Towards a “Practical Formal Method” for Test Derivation

R. L. Probert and L. Wei

TSERG, Department of Computer Science, University of Ottawa

150 Louis Pasteur, Ottawa, Canada K1N 9B4

Tel. (613) 562-5800 Ext. 6709 Fax (613) 562-5185 bob@csi.uottawa.ca

Abstract

We present a new semantic approach, called Nondeterministic Ripple Set (NRS), for modelling process behaviors in a distributed system. Then, we describe its application and contributions to the theory and practice of system testing. This approach considers both *environment control (test purpose)* and *process nondeterminism* as complementary factors determining system behaviours, and captures system behaviours in terms of the mutual influences of these two factors. We then illustrate systematic test derivation by applying this approach to the INRES service as a case study. This semantic approach brings a new dimension, *environment control*, into a formal semantics for testing in distributed systems, and shows a means of capturing the practical notion of “test purpose” within an algebraic context. It thus contributes to bridging the gap between theory and practice in testing communications systems.

Keywords

semantics of distributed processes, communications systems, testing, use cases, scenarios.

1 INTRODUCTION

Computer communications systems are notable for their strategic importance to many sectors of society, including government, industry and commerce, the military, and the educational infrastructures. At the same time, interoperability of these systems remains a major concern, even when two communicating systems both claim conformance the same an international standard (specification). To promote interoperability, ISO and ITU (formerly CCITT) have developed and published standardized tests which must be passed by systems which claim conformance to the service or protocol. The process of developing and maintaining these standardized tests generally helps to improve the correctness and completeness of manufacturers' understanding of the international standards, and the quality of the standards themselves.

Product test engineers are aware that products must satisfy customer requirements as well as international standards. Accordingly, product tests are designed in such a way that the purpose of each test is aligned with one or more of these customer requirements. A complete computer communications system consists of the interactions between a distributed service provider

(called the process) and a user (called the environment). In this paper, we sketch the main parts of a formal theory of modelling the behaviours of such systems. We show how this new theory of Nondeterministic Ripple Sets (NRS) captures the interactions between environment (which tries to satisfy requirements by making choices in a controlled way) and the service process (which can make choices which appear externally to be nondeterministic). We then show how to apply this theory to the INRES service for test design. The final sections of the paper describe the general contributions of the approach to conformance testing, and conclude with a brief discussion of benefits of our approach.

2 BACKGROUND

In this paper, we use a standard representation of processes, namely a labelled transition system (Bolognesi, 1989; Brinksma, 1989; Milner, 1988). Labelled transition systems (LTS) are widely used to represent distributed systems. Figure 2.1 depicts (a) a given distributed communicating system, (b) its abstraction into an environment interacting with a nondeterministic process, and (c) an example LTS which represents the interactions between the process and its environment.

Figure 2.1 Three related representations of a distributed communications system.

In Figure 2.1(a), user entities u_1 and u_2 communicate via a link between devices P_1 and P_2 . In Figure 2.1(b), if the PCO (Point of Control and Observation) corresponds to the $\bullet$ between u_1 and P_1 in (a), then u_1 constitutes the environment and the combined interplay of P_1 , P_2 , and u_2 constitute a nondeterministic process (from u_1 's viewpoint). In Figure 2.1(c), the LTS shows that when u_1 offers a message "a", the nondeterministic process (P_1 , P_2 , u_2) can go from state 1 to either state 2 or state 3. Labelled transition systems are a generalization of finite state machines/automata (Hopcroft, 1979), and provide a natural operational model for formal languages such as CCS, CSP, and Lotos (Milner, 1980; Hoare, 1985; Brinksma, 1987). As well, LTS provide a useful starting point for the NRS approach presented here.

In verification or testing of a nondeterministic process (or a distributed system), three practical aspects, among others, should be addressed:

(1) Environment Constraint:

Each time an observer or tester (environment) interacts with a process, he has a *specific purpose* in his mind, usually called a *verification goal* or *test purpose*. From the environment point of view, the purpose of the tester (or test case) is to interact with the process at the PCO and try to *constrain* the process to follow a path which corresponds to the implementation of a customer requirement. Thus, at every state of the process encountered during a test execution, the tester knows which action he should offer according to the test purpose. His test strategy is called *environment constraint*.

(2) Non-deterministic choice of process:

Nondeterminism in the process can interfere with the purpose of the tester. For example, the tester tries to guide the process to follow one specific path determined by his purpose; while the nondeterministic transition at a state may steer the process into another path which fails the test purpose. A practical scenario is that a tester tries to establish a connection between two communicating entities to test a transfer of a data file. But, one process disconnects itself. In this case, he is not successful in achieving his test purpose.

(3) Termination of a test and test management:

When a path satisfying the test purpose cannot be followed because of the process's nondeterminism, test execution can usually not be stopped right away. The process has to be guided further until it reaches a proper stopping point. For example, the operation of a communication system can be continuous, however, a reasonable *session* could be a complete data transfer from connection request to disconnection confirmation, or a complete duration from login to logout. Stopping at a recognizable state is practical for managing test executions.

According to these aspects of testing, we know that the behavior of a process cannot be decided by either the process or its environment alone. Given a *test purpose* of an environment, the environment constraints can be derived in advance from the *test purpose*. Then, in test execution, a *set of traces* has the potential to be followed due to nondeterminism inside the process. This forms a semantic denotation to characterize behaviors of the process (Wei, 1994).

Test cases in practical test suites such as for X.25 (ISO/IEC 8882-3) reflect this observation. For example, each test case in the standard X.25 test suite described in TTCN (Probert, 1992) has two essential parts, (1) test purpose, and (2) dynamic behavioral description. Each dynamic behavioural description is derived based on the test purpose and has a tree structure as follows:

$$\begin{array}{ll}
 !Act\ a & \text{(send "a")} \\
 ?Act\ b & \text{(receive "b")} \\
 !Act\ c & \\
 & \dots \\
 ?Act\ d & \text{(or receive "d")} \\
 !Act\ e &
 \end{array}$$

where $!Act\ a$ is an action (send message a to u_2) performed by entity u_1 in Figure 2.1, and $?Act\ b$ and $?Act\ d$ are potential responses from u_2 to u_1 's action. Because u_1 does not know which response will come from u_2 , u_1 must be prepared to deal with either of them in the test case. Such a behavioral tree can be unwound into a set of traces. The test cases derived from other approaches such as Canonical Tester (Brinksma, 1989; Langerak, 1989; Wezeman, 1989) also have this characteristic. Test generation in their approach is not directly based on customer requirements.

Note that to limit the length of the paper, we intend here to illustrate our approach rather than present a complete mathematical theory of NRS (Wei, 1994). For our purpose, NRS should be understood to be a set of potential traces of execution of the nondeterministic process by the environment exercising environmental constraint to achieve a specific test purpose. We formalize this in the next section.

3 NRS: A NEW BEHAVIORAL VIEW OF PROCESSES

In this section, we formalize the semantic view and nondeterministic ripple sets based on representing processes as labelled transition systems (Bolognesi, 1989; Cleaveland, 1993). We begin with reviewing basic terminology and discussing some operational characteristics in labeled transition systems.

Definition 3.1 A *labeled transition system* (LTS) is a triple $(P, Act, \longrightarrow)$, where

- i) P is a set of states(processes)
- ii) Act is a set of actions
- iii) $\longrightarrow$ is a transition relation in $P \times Act \times P$

Intuitively, $\longrightarrow$ is the set of transitions which may result when an action is executed in a state. For simplicity of presentation, we assume that Act does not include invisible actions such as τ (This is not necessarily a restriction (ISO/IEC JTC1/SC21/P.54)). We write $p \xrightarrow{a} p'$ in place of $(p, a, p') \in \longrightarrow$ for convenience. Furthermore, the relations $\xrightarrow{a}$ are extended to the sequence of actions(trace) $p \xrightarrow{s} p'$, for every s in Act^* , as follows,

- i) $p \xrightarrow{\epsilon} p'$ if p' is p
- ii) $p \xrightarrow{as} p'$ if $p \xrightarrow{a} p''$ for some p'' such that $p'' \xrightarrow{s} p'$.

This means that $p \xrightarrow{s} p''$ if p can evolve to p'' by performing the sequence of actions s . We also use $p \xrightarrow{s}$ to mean that there exists a p' such that $p \xrightarrow{s} p'$. For technical convenience, we shall only consider LTSs with *finite* branching at their states, that is, for each p in P , the set $\{p' \mid \exists a, p \xrightarrow{a} p'\}$ is finite. For a state r in P , we will represent a process with initial state r as

$LTS_r = (P_r, Act, \longrightarrow_r)$ derived from $LTS = (P, Act, \longrightarrow)$,

where P_r is the set of all states reachable from r by some finite sequence of transitions, and $\longrightarrow_r$ is the restriction of $\longrightarrow$ to P_r . Therefore, a state in P can be used to define a process. We will use the graphical notation for labelled transition systems denoted *synchronization tree* or *behavioral tree* (Guillemot, 1989; Milner, 1980; Velthuys, 1992) in examples for simplicity.

Now, we discuss some operational characteristics of testing based on labelled transition systems, that is, *environment constraint and process choice*.

This can be shown with a state transition as in Figure 3.1. There are three groups of transitions at state p_1 labelled with a , b and c , respectively. In process execution, at this state, which group of transitions will be executed depends first on the environment choice (say " c "). We call this the *environment constraint* of the system behavior. However, after c is offered, all the transitions labelled with c have the potential to be executed. Which transition will be chosen depends on the process choice. Thus, both environment and process exert influences on each other's behavior (transition choice).

Figure 3.1 Environment constraint and process choice (environment chooses " c ", process has 3 choices of transitions).

Thus, a path traversed during execution is decided by two factors as a point in a plane is decided by the point's two coordinates. This path cannot be determined *in advance*, as it depends on the dynamic choices of both sides.

We now consider a process behavior from its environment (testing) side. Testing treats the implementation as a black box. However, the tester can work with the process *specification*, and *imagine* its potential execution paths based on its operational semantics (LTS) (this corresponds

to a test plan). Thus, it is not necessary to look inside the process. The tester's strategy (plan) is driven by a test purpose.

Due to nondeterministic choices of the process, the tester has to cooperate with the process by choosing proper actions at different potential states to keep the execution aligned to his test purpose. Thus, test purpose implies a specific, biased set of paths has the potential to be traversed. The set of traces labeling these paths is a test purpose oriented behavioral characterization of the process.

We now formally define the ideas discussed above. First, we formalize the process choice.

Definition 3.2 (*Ripples or Derivatives at a state for process choice*) For an LTS = $(P, Act, r, \longrightarrow)$ and $p, p' \in P, a \in Act$, we define

$D(p, a) = \{p' \mid p \xrightarrow{a} p'\}$, $D(p, a)$ is called the *Ripples*, or *Derivatives* at p with respect to a .

At any state p of an LTS, let $D(p, a) \neq \emptyset$. Then, during a transition with action a at p , the process has the potential to reach any p' in $D(p, a)$ after performing action a . We call this **nondeterministic rippling effect of the process with respect to action a at state p** , or simply **nondeterministic rippling**. *Nondeterministic rippling effect* intuitively denotes the spreading of possible execution paths from a state, and formalizes the effect of **process choice** on system behavior.

Definition 3.3 For an LTS = $(P, Act, r, \longrightarrow)$ and $p, p' \in P, a \in Act, s \in Act^*$, respectively,

i) $D(p, s) = \{p' \mid p \xrightarrow{s} p'\}$, the *s-Ripples*, or *-Derivatives* of p .

ii) $S(p) = \{a \mid p \xrightarrow{a} \cdot\}$, the *Successors* of p .

Now, we define the environment constraint. All choices made by the environment at all potential states encountered during an execution can be represented as a set C which is a subset of the set of pairs: $\{(p, a) \mid p \in P \text{ and } a \in S(p)\}$, i.e., $C \subseteq \{(p, a) \mid p \in P \text{ and } a \in S(p)\}$.

Each (p, a) in C stands for the action a chosen by the environment at state p . We take PS to represent the set of all states which may be potentially encountered during an execution. Then, the environment constraint can be described by the choice pattern as defined below.

Definition 3.4 (*Choice pattern for environment constraint*). C is called a *choice pattern*, if C and PS satisfy the following conditions.

- 1) root condition: $r \in PS$,
- 2) control condition: If $p \in PS$ and $S(p) \neq \emptyset$, then for exactly one $a \in S(p)$, $(p, a) \in C$,
- 3) rippling condition. If $(p, a) \in C$, then for all $q \in D(p, a)$, $q \in PS$.

Note that in definition 3.4, the auxiliary set of states, PS , is defined with the choice pattern C in parallel. In a choice pattern, exactly one action can be selected at each state. We represent the set of all choice patterns of a process P as $CS(P)$. In practice, the choice pattern is derived from the test purpose.

Definition 3.5 (*Nondeterministic ripple set(nrs)*)

A **nondeterministic ripple set** of a process is a non-empty set of maximal traces followed potentially during an execution which is under the constraint of the nondeterministic rippling effect of the process and action selections specified by a choice pattern C at all states encountered during the execution.

Operationally, given a process P , and a choice pattern C , a nondeterministic ripple set of P , denoted nrs , can be calculated inductively as follows, where nrs_n is an auxiliary variable denoting a set of traces:

```

 $nrs \leftarrow \{\varepsilon\}.$ 
repeat  $nrs_n \leftarrow \emptyset.$ 
  for each  $s \in nrs$ , and  $p \in D(r, s) \cap PS$ 
    if there exists a  $(p, a) \in C$ ,
      then  $nrs_n \leftarrow nrs_n \cup \{s.a\}, C \leftarrow C - \{(p, a)\}$ 
    else  $nrs_n \leftarrow nrs_n \cup \{s\}.$ 
   $nrs \leftarrow nrs_n.$ 
until  $C = \emptyset.$ 

```

The name *nondeterministic ripple set* is used because it is the result of nondeterministic rippling effect of the process choices on its traces. For an empty choice pattern C , the corresponding nrs is $\{\varepsilon\}$, while for a process with infinite or recursive behaviors, the computation in this definition may not terminate unless some pre-selected exit point in the process is specified as in real testing activities.

Each nondeterministic ripple set characterizes a specific possible nondeterministic behavior of a process during an execution. All possible nondeterministic ripple sets generated by all different choice patterns during all possible executions characterize all possible nondeterministic rippling behaviors of the process. Thus, we can define NRS to characterize all behaviors of the process.

Definition 3.6 (Characterization set NRS of a process) For a process P , the *characterization set of the behaviors of P* , represented as $NRS(P)$, is the set of all nondeterministic ripple sets of P , i.e.
 $NRS(P) = \{nrs \mid nrs \text{ is a nondeterministic ripple set for some choice pattern of } P\}.$

Thus, the $NRS(P)$ is the set of all nrs sets derived from all choice patterns of P . We also call $NRS(P)$ the *characterization* of P , playing the same role as failure sets or acceptance trees (Brookes; Hennessy, 1985; Hennessy, 1988). Examples of characterization given in Figure 3.2.

The sets of choice patterns for these processes are $CS(P1) = \{(r_1, a), (1, b)\}, \{(r_1, a), (1, c)\}, \{(r_1, a), (1, b)\}$, $CS(P2) = \{(r_2, a), (1, b), (2, d)\}, \{(r_2, a), (1, c), (2, d)\}$, and $CS(P3) = \{(r_3, a), (1, b), (2, c), (3, d)\}$, respectively. Their corresponding characterization sets are shown in (b). Intuitively, that any of the three nrs 's in $NRS(P1)$ contains only one trace implies that a tester can control $P1$ to follow any of its three traces ab , ac , and ad in an execution depending on his purpose. Note that $P1$ is deterministic. For process $P2$, no matter which trace the tester wants to follow, there is another trace which has the potential to be followed due to the nondeterministic rippling at the root node. This is why each nrs in $NRS(P2)$ contains two traces. $P3$ is fully nondeterministic, no matter which trace in it is intended to be followed, the other two traces has the potential to be traversed. Thus, $P3$ has only one nrs including all its three traces ab , ac , and ad .

Figure 3.2 Characterization sets of processes.

Finally in this section, we prove an alternative representation for $NRS(P)$ to help apply this theory to test generation and test specification. We transform the trace set representation of $NRS(P)$ as in definition 3.6, into a set of deterministic tree representation. To do this, we first attach a distinguished symbol σ ($\notin Act$) to the end of each finite trace in $nrs \in NRS(P)$. This is to prevent a short trace from being covered by a longer one in the transformation if the short one is a prefix of the longer one. Next, given a set of traces, it is straightforward to arrange all the traces in the set to share all their common prefixes among them to form a deterministic tree. Formally,

Let $nr\sigma = \{s\sigma \mid s \in nrs\}$, and

$NR\sigma(P) = \{nr\sigma \mid nr\sigma \text{ is computed for each } nrs \in NRS(P)\}$.

Let nrt represent the deterministic tree obtained from a $nr\sigma$ by arranging all the traces in the $nr\sigma$ to share all their longest common prefixes, and

$NRT(P) = \{nrt \mid nrt \text{ is the deterministic tree obtained from each } nr\sigma \in NR\sigma(P)\}$.

We have the following proposition.

Proposition 3.1: Given a process P and its $NRS(P)$, a one to one correspondence exists between the elements of $NRS(P)$, $NR\sigma(P)$ and $NRT(P)$.

Proof: The following is a one to one correspondence between the elements of $NRS(P)$, $NR\sigma(P)$ and $NRT(P)$,

$$nrs \Leftrightarrow nr\sigma \Leftrightarrow nrt$$

where $nrs \in NRS(P)$, $nr\sigma \in NR\sigma(P)$, and $nrt \in NRT(P)$; $\Leftrightarrow$ means "corresponds to". The correspondence relations are

$$nrs \Leftrightarrow nr\sigma \text{ iff } nrs = \{s \mid s\sigma \in nr\sigma\}, \text{ and } nr\sigma = \{s\sigma \mid s \in nrs\}$$

$$nr\sigma \Leftrightarrow nrt \text{ iff } nr\sigma = \{s\sigma \mid s\sigma \text{ is a trace of } nrt\}, \text{ and } nrt \text{ is the deterministic tree obtained from } nr\sigma$$

♦

Theorem 3.1 A process P can be characterized by a set of deterministic trees, $NRT(P)$. Each tree in the $NRT(P)$ describes a behavior of the P resulting from the mutual influences between P 's nondeterministic rippling and the environment/test purpose control.

Proof: We prove by construction.

1. Compute the characterization set $NRS(P)$ of P by Definition 3.6.

2. Transform the $NRS(P)$ into $NRT(P)$ according to Proposition 3.1. ♦

The deterministic tree representations $NRT(P)$'s of the $NRS(P)$'s for processes in Figure 3.2(a) are shown in Figure 3.2(c). Deterministic trees are popular specification method for test cases. They have been practically used in the test suite specification written in TTCN (Probert, 1992) and the tests derived from Canonical tester (Brinksmma, 1989), for example. As similar testing mechanisms defined in (Brinksmma, 1989) or (Hennessy, 1988), each tree in a $NRT(P)$ can be seen as a test case for the process concerned, if we add some related verdicts such as *pass* or *fail* to each end of the trace in the tree and synchronize the tree with the process. From this point of view, the characterization set of a process can actually serve a test suite for test selection, and the procedure for generating it be a test generation process. This show that this semantic approach is motivated from testing, and applying it back to the practical testing environment seems promising. In next part of this paper, we show an application of this theory for test generation and test specification.

4 A CASE STUDY OF APPLICATION OF NRS TO TEST GENERATION FOR INRES SERVICE

To illustrate the practicability of this semantic approach, in this section, we apply it to a small but practical data transfer service protocol INRES for test generation. INRES is a simplified version of the Abracadabra service introduced in (Hogrefe; Velthuys, 1992) and has been selected by the international standards organizations ISO/IEC JTC1/SC21/P.54 and CCITT SG X/Q.10 as a reference protocol for studying testing techniques in the context of formal methods(FMCT) (ISO/IEC JTC1/SC21/P.54). The detailed structure and Lotos specification of the service are included in the Appendix of this paper. We shall focus on the testing issues related to this application. We first review some testing terminology and then, turn to the discussion for test generation for INRES by applying the NRS theory of the last section. Some basic OSI (Open Systems Interconnection) concepts are used in this application and we refer the reader to the ISO document: conformance testing framework and methodology (ISO/IS9646) for more details.

The INRES (Initiator-RESPonder) data transfer service is a connection oriented service. It consists of a Service Provider, an Initiator and a Responder. The service can be accessed from two SAPs(Service Access Point). On the one SAP(ini), the initiator must initiate a connection request before sending data. On the other SAP(res), the responder can accept the connection or reject it. After a connection is established, the responder can receive data from the initiator until a disconnection request is sent by the responder.

Figure 4.1 Test configuration for INRES Service.

Suppose that we use the **remote testing method** (ISO/IS9646) to test this service. Then, the testing configuration looks like the one shown in Figure 4.1. This testing method assumes that there is no test interface at the top of the IUT (Implementation Under Test) and no formal Test Coordination Procedures (TCP) between LT (Lower tester) and UT (Upper Tester, not appear in Figure 4.1) exist (only manual coordination, if any). The method relies solely on the protocol to be tested for synchronization between LT and IUT. The state of the IUT is assumed to be known from actions specified for the LT. Verdicts are formulated based on stimulus provided by the LT and responses of the IUT observed by the LT. The remote testing method is widely used for testing implementations of X.25 protocol, where the IUT is the Data Terminal Equipment (DTE) and the LT emulates the Data Communications Equipment (DCE). In our case, the IUT is the entire service provider.

Now, we transform the remote testing configuration and the INRES service specification into the structures which satisfy the conditions for applying our NRS approach.

According to the remote testing method, the whole system behaviors have to be controlled and observed at the SAP: ini. Thus, it is sufficient to generate test cases from the specified behaviors at this point. We can transform the configuration in Figure 4.1 into the similar interactive structure as shown in Figure 4.2 which is modelled by our NRS approach where Tester represents the environment, and both INRES Service Provider and the Responder user (called system under test) represents the process which interacts with the tester.

Figure 4.2 The process to be tested and its environment.

Next, we need the system behavioral specification at the point: ini, i.e., between the process and its environment. This is obtained by projecting the whole system specification into the SAP: ini, and is shown as the behavioral tree in Figure 4.3 which consists of all interactive actions controllable and observable at ini and time orders between them. The black and grey states represent the states in which the system has the same behaviors.

Figure 4.3 Behavioral tree at the SAP: ini.

Furthermore and importantly, we need to determine the nondeterministic rippling effect in the behavioral tree of Figure 4.3 because what is described in the behavioral tree still reflect the system overall behaviors at the interaction point, and the nondeterministic factors in the process to its environment is not specified. This usually depends on the specific configuration (i.e. testing method used) and many practical conditions which may not be described in a formal theory. Note that this is also the normal case when a formal theory is applied to practical situations, many practical concerns must be handled, and often, the detailed realization can only be informally dealt with. For example, in this case, the behavioral tree derived is not like the abstract ones where the initiators of actions are not distinguished. Here, for an action, we now understand that it is the combination of both its initiator and its action symbol. As such, !.a and ?.b are examples of actions.

We decide the nondeterminism in the process according to the usual sense in the context of interactions between two communicating entities: two branches emitting at a state are nondeterministic if the actions (prefixed with ?) labelling these branches are the potential answers to the action (prefixed with !) labelling the branch leading to this state. A similar method is used in (Phalippou). According to this definition and the conditions in our testing configuration, three cases in a behavioral tree should be considered as shown in Figure 4.4.

Figure 4.4 Nondeterministic factors in a behavioral tree.

In Case 1 of Figure 4.4, the two branches labeled with ?b and ?c are nondeterministic because ?b and ?c are responses from the responder entity to action !a. Note that the action ?a is possible before ?b and ?c and it results in two consecutive received actions. This situation (also two consecutive sending actions prefixed with ! such as in case 2) is called *unsynchronized interaction* in protocol design and is undesirable (Sarıkaya, 1984). The two branches in Case 2 which are labelled by the two actions initiated from the initiator are deterministic because which one of them can be sent is under the control of the initiator (i.e., environment). Finally, Case 3 describes a situation where after receiving action a, the initiator wants to send action b, but unexpectedly, action c may arrive. In this situation, whether to accept c or send b first is under

the control of the initiator, if the medium uses two channels for output and input messages, which is our assumption in this configuration. The two branches labeled by !b and ?c can be deterministic. However, if action c contains a critical message such as disconnection indication as in the behavioral tree of Figure 4.3. Then, whenever in such a state, the initiator(environment) has to check for its arrival and accept it first before doing anything else. Here, the action ?c should not happen is a pre-condition of !b. This argument concerns selection of a *choice pattern* in a test case derivation.

For the behavioral tree in Figure 4.3, by the method discussed above, we can decide that the relation between branches at state 2 labeled with ?DISind and ?CONconf are nondeterministic, and the relation between branches at state 4, 5, and 8 labeled with !DATreq and ?DISind is deterministic. *Note that this behavioral tree has infinite behaviours, that is, action ?DATreq can repeatedly appear, and after action !Disind, the tree would repeat the same behaviour as at state r. In this case, as discussed in introduction section, some proper exit point should be made in test generation.* We can decide that the nondeterministic rippling at each state is,

$$\begin{aligned}
 (*) \quad & R(r, !CONreq) = \{2\}, & R(2, ?CONconf) = \{3, 4\}, & R(2, ?DISind) = \{3, 4\}, \\
 & R(4, !DATreq^*) = \{5\}, & R(4, ?DISind) = \{6\}, & R(5, !DATreq^*) = \{7\}, \\
 & R(5, ?DISind) = \{8\}, & R(7, ?DISind) = \{10\}, & \dots\dots
 \end{aligned}$$

Note that we replace action !DATreq with action !DATreq* which is similar to !DATreq, but with an additional check for action ?DISind's arrival before doing !DATreq. Formally, DATreq* is understood as an atomic action: "if not (?DISind) then !DATreq".

Now, we are in a position to generate the tests for the INRES service under the remote testing configuration. We first determine test purposes. As usual, test purposes include:

- 1) check for connection establishment.
- 2) check for data transfer of one data unit.
- 3) check for connection release

Next, choice patterns determined by these test purposes are, (*note that an exit point is made at the completion of each test purpose*)

$$\begin{aligned}
 c1 &= \{(r, !CONreq), (2, ?CONconf), (4, ?DISind)\}, \\
 c2 &= \{(r, !CONreq), (2, ?CONconf), (4, !DATreq^*)\} \\
 c3 &= \{(r, !CONreq^*), (2, ?CONconf), (4, !DATreq^*), (5, ?DISind)\}
 \end{aligned}$$

Figure 4.5 Three finite elements in NRT(ini) determined by choice patterns c1 - c3.

Now, we can generate the finite elements in NRT(ini) based on the choice patterns c1 - c3. *Note that nondeterministic rippling in process ini is based on ones in (*) (not the ones as in the ideal case that branches are labelled by identical actions).* The elements in Figure 4.5 are finite deterministic tree representations of nondeterministic ripple sets for process *ini*. If used as test cases, they can be readily specified by formal description techniques such as Lotos. The deterministic tree representations of NRS can *also* be easily specified in practical situations by TTCN, the international standard test specification notation (Probert, 1992). In TTCN-like

notation, the three elements in Figure 4.5 can be described as test cases with verdicts added as in tables 1 - 3:

Test case 1. Test purpose: check for connection establishment.		
Dynamic behavior tree:	!CONreq.	
	?CONconf	Pass
	?DISind	Inconclusive
	?DISind	

Table 1 Test case 1 for test purpose 1

Here, for assigning a verdict to a test, we only consider the path which completes the function of the test purpose as Pass (success), if it is executed. For other paths, any verdicts such as Pass, Fail or Inconclusive may be assigned depending on the practical considerations.

Test case 2. Test purpose: check for data transfer of one data unit.		
Dynamic behavior tree	!CONreq.	
	?DISind	Inconclusive
	?CONconf	
	!DATreq	Pass

Table 2 Test case 2 for test purpose 2

Test case 3: Test purpose: check for connection release		
Dynamic behavior tree	!CONreq.	
	?DISind	Inconclusive
	?CONconf	
	!DATreq*	
	?DISind	Pass

Table 3 Test case 3 for test purpose 3

These three tests are the main representatives in the test suite for INRES service. Other tests actually consist of one of these three tests with some repetitions of its subpart such as sending more data units or disconnection after sending more data units.

In the above discussion, we have outlined a method of applying our test-purpose directed semantics to test generation in conformance testing for a formal Lotos specification of INRES service protocol. We may remark that the method is also applicable to FSM-based formal specification techniques such as Estelle and SDL. We can also combine the D-method under the same framework of this test generation methodology (Chan, 1989; Sidhu, 1989). D-method is a commonly-used test generation method for protocols specified by FSM-based specification techniques. This can be illustrated by using the same example above, if we directly start with the behavioral tree of the service shown in Figure 4.3. By combining the states of the same behaviors together in Figure 4.3, a finite state machine can be formed as the FSM-based behavioural specification of the INRES service at SAP *ini* (similar Estelle and SDL specifications of INRES service can be found in (Hogrefe)). Then, the same procedure as described above can be followed for test generation. If the *distinguishing sequence* for each state is available, it can be used to verify the state following the transition to be tested as practised in D-method (Sarikaya, 1984). Accordingly, the dynamic behavioral tree for test purpose 1 is as in Table 4:

Preamble	!CONreq.	
Test-body	?DISind ?CONconf	(Pass)
Post-amble	+Check(state r) +Reset	Pass

Table 4 The test case with preamble and postamble for test purpose 1.

The test above has a similar format to a real test such as in the test suite for X.25 protocol (ISO/IEC 8882-3; Probert, 1992).

In summary, this section outlined a methodology for test generation in conformance testing by applying our NRS semantic approach to the INRES service. The result obtained is close to the ones obtained by informal and manual methods, and shows support for the use of formal semantics as a disciplined means of test generation and test specification in practical situations.

5 CONTRIBUTIONS OF NRS APPROACH TO INTEGRATING THEORY AND PRACTICE OF TESTING

As mentioned in the introduction section, This NRS semantic approach comes from our experience in observing process behaviours in testing telecommunications systems. The approach considers both contributing factors to a process behaviour, nondeterminism inside a process and the environment control determined by a test purpose, and thinks that a process behaviour is the result of the mutual influences of these two factors. Different from other semantic modelling methods, by considering the role of the environment control in a process behaviour, the NRS semantic approach brings a new dimension into the formal modelling techniques of process behaviours in a distributed system. This makes possible to handle the practical notion of "*test purpose*" within a formal algebraic context. Furthermore, by describing a process behaviour to be the result of the mutual influences of the two factors, the NRS semantic view appears to be natural for modelling interactive behaviours of processes in testing. These considerations may give rise to some *significant* effect in applying a formal theory to conformance testing of telecommunication systems.

In the following, we discuss a major contribution of the NRS approach to the field of conformance testing of telecommunications systems. We first briefly discuss the current state of conformance testing, and point out a gap between theoretical studies and practical applications in this field. Then, we discuss the special contributions of NRS semantics to bridging this gap.

•A gap between theory and practice of conformance testing

In recent years, theory and practice on conformance testing have been actively carried out in both academic and industrial worlds. In theoretical aspect, the representative testing models: bisimulation testing (Abramsky, 1988; Milner, 1980), failure/testing equivalence (Brookes; Hennessy, 1988; De Nicola, TCS) have been developed. Many other theories also exist but they mainly follow the formats of these models. The main advantage of these models is that they are precise, mathematically-based theories and possess well-defined theorems for discussing process properties. But, in considering applications of these theories to practical testing, some unfavorable features may be observed as.

1. Tests in these models are terms of some algebra. The set of tests appears identical for every process without discrimination, and thus, even for trivial processes, the tests considered can be infinite. This seems less practical.

2. There are no direct relations between tests and requirements-directed test purposes. This point is extremely relevant to practical testing. Thus, the present formal testing models still need to be improved to be more practical for their applications in realistic situations.

In the practical world of testing, the situation is different. At present, aspects 1. and 2. listed above are practised in realistic testing as follows:

- (i) Tests are derived from the specific process(specification) whose implementation is under test.
- (ii) Each test is derived directly from a test purpose.

In general, the practical testing techniques are ad hoc and informal. Test generation and selection are mainly manual which is tedious and error-prone. Furthermore, most practical techniques deal with only deterministic processes.

Most systems are specified in natural languages or informal notations. However, formal specification and testing techniques are gradually becoming of interest the industry world. Current testing techniques are not adequate for test generation and verification. Therefore, applying formal methods to testing is important.

•NRS contributions to bridging the gap in theory and practice of testing

The above discussion points out a gap between theoretical studies and practical applications in conformance testing, that is, the existing testing theory is formal but less practical, and practical testing techniques are practical but informal. Two *crucial* points of testing are handled differently in the two sides. For a formal testing theory to be practically applicable, its processing of these two points has to be matched with realistic testing techniques, that is, the gap between theory and practice of conformance testing has to be bridged.

The NRS approach is considered having made some contributions to bridging this gap. How the theoretical and practical differences of conformance testing are harmonized in the NRS semantics is indicated as follows:

(1-i) NRS approach contributes to harmonizing the differences in test generation between theory and practice. That is, the testing framework derived from NRS approach generates and selects tests from a specification if an implementation of the specification is under test. Furthermore, the three important aspects in testing as mentioned in the introduction section have been precisely modelled in the approach. This renders the number of tests generated from this approach to be in the comparable level required in realistic situation for test selection. This is illustrated as in Figure 3.2 and the case study in section 4.

(2-ii) NRS semantics contributes to solving the problem that a test in a testing theory should be related to a test purpose. In NRS semantics, a nondeterministic ripple set is derived from a choice pattern which is directly determined by a test purpose. This allows that in a formal context, each semantic object has a corresponding test purpose attached. As a whole, the NRS(P) for a process P characterizes all test purposes for testing the process P in the sense that each test purpose is to test a transition or a path of the process.

In addition, NRS approach shows a direct application of a semantic theory to testing practice. The tests derived in NRS semantics can be directly mapped to TTCN (Probert, 1992) or specified by Lotos (Brinksma, 1987; Brinksma, 1989), since each deterministic tree representation of the elements in NRS set can be actually seen as a dynamic behavioral tree of the implementation under test. This is illustrated by the example in section 3.

6 CONCLUSION

We have defined a new behavioral view of processes based on mutual influences between nondeterminism inside a process and the environment control determined by a test purpose. Studies of this semantic view is in the similar range to other well-known semantic views such as bisimulation (Milner, 1980) and failure/testing semantics (Brookes; De Nicola, TCS; Olderog, 1983). Compared with other semantic views, the special contribution of our semantic view (and also the main difference from others) is its consideration of both contributing factors to a process behavior. Other semantic views seem focusing more on the nondeterministic side of a process.

By considering the role of the environment control in a process behaviour, this method introduces a new dimension into the semantic modelling techniques. As such, it appears that our semantic approach is closer to a practical model of process behaviors in a testing situation. This is why some attributes of our NRS approach do not appear in other semantic theories.

As discussed in sections 3 and 4, application of this semantic method to practical testing is promising: (1) the number of semantic items (i.e., deterministic trees in NRT(P) produced from each process P is similar to those generated in practice; (2) each deterministic tree is already in a format similar to the graphical representation of Lotos specifications or the standard test specification in TTCN, and supports direct transformation into a real test case described by these formal description techniques; (3) each tree has a test purpose attached. These are our observations based on our experience, and proof of practicality can only be done in practical applications of this approach to realistic examples of telecommunication system specification and testing.

More theoretical studies of this semantic approach have been done. As usual, the semantic object NRS(P) can induce an equivalence between processes. This equivalence has been shown to imply failure equivalence. In addition, an effective algorithm has been derived to generate *nrs* sets by calculus from CCS-like process notations without enumerating all choice patterns. Furthermore, this semantic view also helps performing trace analysis, and formalizing process relations in system design. Interested readers are referred to (Probert, 1995; Wei) for studies of these aspects.

Finally, we hope that this approach supports and improves on application of formal methods to specification and testing of telecommunication/distributed systems by addressing the practical issue of test purpose. However, compared with other well-studied semantics, much remains to be studied for this semantic approach. Presently, we are extending its theoretical aspects and formulating a testing theory in a realistic environment.

ACKNOWLEDGEMENTS

The authors are very grateful for useful discussions with Professor. L. Logrippo and other colleagues of the University of Ottawa. As well, the authors are grateful to Professor M. Hennessy for many comments and suggestions. This research was supported by a grant from the Telecommunications Research Institute of Ontario.

REFERENCES

- Abramsky S. (1988) Observational equivalence as a testing equivalence, *Theoretical. Computer Science* 53, 225-241.
- Bolognesi T, Caneve M. (1989) Equivalence verification: theory, algorithms and a tool, *The Formal Description Techniques Lotos*, Elsevier Science Publishers B. V.
- Brinksma, H. (1987) An introduction to Lotos, in *Protocol Specification, Testing, and Verification VII*, North-Holland.
- Brinksma H. (1989) A theory for derivation of tests, *The Formal Description Techniques Lotos*, Elsevier Science Publishers B. V.
- Brookes S., Hoare C. and Roscoe A. A theory of communicating sequential processes, *JACM*, 31, No. 7, 560-599.
- ISO/IS 9646, *Conformance Testing Methodology and Framework*, Part 1-Part 6.
- Chan W.M., Vuong S.T., Ito M.R. (1989) The UIOV-method for protocol test sequence generation, *Proc. 2nd Intl. Workshop on Protocol Test Systems*, Berlin, Germany.
- Cleaveland R. and Hennessy M. (1993) Testing equivalence as a bisimulation equivalence, *Formal Aspects of Computing* 5, 1-20.
- ISO/IEC JTC1/SC21/P.54 and CCITT SG X/Q.10 (1993) *Formal methods in conformance testing*, (working Draft for review and comments).
- Luo, G. Bochmann, G., Das, A. and Wu, C. (1992) Failure-equivalent transformation of

- transition systems to avoid internal actions, *Information processing letters* 44, 333-343.
- Guillemot, R. and Logrippo, L. (1989) Derivation of useful execution trees from Lotos specifications by using an interpreter, *FORTE'89*, 311-320.
- Hennessy M. (1985) Acceptance trees, *JACM* 32, 896-928.
- Hennessy M. (1988) *Algebraic theory of processes*, The MIT Press.
- Hoare C.A.R. (1985) *Communicating Sequential Processes*, Prentice-Hall.
- Hogrefe, D. *OSI formal specification case study: the Inres protocol and service, revised*, Institut fur Informatik, Universitat Bern.
- Hopcroft J.E., Ullman J.D. (1979) *Introduction to automata theory, languages and computation*, Addison-Wesley.
- ISO/IEC 8882-3 X.25 DTE Conformance testing-Part 3 Packet Level conformance test suite.
- Langerak R. (1989) A testing theory for Lotos using deadlock detection, 9th IFIP WG 6.1 Intel Symposium on *Protocol Specification, Testing, and Verification*, The Netherlands.
- Milner R. (1980) *A Calculus of Communicating System*, LNCS, Vol 92, Springer Verlag.
- Milner R. (1988) *Operational and algebraic semantics of concurrent processes*, ECS-LFCS-88-46, Department of Computer Science University of Edinburgh.
- De Nicola R. and Hennessy M. *Testing equivalences for processes*, TCS.
- Olderog E. and Hoare C. (1983) Specification-oriented semantics for communicating processes, *Acta Informatica*, Vol.23, 9-66, 1986.34, 83-133.
- Phalippou M. Test suite generation for the Inres protocol with TVEDA tool, *CNET/LAA/SLC/EVP*, BP. 40 F-22301 Lannion CEDEX France.
- Probert, R., Monkewich O. (1992) TTCN - The international notation for specifying tests of communication systems, *Computer Networks and ISDN Systems*, Vol. 23, No. 5, 417-438.
- Probert R., and Wei L. (1995) Testing Behaviours of Processes Based On Nondeterministic Rippling and Environment Control, The 7th International Conference on computing and information, ICCI'95, Trent University Peterborough, Ontario, Canada.
- Sarikaya B. and Bochmann G. (1984) Synchronization and specification issues in protocol testing, *IEEE Transactions on Comm.unications*, Vol. 32, No. 4.
- Sidhu, D. and Leung T. K. (1989) Formal methods for protocol testing: a detailed study, *IEEE Transactions of Software Engineering*, Vol 15, 413-426.
- Velthuys R. J., Schneider J.M. and Zornitlein G. (1992) A test derivation method based on exploiting structure information, Proc. 12th IFIP *Protocol specification, testing, and verification*.
- Wezeman C.D. (1989) The CO-OP method for Compositional derivation of conformance testers, Proc. 9th IFIP on *Protocol Specification, Testing, and Verification*.
- Wei L., *Semantics of specification and testing of distributed systems*, Ph.D. Thesis, Department of Computer Science University of Ottawa, Canada, December, 1994.

BIOGRAPHY

Robert L. Probert is Professor of Computer Science and Coordinator of the Telecommunications Software Engineering Research Group at the University of Ottawa. He was the founding Chair of the ACM Principles of Distributed Computing Conformance, co-chair of the 10th IFIP Symposium on Protocols Specification and Testing, and Scientific Leader for the team that developed the TTCN Workbench, a comprehensive environment for designing test suites for interactive systems. Probert has worked with industry in the area of Software Quality Engineering. His current interests include grey-box analysis, a use-case driven technique for enhancing the quality of software designs.

Linsheng Wei received his doctorate (Computer Science) from the University of Ottawa in the Spring 1995. He is currently employed at Bell-Northern Research Ltd. in Ottawa.