Informatica, Vol. 18, 1982, pp31-45.

[Wey86] Weyuker, E. J., Axiomatizing software test data adequacy, IEEE TSE, Vol.SE_12, No.12, December 1986, pp1128-1138.

[ChS87] Cherniavsky, J. C. & Smith, C. H., A recursion theoretic approach to program testing, IEEE TSE, Vol. SE_13,No.7, July 1987, pp777-784.

[DaW88] Davis, M. & Weyuker E., Metric space-based test-data adequacy criteria, The Computer Journal, Vol.13, No.1, February 1988, pp17-24.

[Wey88] Weyuker, E.J., The evaluation of program-based software test data adequacy criteria, Communications of the ACM, Vol.31, No.6, June 1988, pp668-675.

[PZ91] Parrish, A. & Zweben, S.H., Analysis and refinement of software test data adequacy properties, IEEE TSE, Vol. SE_17, No. 6, June 1991, pp565-581.

[PZ93] Parrish, A.S. and Zweben, S.H., 1993, Clarifying Some fundamental Concepts in Software Testing, IEEE TSE, Vol. 19, No.7, July 1993, pp742~746.

[FW93] Frankl, P.G. & Weyuker, J.E., A formal analysis of the fault-detecting ability of testing methods, IEEE TSE, Vol. 19, No. 3, March 1993, pp202- 213.

[ZH93] Zhu, H. & Hall, P., Test data adequacy measurement, SEJ, Vol. 8, No.1, Jan. 1993, pp21~30.

[GS90] Gunter, C. A., Scott, D. S., Semantic domains, In Handbook of Theoretical Computer Science, Vol. B., Formal Models and Semantics, Ed. J. van Leeuwen, The MIT Press/Elsevier, 1990, pp633~674.

[DLS78] DeMillo, R.A., Lipton, R.J. & Sayward, F.G., Hints on test data selection: Help for the practising programmer, Computer, Vol.11, April 1978, pp34-41.

[Bud81] Budd, T. A., Mutation analysis: Ideas, examples, problems and prospects, in Computer Program Testing, Chandrasekaran, B., and Radicchi, S., (eds), North-Holland, 1981. pp129~148.

[How82] Howden, W.E., Weak mutation testing and completeness of test sets, IEEE TSE, Vol.SE-8, No.4, July 1982, pp371-379.

[Mye79] Myers, G. J., The art of software testing, John Wiley and Sons, New York, NY, 1979.

[Bei90] Beizer, B., Software testing techniques, 2nd Edition, New York, Van Nostrand Reinhold, 1990.

[How76] Howden, W.E., Reliability of the path analysis testing strategy, IEEE TSE, Vol.SE_2, September 1976, pp.208-215.

[KO91] King, K.N. & Offutt, A.J., A FORTRAN language system for mutation-based software testing, Software--Practice and experience, Vol.21, No.7, July 1991, pp685-718.

[Mur89] Murata, T., Petri nets, Properties, analysis and applications, *Proc. of IEEE*, vol. 77, no.4, 1989, pp541-580.

[GL81] Genrich, H. J. and Lautenbach, K., System modelling with high level Petri nets, *Theoretical Computer Science*, vol.13, 1981, pp109-136.

[He96] He, X., A Formal Definition of Hierarchical Predicate Transition Nets, Proc. of the 17th International Conference on Application and Theory of Petri Nets (ICATPN'96), Lecture Notes in Computer Science, vol. 1091, Osaka, Japan, 1996, pp212-229.

[Rei85] Reisig, W., On the semantics of Petri Nets, in Formal Models in Programming, Neuhold, E. J. and Chroust, G. eds, North Holland, 1985, pp347~372.

[ZHM95] Zhu, H., Hall, P., and May, J., Understanding software test adequacy -- An axiomatic and measurement approach, Mathematics of Dependable Systems, Edited by Mitchell, C., and Stavridou, V., Oxford University Press, 1995, pp275~295.

[Zhu95] Zhu, H., Axiomatic assessment of control flow based software test adequacy criteria, SEJ, Sept. 1995, pp194~204.

[Zhu96a] Zhu, H., A formal interpretation of software testing as inductive inference, Journal of Software Testing, Verification and Reliability, UK., Vol. 6, July 1996, pp3~31.

[Nta88] Ntafos, S.C., A comparison of some structural testing strategies, IEEE TSE, Vol. SE-14, June 1988, pp868-874.

[FW88] Frankl, P.G. & Weyuker, J.E., An applicable family of data flow testing criteria, IEEE TSE, Vol.SE_14, No.10, October 1988, pp1483-1498.

[CPR89] Clarke, L.A., Podgurski, A., Richardson, D.J., & Zeil, S.J., A formal Evaluation of data flow path selection criteria, IEEE TSE, Vol.15, No.11, November 1989, pp1318-1332.

[FW93a] Frankl, P.G. & Weyuker, J.E., A formal analysis of the fault-detecting ability of testing methods, IEEE TSE, Vol. 19, No. 3, March 1993, pp202- 213.

[Zhu96b] Zhu, H., A formal analysis of the subsume relation between software test adequacy criteria, IEEE Transactions on Software Engineering, Vol. 22, No. 4, April 1996, pp248~255.

[FW93b] Frankl, P. G., Weyuker, E. J., Provable Improvements On Branch Testing, IEEE TSE, Vol.19 No.10, 1993, pp.962-975

[Gou83] Gourlay, J. A mathematical framework for the investigation of testing, IEEE TSE, Vol.SE_9, No.6, November 1983, pp686-709.

[MP90] Morasca, S. and Pezze, M., Using high-level Petri nets for testing concurrent and real-time systems, Real-Time Systems, Theory and Applications (H. Zedan ed.), North Holland, 1990, pp119-131.

[LaK83] Laski, J. and Korel, B, A data flow oriented program testing strategy, IEEE TSE, Vol. SE-9, May 1983, pp33-43.

[Nta84] Ntafos, S.C., On required element testing, IEEE TSE, Vol. SE_10, No. 6, November 1984, pp795-803.

[RaW85] Rapps, S. & Weyuker, E.J., Selecting software test data using data flow information, IEEE TSE, Vol.SE_11, No.4, April 1985, pp367-375.

[FW88] Frankl, P.G. & Weyuker, J.E., An applicable family of data flow testing criteria, IEEE TSE, Vol.SE_14, No.10, October 1988, pp1483-1498.

[SC96] Stock, P. and Carrington, D., A framework for specification-based testing, IEEE TSE,

Vol. 22, No. 11, 1996, pp777~793.

[BGM91] Bernot, G., Gaudel, M.C. & Marre, B., Software testing based on formal specifications: a theory and a tool, SEJ, November 1991, pp387-405.

[Hie92] Hierons, R., Software Testing from Formal Specification, Ph.D. Thesis, Brunel University, UK, 1992.

[RiC85] Richardson, D.J. & Clarke, L.A., Partition analysis: A method combining testing and verification, IEEE TSE, Vol. SE_11, No. 12, December 1985, pp1477-1490.

[Kem85] Kemmerer, R.A., Testing formal specifications to detect design errors, IEEE TSE, Vol.SE_11, No.1, January 1985, pp32-43.

[Sto77] Stoy, J. E., The Scott-Strachey approach to programming language theory, MIT Press, Cambridge, MA, 1977.

[Plo76] Plotkin, G. D., A powerdomain construction, SIAM Journal of Computing, Vol. 5, 1976, pp452~487.

[Smy78] Smyth, M., Power domains, Journal of Comput. System Sci., Vol. 16, 1978, pp23~36.