On testing and diagnosis of communication protocols based on the FSM model

T Ramalingam*, Anindya Das† and K Thulasiraman*

A formal analysis of the fault diagnosis capabilities of various FSM-based test sequence selection methods has been carried out[1]. It is shown[1] that the Wp-method[2] has the best fault-resolution capability among all the known test sequence selection methods based on the FSM model when the implementation has at most one faulty transition. In this paper, we present a test sequence selection method with a fault resolution capability better than that of the Wp-method when the implementation has at most one faulty transition, and when the specification meets certain conditions. This method is based on the Wp-method. Approaches for minimizing the test sequence further, and for exactly locating the fault or improving the fault resolution capability of our method, are also presented.

Keywords: fault diagnosis, test sequence selection, FSM model

The goal of interworking among heterogeneous systems can be achieved through conformance testing. Conformance testing of communication protocols has been an active area of research since the emergence of the protocol engineering area[3]. Conformance testing (CT) is intended to ensure that a given implementation of a protocol is equivalent to the standard specification of the protocol[4,5]. The *OSI conformance testing methodology and framework*[5] defines a test suite as a set of test cases, one for each test purpose. A test case is a set of event sequences. A verdict of *pass*, *fail* or *inconclusive* is also assigned to each sequence in the test case. A verdict for an event sequence depends upon the specification of the protocol and the test purpose. If a protocol is specified as a deterministic finite state machine (FSM), then its test suite is usually specified as a single sequence of labels (input and expected output pairs) of the transitions in the FSM. Such a test suite is referred to as a *test sequence*[6]. In such a case, we assume that an implementation passes the conformance testing if it exhibits the output part of the test sequence while the input part of the test sequence is applied. Otherwise the implementation is said to fail. The specification of a protocol is normally described in formal description techniques (FDT) such as LOTOS[7], Estelle[8] or SDL[9]. Such a specification has many advantages, including automatic test suite selection[10]. In this paper, we consider only the control flow aspect of testing. For deriving the control flow behaviour of a protocol, the control structure of the specification can be transformed into an equivalent finite state machine[11]. Henceforth, we refer to the FSM representation of the control flow aspects of a specification and an implementation as SPEC and IUT, respectively. A number of methods for selecting test sequences from SPEC which adequately test IUTs are available in the literature[2,6,12–15].

We have shown[1] that the Wp-method has the best fault diagnosis capability among all the FSM-based test sequence selection methods when the IUT has at most one faulty transition. However, as we pointed out[1], some of the desirable properties (Claim 1 and Claim 2 in Fujiwara *et al.*[2]) for fault diagnosis do not always hold for the Wp-method. In this paper, we develop a test sequence selection method, called the *Detection and Diagnosis* (DD) method, which provides a better fault diagnosis capability than the Wp-method. Our DD-method is based on the Wp-method, and it guarantees the above properties. Also, by incorporating the rural postperson optimization technique[6], the DD-method minimizes the length of the test sequence. Our method requires a set of UIO-sequences[6] satisfying the *Tree UIO Label Disjoint* (TULD) property (defined later). It should be noted that there may exist some protocols which may not have the UIO-sequences with the required property. All the real-life protocols we have analysed so far do have such sequences. We have

*Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada
†DIRO, University of Montreal, Montreal, Canada
Paper received: 2 August 1993; revised paper received: 16 February 1994

0140-3664/95/$09.50 © 1995–Elsevier Science B.V. All rights reserved

assumed that the number of states in the IUT is no more than the number of states in the SPEC.

The paper is organized as follows. Basic definitions and a brief description of the Wp-method are first given. Our DD-method is presented, and approaches for minimizing the test sequence further and for exactly locating the fault or improving the fault resolution capability of our method are given. Finally, the paper concludes with discussions and directions for further research.

PRELIMINARIES

Finite state machine

An FSM M can be formally defined as a 5-tuple $M = (S, s_1, I, O, T)$, where S is the nonempty set of states of M, s_1 is a designated state called the *initial state*, and I and O are nonempty sets of possible inputs and outputs of the protocol, respectively. The transition function T is a partial function defined as $T: S \times I \rightarrow S \times O$. $T(s_i, a) = (s_j, o)$ means that FSM M at state s_i makes a transition to state s_j when the input a is applied, producing the output o. Graphically, this is also represented as $s_i - a/o \rightarrow s_j$.

An FSM $M = (S, s_1, I, O, T)$ can also be represented by a directed labelled graph $G(V, E)$, where $S = V$ and each transition $s_i - a/o \rightarrow s_j$ corresponds to an edge in E directed from s_i to s_j with label a/o. Thus an edge in E is specified by a triple $(s_i, s_j; a/o)$. An FSM is said to have *reset capability* if for each state s_i in S there exists a transition $(s_i, s_1; r/-)$, called a *reset transition*, which resets the FSM to its initial state, where 'r' denotes the 'reset' command and '$-$' denotes that the FSM does not produce any output for the reset command.

We call an FSM M *completely specified* if, at each state s_i in M and for each input a in I, there is an outgoing transition from s_i with input a. An FSM can be modified into a completely specified one by using what is called a *completeness assumption*[16]. Under this assumption, for each $s_i \in S$ and for each $a \in I$, if s_i does not have an outgoing transition with input a then a self-loop transition $(s_i, s_i; a/-)$ is added to s_i.

We denote an ordered pair (a, b) of input and output by a/b. An input-output sequence is a sequence of input-output pairs. We use the operator '$\bullet$' for concatenating input-output pairs or input symbols. '@' is an operator for concatenating input-output or input sequences. These operators are omitted in certain sequences whenever there is no confusion. An input-output sequence is said to be *applicable* at a state of an FSM if the output part of the sequence is observed on applying the input part of the sequence to the FSM at that state. More formally, a sequence $X = a_1/o_1 \bullet a_2/o_2 \bullet \ldots \bullet a_l/o_l$ is applicable at state s_i iff $\exists s_{i_j} \in S,\ j = 1, 2 \ldots l, l \geqslant 1$ such that $s_i - a_1/o_1 \rightarrow s_{i_1}$ and $s_{i_{j-1}} - a_j/o_j \rightarrow s_{i_j}$ for $2 \leqslant j \leqslant l$.

Suppose $G(S, E)$ is the graphical representation of an FSM M. Then for $E' \subseteq E, G[E']$ will denote the subgraph of G induced by E'. A directed labelled spanning tree rooted at the initial state of G is called a *state cover tree* of M. Note that each state in M can be reached from the initial state using a state cover tree.

A set $C = \{C_1, C_2, \ldots, C_s\}$ of input sequences of an FSM M is called a *characterizing set* if no two states in M have the same set of output sequences when all the sequences from C are applied to them. Formally, suppose $O_i(C_k)$ denotes the output sequence obtained by applying C_k at the state s_i. C is a characterizing set if for any two distinct states s_i and s_j in S, $\{O_i(C_k) | C_k \in C\} \neq \{O_j(C_k) | C_k \in C\}$. Each sequence in C is called a *characterizing sequence*[17]. By applying all the sequences from a characterizing set at an unknown state in M and observing the outputs, one can determine the state.

A set $V_i \subseteq C$ is called an identifying set[17] of the state s_i if V_i is a minimal subset of a characterizing set C such that $\{O_i(v) | v \in V_i\} \neq \{O_j(v) | v \in V_i\}$ for any state $s_j \neq s_i$. Sequences in an identifying set of a state are basically used for distinguishing that state from all other states in M.

An input-output sequence, denoted by UIO_i, is called a Unique Input Output (UIO) sequence[6] of the state s_i of an FSM M if UIO_i is applicable at s_i and it is not applicable at any other state in M. The set $U = \{UIO_i | s_i \in M\}$ is referred to as a *UIO set* of M. Clearly, the set obtained from U by selecting the input part of each sequence in U is a characterizing set for M. For each state s_i, the singleton set containing the input part of UIO_i is an identifying set of s_i.

In our fault model, the IUTs are assumed to have no more than the number of states in the SPEC. As elsewhere[16, 18], our fault model consists of two types of faults, namely, the label fault and the tail state fault. A transition $(s_i, s_j; a/o)$ of the SPEC is said to have a *label fault* if the corresponding transition in the IUT is $(s_i, s_j; a/o')$ where $o' \neq o$. A transition $(s_i, s_j; a/o)$ of the SPEC is said to have a *tail state fault* if the corresponding transition in the IUT is $(s_i, s_p; a/o)$, where $p \neq j$. The *fault coverage* of a test sequence selection method is the percentage of faulty IUTs the method can detect from the set of all IUTs with any number of label faults and/or tail state faults. Thus, a method is said to have *complete fault coverage* if it can detect any faulty IUT.

To detect and diagnose the fault, the input part of each input-output pair of a test sequence is applied to the IUT one-by-one, and the output is observed. If the output is different from that expected, the testing process is stopped and the IUT is declared faulty. The output sequence obtained thus far is analysed for diagnosing the fault. A test sequence selection method has *t-fault resolution capability of level k* if for any IUT with at most t faulty transitions, a test sequence selected by the method can localize at least one faulty transition to within a set of k transitions provided the IUT is faulty. In this paper, we focus on test sequence selection methods with 1-fault resolution capability. A test sequence selection method has *1-fault location*

capability if the sequence selected can exactly locate the fault.

Wp-method

The Wp-method introduced by Fujiwara *et al.*[2] is based on the W-method[12]. The Wp-method assumes that the SPEC is strongly connected, reduced and completely specified. The SPEC and the IUT have a reset capability and the same input set. It further assumes a finite upper bound, say m, on the number of states of the IUT. The Wp-method is described here for $m \leqslant n$.

Let C be a characterizing set of the SPEC. Let $V_i \subseteq C$ be an identifying set of the state s_i in the SPEC for each state s_i. Let T be a state cover tree of the SPEC. Let P_i denote the unique path in T from the initial state to each state s_i. The Wp-method consists of two phases: the state verification phase and the transition testing phase. The state verification phase is mainly to verify whether C is a characterizing set of the IUT. This is done by applying C at each state in the IUT. While a path in the state cover tree T is used for putting the IUT in a given state for applying C, the reset transitions are used for resetting the IUT to the initial state. All the transitions in the state cover tree T are also tested in this phase. In the transition testing phase, all the transitions of the SPEC which are not in the state cover tree T are tested. The tail state of a given transition, say, $(s_i, s_j; a/o) \in E - T$ in the IUT is confirmed by applying every sequence in V_j. Reset transitions, the path P_i, and the transition $(s_i, s_j; a/o)$ are used to reach the state s_j for applying sequences in V_j.

The Wp-method provides complete fault coverage[2]. It is shown[1] that for all IUTs with at most n states, this method has the 1-fault resolution capability of level $n + l_c$, where n is the number of states in the SPEC and l_c is the length of a longest sequence in C. Thus if $U = \{UIO_i \mid s_i \in SPEC\}$ is used as the required characterizing set, then the 1-fault resolution capability of this method becomes $n + l_u$, where l_u is the length of a longest UIO-sequence in U.

It is claimed[2] that if an IUT passes the state verification phase, then:

- Claim 1: all transitions in T are implemented correctly in the IUT, and
- Claim 2: V_i is an identifying set for the state in the IUT corresponding to the state s_i of the SPEC.

We have proved[1] that these two claims may not hold in general. It is also proved[1] that if these claims hold true on successful completion of the first phase, then the level of 1-fault resolution capability of the Wp-method reduces to $1 + l_c$.

DETECTION AND DIAGNOSIS METHOD

In this section, we present our DD-method for selecting test sequences for protocol testing. The method is based on the Wp-method, and so it provides complete fault coverage[2]. By using the test sequence selected in the DD-method, one can detect and diagnose the fault, if any, when the IUT has at most one fault. The DD-method has better 1-fault resolution capability than the Wp-method. Moreover, by incorporating the Rural postperson optimization technique[6], our method minimizes the length of the test sequence. To apply the DD-method on a protocol, the protocol needs to satisfy all the basic assumptions made in the Wp-method. We further assume that the number of states in the IUT is no more than the number of states in the SPEC. However, using an approach similar to that of Fujiwara *et al.*[2], our method can be extended for IUTs with more states than that of the SPEC. In such a case, as in the W- and Wp-methods, the length of the selected test sequence will be greater. For identifying the states we use UIO-sequences which satisfy certain requirements. Our method can very easily be adapted if one wishes to use the characterizing set, distinguishing sequence[17] or any other suitable set of sequences for identifying the states.

The DD-method uses a state cover tree, say T, to reach different states from the initial state. It uses the sequences from a UIO set, say U, for checking the tail states of transitions. Unless otherwise stated, by a UIO-sequence we refer to a sequence in U. A transition is called a *T-transition* if it is on the state cover tree T. A transition is called a *U-transition* if it is part of a UIO-sequence. As noted earlier[1], for better fault resolution it is necessary that we test all the T-transitions before using them for testing other transitions. It is also necessary to verify if the UIO-sequences are also the UIO-sequences of the corresponding states in the IUT. Meeting these two requirements is difficult, as T-transition testing requires verified UIO-sequences and UIO-sequence verification requires tested T-transitions. This could be achieved if T and U have the *Tree UIO-sequence Label Disjoint* (TULD) property defined below:

> For each transition $(s_i, s_j; a/o)$ in T, the label a/o does not appear in the UIO-sequence UIO_j corresponding to the state s_j.

We have analysed a number of protocols (reported later), and found that they have T and U with the TULD property. However, there may exist some protocols which do not have such T and U.

The DD-method is described in the algorithm U-DD. This algorithm first invokes the procedure *constreeuio* (to be discussed) for computing a state cover tree and a UIO set with the TULD property. The algorithm then invokes the procedure *select_seq* for selecting the required test sequence. The procedure *select_seq* consists of two phases. In the first phase, T-transitions are tested in a breadth-first fashion by applying all the sequences in U at the tail state of each transition under test. The reset transition and the unique path P_i from the initial state to the starting state s_i of the transition under test are used for reaching the transition. It is

established later in this section that the test sequence selected in the above scheme is also sufficient for verifying if the UIO-sequences are also the UIO-sequences of the corresponding states in the IUT.

The second phase consists of two steps. In the first step all U-transitions which are not T-transitions are tested. We use the unique path in T for reaching the U-transitions. In the second step, we test all transitions which are neither T-transitions nor U-transitions. At the end of this step, all the transitions in the state cover tree and in the UIO-sequences have been successfully tested. At this point we can obtain an optimal test sequence for testing the remaining transitions by applying the RPT algorithm of Aho *et al.*[6], provided certain conditions, described below, are satisfied. This optimization step does not, however, affect the fault detection or fault resolution capability. Let $E_c = \{(s_i, s_p; a/o@UIO_j) | (s_i, s_j; a/o) \in E - (E(T) \cup E(U) \cup E_r) \wedge tail(UIO_j) = s_p\}$, where $E(T)$, $E(U)$ and E_r denote the sets of all T-transitions, U-transitions and reset transitions, respectively. $tail(UIO_j)$ is the state of the SPEC after applying UIO_j at s_j. Let G' be the graph defined by $G'(S, E_c \cup E(T) \cup E(U) \cup E_r)$. An optimal test sequence for testing the remaining transitions (i.e. $E - E(T) \cup E(U) \cup E_r$) corresponds to a minimal tour, called a rural postperson tour (RPT)[19] of the graph $G'(S, E(T) \cup E(U) \cup E_r)$, which covers each edge in E_c at least once. This RPT can be obtained using the efficient RPT-algorithm of Aho *et al.*[6] provided $G'[E_c]$ is weakly connected. If $G'[E_c]$ is not weakly connected, then we test the transitions in $E - (E(T) \cup E(U) \cup E_r)$ as in the first step of this phase. The algorithm *U_DD* is formally described below. At any step in the procedure *select_seq*, if the observed output is different from the one expected, the procedure terminates:

Algorithm *U-DD*(SPEC, IUT)
constreeuio(SPEC, MU, T, U);
select_seq(SPEC, IUT, T, U);
end *U_DD*.

procedure *select_seq*(SPEC,IUT,T,U)
Phase 1:
Test the state cover tree T in a breadth-first fashion as follows until all the T-transitions are tested:
(i) Let $(s_i, s_j; a/o) \in T$ be the current transition;
(ii) Apply r to reset the IUT into the initial state s_1;
(iii) Use the unique path P_i in T to reach s_i from s_1;
(iv) Apply a and observe o;
(v) Apply UIO_j at the tail state to check that UIO_j is applicable;
(vi) Apply $U - \{UIO_j\}$ at the tail state and check that they are not applicable. Use the reset transitions and the state cover tree T to reach the state s_j;
Phase 2:
Step 1 Repeat the following until all the U-transitions are covered:
(i) Let $(s_i, s_j; a/o)$ be an untested U-transition;
(ii) Apply r to observe $-$;
(iii) Use the unique path P_i in T to reach s_i from s_1;
(iv) Apply a and observe o.
(v) Apply UIO_j at the tail state and check that it is applicable;
Step 2:
(i) Compute $E_c = \{(s_i, s_p; a/o@UIO_j) | (s_i, s_j; a/o) \in E - (E(T) \cup E(U) \cup E_r) \wedge tail(UIO_j) = s_p\}$
(ii) If $G'[E_c]$ is weakly connected in $G'(S, E_c \cup E(T) \cup E(U) \cup E_r)$ then
 (a) Apply the RPT algorithm to compute a minimal tour Γ in $G'(S, E_c \cup E(T) \cup E(U) \cup E_r)$ which traverses each transition in E_c at least once.
 (b) Test the transitions in $E - (E(T) \cup E(U) \cup E_r)$ using the input-output sequence along Γ.
(iii) If $G'[E_c]$ is not weakly connected in $G'(S, E_c \cup E(T) \cup E(U) \cup E_r)$ then repeat the following until all the transitions in $E - (E(T) \cup E(U) \cup E_r)$ are tested
 (a) Let $(s_i, s_j; a/o)$ be an untested transition in $E - (E(T) \cup E(U) \cup E_r)$;
 (b) Apply r to observe $-$;
 (c) Use the unique path P_i in T to reach s_i from s_1;
 (d) Apply a and observe o.
 (e) Apply UIO_j at the tail state and check that it is applicable;
end *select_seq*.

We will shortly give a procedure for finding a state cover tree T and a UIO-set U satisfying the TULD property.

The following theorem shows that the successful completion of phase 1 of the procedure *select_seq* will guarantee a fault-free state cover tree and a verified set of UIO-sequences for the IUT.

Theorem 1 *Suppose that an IUT has at most* 1 *fault and it passes phase* 1 *of* select_seq *successfully; then the following are true:*

1. *The state cover tree T obtained from the SPEC is fault-free in the IUT.*
2. *The UIO-sequence of each state of the SPEC from the set U is also a UIO-sequence of the corresponding state in the IUT.*

Proof We will prove the first part by induction on the level number of the state cover tree T. We claim that all transitions of level 1 are fault-free in the IUT. Let $(s_1, s_i; a/o)$ be a T-transition of level 1. Suppose it has a label fault in the IUT. Then when we apply the input a, we will get some output $o' \neq o$. Therefore, the IUT will fail in phase 1. Suppose the transition has a tail state fault in the IUT. Let its faulty tail state be s_j, where $j \neq i$. In other words, the corresponding transition in the IUT is $(s_i, s_j; a/o)$. Since, by the TULD property, no transition in the sequence UIO_i has the label a/o, and since there is only one fault which is at the transition $(s_1, s_i; a/o)$, it follows that when UIO_i is applied at the faulty tail state s_j, it will not be applicable. Therefore the IUT will fail in phase 1. Assuming all the T-transitions up to level l are fault-free, we have to prove that all the T-transitions of level $l+1$ are also fault-free. Let $(s_i, s_j; a/o)$ be the current T-transition under test of level $l+1$. Since all the transitions up to level l are fault-free, we can reach s_i by traversing the fault-free path P_i from s_1. It is easy to see that phase 1 fails if the IUT has a label fault in $(s_i, s_j; a/o)$. Suppose $(s_i, s_j; a/o)$ has a tail state fault and let its tail state in the IUT be s_k. Let us observe the behaviour of the IUT for the sequence $P_i@a/o@UIO_j$. Clearly, $P_i@a/o$ puts the IUT in the state s_k. When UIO_j is applied at s_k, the

IUT traverses only fault-free transitions, as the label (a/o) of the unique faulty transition does not appear in UIO_j. Thus, the observed output is different from the expected output, and the IUT fails. This completes the induction.

Since only T-transitions are used to reach the states in order to apply the UIO-sequences, it follows from the correctness of T and from steps (iv) through (vi) of phase 1 that the UIO-sequence of each state in the SPEC is also a UIO-sequence of the corresponding state in the IUT if the IUT passes phase 1. □

The following theorem establishes the 1-fault resolution capability of our DD-method. Here, by UIO_k^j we refer to the input-output sequence obtained when the input part of UIO_k is applied at the state s_j:

Theorem 2 *Suppose an IUT has at most one fault; then the DD-method detailed in the algorithm* U-DD *has 1-fault resolution capability of level* $1 + l_u$, *where* l_u *is the length of the longest UIO-sequence in the set U.*

Proof Suppose the IUT fails in phase 1 while applying the sequence $P_i@a/o@UIO_k^j$, where $P_i@a/o@UIO_k^j$, for some k, $1 \leqslant k \leqslant n$, is a part of the sequence for testing the T-transition $(s_i, s_j; a/o)$ of the SPEC and verifying the UIO-set U at the state s_j in the IUT. Here, P_i denotes the sequence along the unique path in T from s_1 to s_i, and UIO_k^j is a sequence either to check the applicability of UIO_j (if $j = k$) or to check the non-applicability of the UIO-sequence UIO_k (if $j \neq k$) at state s_j. Since the T-transitions are traversed in breadth-first fashion, it follows from the TULD property of T and U and the single fault assumption that none of the transitions in P_i is faulty. If an output o' is observed instead of o while applying the input a of the T-transition $(s_i, s_j; a/o)$, then the T-transition has a label fault. Otherwise, either the T-transition has a tail state fault or a transition in the path from s_j along the sequence UIO_k^j is faulty in the IUT. Thus the fault can be diagnosed within $1 + l_u$ transitions.

Suppose the IUT fails in step 1 of phase 2 while applying the test sub-sequence $TEST(s_i, s_j; a/o) = P_i@a/o@UIO_j$. Since all the transitions in P_i have already been tested for correctness, and as the UIO-sequences have also been verified in phase 1, the fault can only be in the transition corresponding to $(s_i, s_j; a/o)$ of the SPEC. Now suppose the IUT fails in step 2 of phase 2 while applying the test subsequence $TEST(s_i, s_j; a/o)$ corresponding to the transition $(s_i, s_j; a/o)$. Since all the transitions except $(s_i, s_j; a/o)$ in this subsequence have already been confirmed to be fault-free, it is easy to see that the transition $(s_i, s_j; a/o)$ of the SPEC is faulty in the IUT. Thus if the IUT fails in the second phase then the faulty transition can be diagnosed exactly. □

Observe that the level of 1-fault resolution capability of the DD-method is an improvement over that of the Wp-method by n transitions. As noted by Sabnani *et al.*[20], $l_u \leqslant 5$ for most of the known protocols. Therefore, from Theorem 2 we can deduce that for most of the known protocols, our method localizes the fault to within six transitions in the worst case. Due to the optimization performed in the second step of phase 2, the length of the test sequence selected is less than that selected in the Wp-method. The DD-method provides complete fault coverage, since it is based on the W-method, which is known to provide complete fault coverage[2].

Next, we consider the problem of extracting a state cover tree T and a UIO-set U with the TULD property. Let MU_i be the set of multiple UIO-sequences[21] for the state s_i. Let MU be the collection of all the UIO-sequences in MU_i for every state s_i. The procedure given below constructs T and U with the TULD property, provided such a T and U exist for the given SPEC and MU:

```
procedure constreeuio(SPEC,MU,T,U);
  VT ← {s1};
  NT ← {s1};
  U → {UIO1}, for some UIO1 ∈ MU1;
  repeat
    delete a state si from NT;
    for each outgoing (si, sj; a/o) at si such that sj ∉ VT do
      for each UIOj in MUj do
        if the label a/o is not in UIOj then
        begin
          U ← U ∪ {UIOj};
          VT ← VT ∪ {sj};
          NT ← NT ∪ {sj};
          T ← T ∪ {(si, sj; a/o)};
          break;
        end
  until (VT = V)or(NT = ∅)
  /* if NT = ∅ and VT ≠ V then there is no T and U
  satisfying the required condition for the given G and
  MU*/
end constreeuio.
```

The algorithm guarantees the construction of the required state cover tree and a UIO-set if they exist for the given SPEC, and the given MU. The complexity of the procedure *constreeuio* is $O(|E|n^2 \mu_{max})$, where $n, |E|, \mu_{max}$ are the number of states in the SPEC, the number of transitions in the SPEC and the maximum number of multiple UIO-sequences (from the set MU) of any state in the SPEC, respectively. Here we have used the result of Sabnani *et al.*[20] that a UIO-sequence is of length at most $2n^2$. However, as pointed out earlier, the UIO-sequences for most of the practical protocols are of length at most 5. In such cases, the complexity of our algorithm is $O(|E|\mu_{max})$. Sabnani *et al.*[20] have also presented an algorithm for finding a UIO-sequence of a state of a SPEC. The time complexity of this algorithm is $O(n^2(d_{max})^{2n^2+2})$, where d_{max} is the maximum outdegree of any state in the SPEC. The procedure *constreeuio* can be combined with this algorithm to construct a state cover tree T, and a UIO-set U in a single pass, such that T and U satisfy the TULD property, if such a T and U exist.

In the remaining part of this section, we compare the DD- and Wp-methods with respect to their 1-fault

resolution capabilities. We apply these methods on an abstract protocol of reasonable complexity, and a subset of a transport protocol. The SPEC of the abstract protocol is shown in *Figure 1a*. Clearly, $W = axbxx$ forms the unique element of a characterizing set. With this characterizing set and the state cover tree shown in *Figure 1b*, the Wp-method generates a test sequence of length 301 (input interactions). The *constreeuio* procedure of the DD-method produces the state cover tree T, and the UIO-set $U = \{UIO_j = W | 1 \leqslant j \leqslant n\}$ for the input $MU_j = \{W\}$ for $j = 1, 2, \ldots, n$. With this state cover tree and the UIO-set, the first phase of the *select_seq* results in the following test sequence of length 67. We have only shown the input symbols for convenience. Here, P_i denotes the input sequence along the unique path in the state cover tree T from s_1 to s_i for $i = 2, 3, \ldots, 8$:

$$rWrP_6WrP_7WrP_8WrP_4WrP_2WrP_3WrP_5W$$

The test subsequence for testing the U-transitions has 151 input symbols. Only a prefix of this sequence is shown below:

$$rxWraWrP_7aWrP_7bWrP_7xWrP_8aWrP_8bWrP_8xW$$

There are nine transitions in $E - (E(T) \cup E(U) \cup E_r)$ and E_c induces a weakly connected subgraph in $G'(S, E_c \cup E(T) \cup E(U) \cup E_r)$. So, the RPT-algorithm of Aho *et al.*[6] is applicable. The RPT-algorithm yields the following test sequence of length 70 for testing these transitions:

$$rcWcWcx\mathbf{W}ccWcbbWbcbcWrxbxcWxcWrxcWr$$

Thus, the total length of the test sequence selected in the DD-method is 288. Now let us evaluate how these two methods do in diagnosing the fault in an implementation. Consider the IUT as shown in *Figure 2*. There is a tail state fault in the transition $(s_3, s_3; x/0)$. The corresponding faulty transition is $(s_3, s_1; x/0)$ in the IUT. The IUT passes the test sequence selected in the first phase of the Wp-method. However, the IUT is failed in the second phase while applying the sequence *rbbxcxaxbxx*. The IUT gives the output sequence -2000000101, whereas the expected sequence is -2000000001. Since only the reset transition is known to be fault free in the IUT, the fault could be at any one of the 10 transitions along the path from state s_1 with input sequence *bbxcxaxbxx*. The IUT also passes the first phase and the first step of the second phase of the DD-method. Since $(s_3, s_3; x/0) \in E - (E(T) \cup E(U) \cup E_r)$, this transition is tested only in the second step of the second

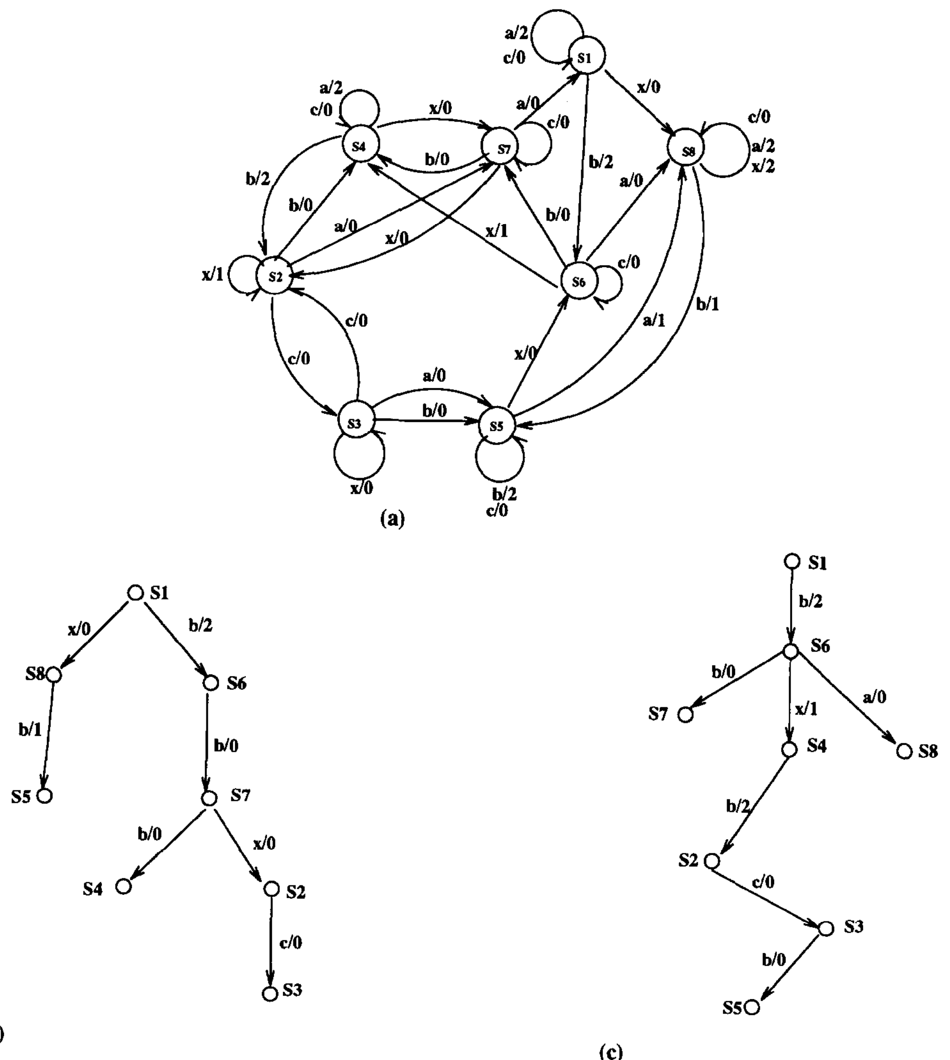

Figure 1 Specification and the state cover trees of a sample protocol. (a) SPEC; (b) state cover tree in Wp-method; (c) state cover tree in DD-method

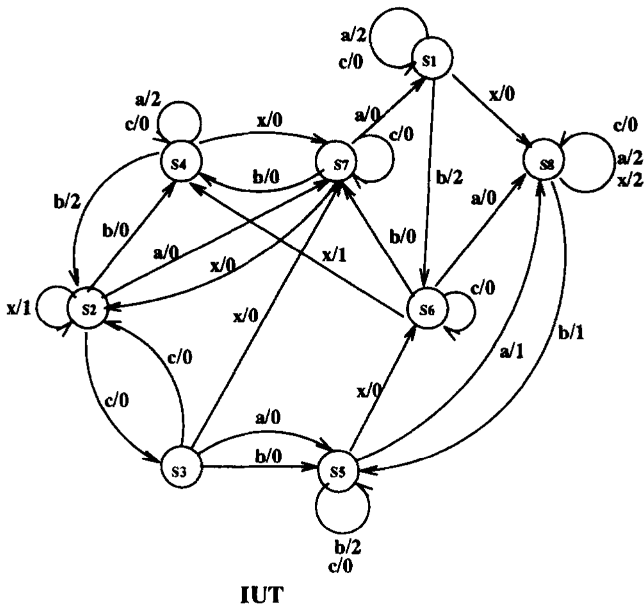

Figure 2 Implementation of the protocol given in *Figure 1a*

phase, with the sequence xW shown in bold in the subsequence of this step. As expected, the IUT fails while applying $xaxbxx$ at state s_3 of the IUT. It is known at this stage that there is no fault in any of the T- or U-transitions. Thus, the transition $(s_3, s_3; x/0)$ is uniquely identified as the faulty transition. Also, in the worst case, our DD-method can localize the fault in any IUT of the abstract protocol (*Figure 1a*) within six transitions. We have also applied the Wp- and DD-methods for a subset of the class 4 transport protocol, NBS TP4, developed by the National Bureau of Standards[22]. Sidhu *et al.*[13] have analysed this protocol for studying different formal methods of test sequence selection based on the FSM model. The protocol has 15 states and 61 core transitions. The state cover tree, UIO-set and the full test sequence selected using the DD-method have been reported elsewhere[23]. The test sequence contains 1146 input interactions. In the worst case, this test sequence localizes the fault in an IUT to within three transitions. For the same state cover tree and the UIO-set (now considered as the characterizing set), the Wp-method selects a test sequence having 1213 input interactions. This test sequence localizes the fault to within seven transitions in the worst case. From these two case studies, we infer that the DD-method improves the 1-fault resolution capability of the Wp-method by about 40–57%.

IMPROVEMENTS OF THE DD-METHOD

As we know, the length of the test sequence and the resolution of the fault are two important factors in any fault diagnosis method. In this section, we present some improvements of the DD-method for further minimizing the length of the test sequence and for achieving 1-fault location capability.

Length minimization

In step 1 of the second phase of *select_seq*, each untested U-transition, say, $(s_i, s_j; a/o)$ is tested by applying $a/o@UIO_j$ after putting the IUT in state s_i using a reset transition and the unique path P_i in T from s_1 to s_i. The purpose of this step is mainly to ensure that the RPT optimization at step 2 does not affect the fault diagnosis capability. This can also be achieved in step 1 by testing if the UIO-sequence of each state ends at the correct state in the IUT without explicitly testing each transition in the UIO-sequences. Thus, the testing of U-transitions can be subjected to the RPT optimization, and the length of the resulting test sequence is, in most cases, much smaller than that selected in the original method. The required modifications in the second phase of *select_seq* are provided below:

Phase 2
  Step 1
    for each state $s_i \in S$ **do**
    begin
      Apply r to observe $-$;
      Use P_i to reach s_i from s_1;
      Check if $UIO_i@UIO_{tail(UIO_i)}$ is applicable;
    end
  Step 2
    Compute $E_c = \{(s_i, s_p; a/o@UIO_j) | (s_i, s_j; a/o) \in E - (E(T) \cup E_r) \wedge tail(UIO_j) = s_p\}$
    If $G'[E_c]$ is weakly connected in $G'(S, E_c \cup E(T) \cup E_r)$ **then**
    begin
      Apply the RPT algorithm to compute a minimal tour Γ
      in $G'(S, E_c \cup E(T) \cup E_r)$ which traverses each transition in E_c at least once;
      Test the transitions in $E - (E(T) \cup E_r)$ using the input-output sequence along Γ;
    end
    If $G'[E_c]$ is not weakly connected in $G'(S, E_c \cup E(T) \cup E_r)$ **then**
    for each transition $(s_i, s_j; a/o)$ in $E - (E(T) \cup E_r)$ **do**
    begin
      Apply r to observe $-$;
      Use the unique path P_i in T to reach s_i from s_1;
      Apply a and observe o;
      Apply UIO_j at the tail state and check that it is applicable;
    end

Since there is a maximum of nl_u U-transitions in the SPEC which are now tested using the RPT algorithm, the modified method in most cases improves the length of the test sequence significantly. If the IUT fails in the first step of phase 2 while applying the sequence $r/-@P_i@UIO_i@UIO_{tail(UIO_i)}$, then the fault is in one of the transitions along UIO_i at s_i. Thus, the improved method maintains the 1-fault resolution capability of the original DD-method.

Fault localization

We define a set containing a faulty transition as a *fault resolution set*. We know that if the IUT fails in the first

phase of *select_seq*, then the fault is located within $1 + l_u$ transitions. Suppose the IUT fails while testing the transition $(s_i, s_j; a/o)$ with the subsequence $r/-@P_i@a/o@UIO_k^j$ for some k, $1 \leqslant k \leqslant n$. Let F be the set containing the transition $(s_i, s_j; a/o)$ and the transitions in the path from s_j along the sequence UIO_k^j. From the proof of Theorem 2, clearly F is a fault resolution set. The faulty transition in F can be further localized by repeatedly applying *select_seq* using state cover trees and a UIO-set with the TULD property such that the sequences selected in the first phase of *select_seq* do not involve at least one transition from F. The procedure *localize_fault* for selecting a test sequence for improving the fault resolution is given below:

```
procedure localize_fault(F,MU,SPEC,IUT);
  let each transition in F be unmarked;
  while (|F| > 1 and F has an unmarked transition) do
  begin
    choose and mark an unmarked transition e in F;
    for j = 1 to n do
      MU_j^e ← {UIO_j ∈ MU_j | e ∉ path from s_k along
        UIO_j^k, 1 ≤ k ≤ n};
    MU^e ← MU_1^e ∪ MU_2^e ∪ ... ∪ MU_n^e;
    constreeuio(SPEC − e, MU^e, T_e, U_e);
    if (constreeuio is successful) then
    begin
      select_seq(SPEC, IUT, T_e, U_e);
      let F_e be the resulting fault resolution set;
      F ← F ∩ F_e;
    end
  end
end localize_fault.
```

At each iteration of the **while** loop an unmarked transition e in F is marked and, if possible, a state cover tree T_e and a UIO-set U_e are selected such that they satisfy the TULD property, and that neither T_e nor any path from s_k along UIO_j^k for $1 \leqslant j, k \leqslant n$ contains the transition e. If such T_e and U_e are found, then the IUT is tested with the corresponding test sequence and a fault resolution set F_e is obtained. If the IUT fails in the first phase then as shown in the proof of Theorem 2 it follows that $e \notin F_e$. Thus the number of transitions in the fault resolution set is reduced by at least one, since $|F \cap F_e| < |F|$ before the execution of the statement $F \leftarrow F \cap F_e$. On the other hand, if the IUT fails in the second phase then as seen in the proof of Theorem 2 we can exactly locate the fault. Thus using *localize_fault* the fault resolution can be improved significantly or the fault can be located exactly. Note that the set MU^e selected in the procedure *localize_fault* may sometimes be empty. In such a case MU^e can be selected as follows:

$$MU^e = MU_1^e \cup MU_2^e \ldots \cup MU_n^e, \text{ where}$$
$$MU_j^e = \{UIO_j \in MU \mid e \notin \text{path from } s_j \text{ along } UIO_j\} \text{ for } 1 \leqslant j \leqslant n$$

Though with such a less restricted set MU^e the fault resolution set is reduced less rapidly, the resulting fault resolution set will still be significantly smaller.

CONCLUSION

Based on the Wp-method, we have developed the DD-method which has a better 1-fault resolution capability than the Wp-method when the IUT has at most one fault and the SPEC meets certain conditions. Our method also results in a test sequence of a shorter length. This method is useful for testing and diagnosing any fault in the implementations. Approaches for further minimizing the test sequence and for exactly locating the fault or improving the fault resolution capability of our method are also presented.

Our DD-method achieves good 1-fault resolution capability on a SPEC if the SPEC has a state cover tree and a UIO-set with the TULD property. Interestingly, such a tree and a UIO-set exist for the simplified transport protocol which we have used for evaluating our method. We have also found that a few other protocols, such as the ISDN-BRI-D-Channel signalling protocol (network-interface side, originating end)[24], a simplified transport protocol[25] and the alternating bit protocol[20], satisfy the required conditions for the DD-method.

We wish to note that our DD-method will also work for the SPEC which has a state cover tree T and a set U of UIO-sequences satisfying the following condition: for each tree transition, say $(s_i, s_j; a/o)$, either the label a/o does not appear in the UIO-sequence UIO_j or the transition $(s_i, s_j; a/o)$ is not a part of the UIO-sequence of any state other than s_j. Note that the above condition includes our earlier requirement. However, finding T and U satisfying this requirement is not easy. Further work on this issue is under way.

An interesting open problem is to determine necessary and sufficient conditions on the structure of SPECs for which our DD-method will ensure 1-fault location capability. In view of our development above, we believe that these conditions are tied to the existence of the appropriate T_e and MU^e, as described in the procedure *localize_fault*. We are also investigating for test sequence selection methods which can diagnose multiple faults.

REFERENCES

1 Ramalingam, T, Das A and Thulasiraman, K 'Analysis of fault detection and diagnosis capabilities of test sequence selection methods based on the FSM model', *Comput. Commun.*, Vol 18 No 2 (February 1995)

2 Fujiwara, S, von Bochmann, G, Khendek, F, Amalou, M A and Ghedamsi, A 'Test selection based on finite state model', *IEEE Trans. Softw. Eng.*, Vol 17 (June 1991) pp 591–603

3 Liu, M T 'Protocol engineering', in M C Yovits (ed), *Advances in Computers*, Academic Press, New York (1989) pp 79–195

4 Stallings, W *Networking Standards: a guide to OSI, ISDN, LAN, and MAN*, Addison-Wesley, New York (1993)

5 ISO/IEC 9646 *Information Technology – Open Systems Interconnection – Conformance Testing Methodology and Framework*, ISO, Geneva, Switzerland (1991)

6 Aho, A V, Dahbura, A T, Lee, D and Uyar, M U 'An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours', *Symposium of Protocol Specification, Testing and Verification* (1988) pp 75–86

7 Bolognesi, T and Brinksma, E 'Introduction to the ISO specification language LOTOS', *Comput. Networks ISDN Syst.*, Vol 14 (1987) pp 25–59

8 Budkowski, S and Dembinski, P 'An introduction to Estelle: a specification language for distributed systems', *Comput. Networks ISDN Syst.*, Vol 14 (1987) pp 3–23

9 CCITT/SGx/WP3-1 *SDL, specification and description language*, CCITT Recommendations Z.100-Z.104 (1988)

10 ISO SC21 WG1 P54 *Information Technology - Open Systems Interconnection - Formal Methods in Conformance Testing*, Working Document, ISO, Geneva, Switzerland (June 1993)

11 von Bochmann, G and Sunshine, CA 'A survey of formal methods', in P E Green (ed), *Computer Networks and Protocols*, Plenum Press, New York (1983) pp 561–578

12 Chow, T 'Testing software design modeled by finite state machine', *IEEE Trans. Softw. Eng.*, Vol 4 (March 1978) pp 178–187

13 Sidhu, D P and Leung, T K 'Formal methods for protocol testing: A detailed study', *IEEE Trans. Softw. Eng.*, Vol 15 No 4 (1989) pp 413–426

14 Chan, W Y L, Vuong, S T and Ito, M R 'An improved protocol test generation procedure based on UIOs', *ACM SIGCOMM* (1989) pp 283–293

15 Ural, H 'Formal methods for test sequence generation', *Comput. Commun.*, Vol 15 No 5 (June 1992) pp 311–325

16 Dahbura, A T and Sabnani, K K 'An experience in estimating fault coverage of a protocol test', *IEEE INFOCOM* (March 1988) pp 71–79

17 Kohavi, Z *Switching and Finite Automata Theory*, McGraw-Hill, New York (1978)

18 von Bochmann, G, Das, A, Dssouli, R, Dubuc, M, Ghedamsi, A and Luo, G 'Fault model in testing', *4th Int. Workshop Protocol Test Systems*, Leischendam, The Netherlands (October 1991)

19 Edmonds, J and Johnson, E L 'Matching, Euler tours and the Chinese postman', *Math. Program.*, Vol 5 (1973) pp 88–124

20 Sabnani, K and Dahbura, A 'A protocol test generation procedure', *Comput. Networks ISDN Syst.*, Vol 15 (1988) pp 285–297

21 Shen, Y N, Lombardi, F and Dahbura, A T 'Protocol conformance testing using multiple UIO sequences' *IEEE Trans. Commun.*, Vol 40 (August 1992) pp 1282–1287

22 National Bureau of Standards. *Specification of a transport protocol for computer communications, vol. 3: class 4 protocol*, Report ICST/HNLP-83-4, NBS, Washington, DC (January 1983)

23 Ramalingam, T, Das, A and Thulasiraman, K 'On conformance test and fault resolution of protocols based on FSM model', *Proc. Int. Conf. Comput. Networks, Architecture and Applic.*, Trivandrum, India (October 1992)

24 Dahbura, A T, Sabnani, K K and Uyar, M U 'Algorithmic generation of protocol conformance tests', *AT&T Tech. J.* (January/February 1990) pp 101–118

25 von Bochman, G 'Specifications of simplified transport protocol using different formal description techniques', *Comput. Networks ISDN Syst.*, Vol 17/18 (1990) pp 335–377