Generating Maximal Fault Coverage Conformance Test Sequences of Reduced Length for Communication Protocols

Raymond E. Miller

Department of Computer Science
University of Maryland
College Park, MD 20742
and
CESDIS, NASA
Greenbelt, MD 20771

Sanjoy Paul

AT&T Bell Laboratories
Murray Hill, NJ 07974

Abstract

This paper focuses on a technique to reduce the length of maximal fault coverage test sequences for communication protocols by removing redundant test segments. This approach conceptually begins with all the test segments needed for the generation of maximal fault coverage test sequences as in [MP92], analyzes the structure of the specified Finite State Machine (FSM) for the protocol and shows that certain segments in these tests are unnecessary to guarantee maximal fault coverage. From this analysis an algorithm is proposed for generating the reduced length sequences that still guarantee maximal fault coverage. We describe how these tests are in some sense minimal, or near minimal, length test sequences without losing fault coverage.

1 Introduction

Conformance testing of communication protocols is a problem of considerable complexity. Nevertheless, it is important since it helps insure proper operation of networks. One of the major aspects of conformance testing is the generation of test suites for the control portion of the protocol. A lot of research has taken place in the area of generating test segments for individual transitions in the finite state machine (FSM) based description of a communication protocol using characteristic sets [G70] , distinguishing sequences [NT81] and UIO sequences [SD88] including combining them in an optimal way [SLD89]. Attempts have been made to reduce the overall length of the test sequence even further using overlapping [CCK90], [YU90], [MP91], [MP93]. Some work has also been done to assess the fault coverage of the test sequences [SD88] [SL88]. Although the term *fault coverage* in the context of conformance testing of communication protocols has a clear intuitive meaning, a precise definition of *fault coverage* closely related to that in [CS90] is given in [MP92] and some techniques are presented for the generation of test sequences which guarantee maximal fault coverage.

Although the fault detecting capability of a test sequence is maximized, the price paid is in terms of its length. In this paper, further analysis of the specification machine provides insight into how redundant test segments can be eliminated to reduce the test sequence length. The non-redundant test segments thus obtained are combined to generate a near minimal length test sequence with maximal fault coverage.

This paper is organized as follows. In section 2, the model is discussed. Section 3 introduces Pairwise Distinguishing Sequence (PDS), briefly discusses the generation of a minimal length test sequence and a maximal fault coverage test sequence and gives the motivation for reducing the length of a test sequence without losing any fault coverage. Section 4 covers the theoretical basis for removing redundant test segments while section 5 contains the final algorithm and an example. Section 6 compares our technique with those existing in the literature. We conclude this paper with a summary of our contribution and areas requiring further research.

0-8186-3670-X/93 $03.00

2 The Model

A protocol may be specified by a deterministic FSM with a finite set of states $S = \{s_1, .., s_n\}$, a finite set of inputs $I = \{i_1, .., i_k\}$ and a finite set of outputs $O = \{o_1, .., o_m\}$. The next state (NS) and output (Z) functions are given by a set of mappings $NS : S \times I \rightarrow S$ and $Z : S \times I \rightarrow O$

Usually an FSM is represented by a directed graph $G = (V, E)$ where $V = \{v_1, .., v_n\}$ represents the states $\{s_1, .., s_n\}$ and a directed edge (v_i, v_j) with an input/output label i_p/o_q represents a transition from state s_i to state state s_j under input i_p generating output o_q. The FSM is assumed to be deterministic, implying that there can be no two edges leaving a given state s_i with the same input i_p. Also the specification is *incompletely specified* , that is, it does not necessarily define for every state s_i and input i_p a next state.

An implementation of an FSM is assumed to be a black box with input and output ports. The internal structure of the machine is neither observable nor controllable. Usually each state of the machine has a unique I/O behavior. A sequence of input/output labels is used to identify a given state indirectly. The unique input output (UIO) sequence of a state is a string of input output labels such that the state generates the outputs specified in the sequence when the corresponding input sequence is applied. It is called unique because no other state exhibits the same input output behavior.

Testing conformance of an implementation to the specified FSM is carried out by checking each transition separately, e.g, having placed the machine in state s_i, a transition from state s_i to state s_j under input/output label i_p/o_q is carried out with the following test subsequence: i_p/o_q @ UIO_j where @ stands for string concatenation and UIO_j refers to the unique input/output sequence for state s_j. Thus there are as many test subsequences as the number of transitions in the specification.

These test subsequences are combined to form a test sequence. Thus, generation of a conformance test sequence is aimed at seeing if the input/output transitions defined in the specification machine (an incompletely defined machine) are actually realized in the machine implementation (a completely defined machine) under test. I.e, only transitions in the specification are tested.

3 Motivation for Reducing Length without Reducing Fault Coverage

In [MP92], a technique to generate maximal fault coverage test sequences is provided and proven to be correct. Such a test sequence is generated by combining all possible test segments corresponding to each transition. In this paper, we repeat the definition of pairwise distinguishing sequence (PDS) introduced in [MP92] for the convenience of readers.

Definition 3.1 A Pairwise Distinguishing Sequence for states A and B *in the Specification* denoted by PDS^{spec}_{AB} is an input/output sequence defined from A in the specification that gives an output behavior for A and a different output behavior for B *in the specification* but no prefix of this sequence has this property.

Note that PDS^{spec}_{AB} is different from PDS^{spec}_{BA} because PDS^{spec}_{AB} is an i/o sequence *defined from A* in the specification while PDS^{spec}_{BA} is an i/o sequence *defined from B* in the specification.

In this paper, the notation PDS_{AB} will be used to denote PDS^{spec}_{AB} and the notation PDS_A (or PDS of state A) will be used to refer to a PDS_{AX} for some state X.

A test segment (subsequence) corresponding to a transition is the transition followed by an input/output sequence used to check the ending state of the transition. An input/output sequence obtained by combining the test segments will be referred to as a *test sequence* unless mentioned otherwise.

Usually, a test segment (subsequence) is generated corresponding to a transition by concatenating a unique input/output (UIO) sequence of the tail state of the transition to the input/output label of the transition. However, for generating maximal fault coverage test sequences, PDS's are used, rather than UIO's, for generating test segments[1]. Then a test sequence is generated by combining all the test segments. It is assumed that if a transition of the specification is not implemented correctly in the implementation, then it is a fault and it will be detectable by the test sequence.

In this section, a finite state machine (FSM) specification will be considered and the following test sequences will be generated:

1) A minimal length test sequence and

2) A maximal fault coverage test sequence

This example will illustrate why the minimal length test sequence in (1) does not have good fault coverage.

[1] In this paper, a *test segment* corresponding to a transition refers to a concatenation of the transition and a PDS of the ending state of the transition.

On the other hand, the test sequence in (2) is shown to contain redundancy. This illustrates the possibility of striking a balance between minimality of length and maximality of fault coverage.

Example 1: The specification FSM in Fig.1 does not have *convergence*[MP91]. It has an Euler path.

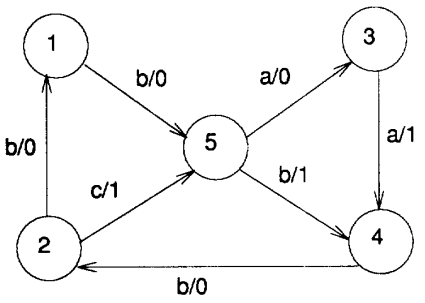

Figure 1: Specification FSM for Example 1

Therefore, a minimal length test sequence can be obtained using the algorithm given in [MP91]. It is given by:

$$2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5 \xrightarrow{a/0} 3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5 \xrightarrow{b/1} 4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5$$

To compute the maximal coverage test sequence , the following PDS's are needed:

$PDS_{13} = PDS_{15} = \{b/0\}$ $PDS_{12} = PDS_{14} = \{b/0\ a/0,\ b/0\ b/1\}$

$PDS_{21} = PDS_{24} = \{c/1\}$ $PDS_{23} = \{b/0\}$

$PDS_{25} = \{b/0,\ c/1\}$

$PDS_{31} = PDS_{34} = PDS_{35} = \{a/1\}$

$PDS_{32} = \{a/1\ b/0\}$

$PDS_{41} = \{b/0\ b/0,\ b/0\ c/1\}$

$PDS_{42} = \{b/0\ c/1,\ b/0\ b/0\ b/0\}$

$PDS_{43} = PDS_{45} = \{b/0\}$

$PDS_{51} = PDS_{52} = PDS_{53} = PDS_{54} = \{a/0,\ b/1\}$

Using the above set of pairwise distinguishing sequences (PDS's) the following test segments can be obtained:

$2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5,\ 2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5 \xrightarrow{a/0} 3,\ 2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5 \xrightarrow{b/1} 4$

$4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5,\ 4 \xrightarrow{b/0} 2 \xrightarrow{b/0} 1$

$5 \xrightarrow{a/0} 3 \xrightarrow{a/1} 4,\ 5 \xrightarrow{a/0} 3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2$

$5 \xrightarrow{b/1} 4 \xrightarrow{b/0} 2 \xrightarrow{b/0} 1,\ 5 \xrightarrow{b/1} 4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5,$

$5 \xrightarrow{b/1} 4 \xrightarrow{b/0} 2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5,\ 5 \xrightarrow{b/1} 4 \xrightarrow{b/0} 2$

$3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2 \xrightarrow{b/0} 1,\ 3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5,$

$3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5,\ 3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2$

$1 \xrightarrow{b/0} 5 \xrightarrow{a/0} 3,\ 1 \xrightarrow{b/0} 5 \xrightarrow{b/1} 4$

$2 \xrightarrow{c/1} 5 \xrightarrow{a/0} 3,\ 2 \xrightarrow{c/1} 5 \xrightarrow{b/1} 4$

corresponding to the transitions $2 \xrightarrow{b/0} 1$, $4 \xrightarrow{b/0} 2$, $5 \xrightarrow{a/0} 3$, $5 \xrightarrow{b/1} 4$, $3 \xrightarrow{a/1} 4$, $1 \xrightarrow{b/0} 5$ and $2 \xrightarrow{c/1} 5$ respectively.

Noting that the testing graph corresponding to the specification machine has no hiding structure [MP92], these test segments may be combined using overlapping to obtain the following maximal fault coverage test sequence:

$$2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5 \xrightarrow{a/0} 3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5 \xrightarrow{b/1} 4 \xrightarrow{b/0} 2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5 \xrightarrow{b/1} 4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5 \xrightarrow{a/0} 3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5$$

Note that the minimal length test sequence cannot detect a combination of a head state fault and a tail state fault. For example, a tail state fault $4 \rightarrow 2$ to $4 \rightarrow 3$ and a head state fault $2 \rightarrow 5$ to $3 \rightarrow 5$ are enough to go undetected. Yet, the maximal fault coverage test sequence detects this fault. Thus the minimal length test sequence does not provide maximal coverage.

Now consider the maximal fault coverage test sequence further. Notice that node 5 has two in-transitions $1 \rightarrow 5$ and $2 \rightarrow 5$ and two out-transitions $5 \rightarrow 3$ and $5 \rightarrow 4$. The maximal coverage test sequence includes all four possible combinations of traversing these transitions:

$1 \rightarrow 5 \rightarrow 3$, $1 \rightarrow 5 \rightarrow 4$, $2 \rightarrow 5 \rightarrow 3$, and $2 \rightarrow 5 \rightarrow 4$

This provides better coverage, but it is claimed that all four of these test segments are not needed to ensure maximal coverage. Just three of them are enough. An informal argument supporting this claim is as follows. Consider the three segments $1 \rightarrow 5 \rightarrow 3$, $2 \rightarrow 5 \rightarrow 3$, $2 \rightarrow 5 \rightarrow 4$. Since in-transition $2 \rightarrow 5$ is followed by both the out-transitions $5 \rightarrow 3$ and $5 \rightarrow 4$, any head state alteration of these out-transitions will be detected by $2 \rightarrow 5 \rightarrow 3$ and $2 \rightarrow 5 \rightarrow 4$. The only possibility of hiding arises when both $5 \rightarrow 3$ and $5 \rightarrow 4$ are altered to $X \rightarrow 3$ and $X \rightarrow 4$ and $2 \rightarrow 5$ is altered to $2 \rightarrow X$ for some state X. In that case, however, $1 \rightarrow 5 \rightarrow 3$ will detect the fault. Thus the segment $1 \rightarrow 5 \rightarrow 4$ is not needed. The reduced length test sequence with maximal fault coverage is developed later after the algorithm is explained.

What follows next is a systematic approach for detecting and eliminating this kind of *redundancy* to shorten the test sequence while not reducing its fault coverage.

3.1 A Clarification

The fault detecting capability of test segments can be analyzed based on the assumption that the correct

starting state X is reached each time a test segment starting with X is exercised [MP92]. Strictly speaking, that assumption is not valid because the correct starting state for each segment cannot be guaranteed in the implementation during the process of testing.

While analyzing the fault detecting capability of a test sequence, the effect of the implementation not being in the correct starting state needs to be taken into account.

In this paper, the fault detecting capability of test segments is analyzed (i.e., the assumption about the correct starting state is made) until theorem 2. In fact, the objective is to generate a minimum number of test segments which can guarantee maximal coverage under the assumption of correct starting state.

Once these non-redundant test segments are obtained, they are connected to generate a test sequence. In order to guarantee maximal coverage for such a test sequence, the *ordering constraints* [MP92] need to be imposed on the connection of the test segments if hiding structures [MP92] are present in the specification.

4 An approach to minimizing length while maximizing fault coverage

We know from [P92] and [MP92] that multiple tail state faults need certain structures to be present for hiding one another. However, a combination of head state faults and tail state faults does not need special structures to hide one another. Thus in the example of the previous section, a tail state fault $4 \rightarrow 2$ to $4 \rightarrow 3$ and a head state fault $2 \rightarrow 5$ to $3 \rightarrow 5$ hid each other in the minimal length test sequence for the implementation.

These considerations lead us to concentrate on one node at a time, along with its in-transitions and out-transitions, and generate test segments by following in-transitions with out-transitions in such a way that no combination of tail state faults and head state faults can hide one another. Since the aim is to generate a minimal length test sequence, it must be insured that *redundant* test segments are not generated.

In trying to maximize fault coverage locally (at a node), it is possible to become overly conservative and add some redundant combinations. Thus, the global picture also must be kept in mind while minimizing length.

Recall that a test segment is formed by following a transition with a PDS of the ending state of the transition. Every out-transition of a node is some PDS (except in a very special case) of the state represented by the node. However, the out-transitions of a node belong to one of the following two categories:

(1) at least one out-transition of a node is a UIO sequence of the state represented by the node.

(2) none of the out-transitions of a node is a UIO sequence.

Case (1) is considered first and some results related to it are proved. Then case (2) is considered and treated in a similar manner.

An important point needs to be made here for further clarification. In the subsequent sections of the paper, we show how non-redundant test segments (a transition followed by a PDS and not a UIO of the ending state of the transition) can be generated. Given these segments, the reduced length test sequence is simply a concatenation of these using overlap as much as possible and transfer sequences if necessary. The introduction of UIO's can be viewed as simply a convenience for proof purposes, even though they may turn up in the test sequence because of relationships between PDS's and UIO's.

4.1 Lemmas and Theorems

Theorem 1 *For an arbitrary FSM specification, if a node i has at least one out-transition which is a UIO sequence, then n+m-1 segments are necessary and sufficient to guarantee detection of any tail state fault of the transitions ending in, and/or head state fault of the transitions starting from, node i where*

n = in-degree of node i and m = out-degree of node i.

Proof: Due to space limitations, the proofs are omitted in this paper. However, they can be found in [MP93(a)]. □

The above theorem takes care of the situation in which at least one out-transition of a node is a UIO sequence of the state represented by the node. Now, the case in which none of the out-transitions is a UIO sequence is considered. There are two cases:

(1) No UIO sequence for the state in question exists.

(2) A UIO sequence for the state in question exists, but no out-transition is a UIO sequence by itself.

First some results for case (1) are stated and then case (2) is considered.

Lemma 1 *If the state represented by node i has no UIO sequence, then at least 2n segments are necessary to guarantee detection of any tail state fault related to the transitions ending in node i where*

n = in-degree of node i.

Lemma 2 *If the state represented by node i has no UIO sequence, then 2n+m-2 segments are sufficient to guarantee detection of any tail state fault of the transitions ending in, and/or head state fault of the transitions starting from, node i where*

n = in-degree of node i and m = out-degree of node i.

A few definitions are in order here.

Definition 4.1: An out-transition set OT_j^i corresponding to an in-transition $X_j \rightarrow i$ of node i is defined as a set of out-transitions of a node i which follow an in-transition $X_j \rightarrow i$ in test segments. If there are m transitions between states X_j and i with different i/o labels, the out-transition set corresponding to the *kth* transition between X_j and i will be denoted by $OT_j^{i,k}$. If there is only one transition between X_j and i, however, the notation OT_j^i will be used. The superscript i will be omitted when it is obvious from the context that node i is being referred to.

Definition 4.2: An out-transition group OTG^i of a node i is defined as a set of out-transitions which *converge* with the out-transitions of a different node. In general, the *kth.* group is referred to as OTG_k^i. Thus if there are two OTG^i's, they are referred to as OTG_1^i and OTG_2^i. The superscript i will be omitted when it is obvious from the context that node i is being referred to.

Lemma 3 *If the state represented by node i has no UIO sequence, then a sufficient condition for the guaranteed detection of any tail state fault of the transitions ending in, and/or head state fault of the transitions starting from, node i is as follows:*

(1) $OT_{(j-1)mod(n-1)} \cap OT_j \neq OT_j \cap OT_{(j+1)mod(n-1)} \ \forall \ 0 \leq j \leq (n-1)$
(2) $OT_{(j-1)mod(n-1)} \cap OT_j \neq \Phi$ $\forall\ 1 \leq j \leq (n-1)$ *and*
(3) $\forall\ 0 \leq j \leq (n-1)$, $\exists\ k \neq l$ *such that* $ot_k, ot_l \in OT_j$, $ot_k \in OTG_k$, $ot_l \in OTG_l$
where n = number of in-transitions of node i,

provided each in-transition and out-transition is a part of some test segment.

From lemma 1, it follows that if a UIO sequence does not exist for a state i, then $2n$ segments are needed for the detection of tail state faults. If those $2n$ segments can be chosen so as to satisfy the conditions of lemma 3, then they will be able to detect any tail state fault and/or head state fault. Thus, it can be concluded that if a UIO sequence does not exist for a state represented by node i, then the number of segments *nseg* needed to guarantee detection of any tail state and/or head state fault is given by:

$$2n \ \leq \ nseg \ \leq \ 2n + m - 2$$

Now, let us consider the case in which a UIO sequence exists for the state under consideration.

Theorem 2 *If the state represented by node i has at least one UIO sequence but none of the out-transitions is a UIO sequence and the specification machine does not have any total hiding structure [MP92], then n+m-1 segments are necessary and sufficient to guarantee detection of any tail state fault of a transition ending in, and/or head state fault of the transitions starting from, node i where*

n = in-degree of node i and m = out-degree of node i.

5 The Algorithm

Based on the lemmas and theorems of the previous section, the following algorithm is proposed for the generation of a near minimal length test sequence with maximal fault coverage.

Algorithm Main

Given: A specification FSM.

Step 1: Examine the structure of the specification FSM to detect the presence of *total hiding structures* in the testing graph.

Step 2:

For each node $i \in N$
if at least one out-transition is a UIO sequence then
Use Algorithm UIO_Out_Trans
else if there is no UIO sequence then
Use Algorithm No_UIO
else
Use Algorithm UIO

Step 3: Combine the test segments. If *total hiding structure(s)* is (are) present, then the ordering constraint [MP92] must be satisfied when combining the test segments.

Algorithm UIO_Out_Trans

Given: A node with at least one out-transition which is a UIO sequence of the state represented by the node.

Step 1:

for each in-transition $X_j \rightarrow i$

Generate a test segment $X_j \rightarrow i \rightarrow Y_k$ where $i \rightarrow Y_k$ is a out-transition which is a UIO sequence.

for each out-transition $i \rightarrow Y_j$

if $j \neq k$ then

Generate $X_l \rightarrow i \rightarrow Y_j$ where $X_l \rightarrow i$ is an in-transition of node i.

Algorithm No_UIO

Given: A node with p groups of out-transitions such that the out-transition of each group *converge* with the out-transitions of a different node i'. Also $p \geq 2$.

Step 1:

for each in-transition $X_j \rightarrow i$

Generate a pair of test segments $X_j \rightarrow i \rightarrow Y_k$ and $X_j \rightarrow i \rightarrow Y_k'$ such that $i \rightarrow Y_k$ and $i \rightarrow Y_k'$ belong to two different groups.

for each out-transition $i \rightarrow Y_j$

if $i \rightarrow Y_j$ has not been covered then

Generate $X_1 \rightarrow i \rightarrow Y_j$

Algorithm UIO

Given: A node with no out-transition which is a UIO sequence but a UIO sequence exists for the node.

Step 1:

for each in-transition $X_j \rightarrow i$

if there is a *total hiding structure* in the testing graph [MP92] and PX_j[2] $\rightarrow Qi$ or $X_jP \rightarrow iQ$ is a part of it (where P and Q are two arbitrary states)then

Generate a test segment $X_j \rightarrow i \rightarrow Y_k$ where $i \rightarrow Y_k$ is an exit from the hiding structure. (Note that $i \rightarrow Y_k$ need not be a UIO sequence).

else

Generate a test segment $X_j \rightarrow i \rightarrow Y_k$ where $i \rightarrow Y_k$ is an out-transition along the UIO sequence of the state represented by node i.

for each out-transition $i \rightarrow Y_k$

if $i \rightarrow Y_k$ has not been covered then

Generate $X_j \rightarrow i \rightarrow Y_k$ for some $1 \leq j \leq n$.

Example 1 (revisited): The new algorithm proposed in this section will be used to work out Example 1. First of all, there is no total hiding structure in the testing graph.

Consider node 1. It has one out-transition $1 \xrightarrow{b/0} 5$ but it is not a UIO sequence of state 1. State 1, however, has a UIO sequence. So "Algorithm UIO" can be used to obtain test segment $TS_1 : 2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5$.

Consider node 2. It has out-transitions $2 \xrightarrow{b/0} 1$ and $2 \xrightarrow{c/1} 5$ and $2 \xrightarrow{c/1} 5$ is a UIO sequence of state 2. Therefore, "Algorithm UIO_Out_Trans" can be used to obtain test segments $TS_2 : 4 \xrightarrow{b/0} 2 \xrightarrow{b/0} 1$ and $TS_3 : 4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5$.

Consider node 3. It has one out-transition $3 \xrightarrow{a/1} 4$ and it is a UIO sequence of state 3. Therefore, "Algorithm UIO_Out_Trans" can be used to obtain test segment $TS_4 : 5 \xrightarrow{a/0} 3 \xrightarrow{a/1} 4$

Consider node 4. It has one out-transition $4 \xrightarrow{b/0} 2$ but it is not a UIO sequence of state 4. Using "Algorithm UIO", test segments $TS_5 : 3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2$ and $TS_6 : 5 \xrightarrow{b/1} 4 \xrightarrow{b/0} 2$ are obtained.

Consider node 5. It has two out-transitions $5 \xrightarrow{a/0} 3$ and $5 \xrightarrow{b/1} 4$ and both of them are UIO sequences of state 5. Therefore, we can use "Algorithm UIO_Out_Trans". First of all, each of the two in-transitions $1 \xrightarrow{b/0} 5$ and $2 \xrightarrow{c/1} 5$ are followed with a specific out-transition $5 \xrightarrow{a/0} 3$ to obtain test segments: $TS_7 : 1 \xrightarrow{b/0} 5 \xrightarrow{a/0} 3$ and $TS_8 : 2 \xrightarrow{c/1} 5 \xrightarrow{a/0} 3$. Then $TS_9 : 2 \xrightarrow{c/1} 5 \xrightarrow{b/1} 4$ is generated to cover out-transition $5 \xrightarrow{b/1} 4$.

Since the specification machine does not have any *hiding structure* in its testing graph, the test segments can be connected in any order. Thus, these test segments may be combined using overlapping to obtain the following test sequence with maximal fault coverage:

$2 \xrightarrow{b/0} 1 \xrightarrow{b/0} 5 \xrightarrow{a/0} 3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5 \xrightarrow{b/1} 4 \xrightarrow{b/0} 2 \xrightarrow{c/1} 5 \xrightarrow{a/0} 3 \xrightarrow{a/1} 4 \xrightarrow{b/0} 2 \xrightarrow{b/0} 1$

[2]The notation PX_j refers to a node of the testing graph. A node of the testing graph is a pair of states such that they both have a transition defined on the same i/o label. In PX_j, P is an arbitrary state and X_j is the state corresponding to the transition $X_j \rightarrow i$.

Note that the length of the minimal length test sequence for the specification FSM of Fig.1 is 9 and the length of the maximal fault coverage test sequence for the same example is 18. These values are obtained from section 3 where example 1 was worked out. The new algorithm gives a test sequence of length 13 with maximal fault coverage. Thus a significant amount of *redundancy* has been removed without sacrificing fault coverage.

Although, it cannot be claimed that this is a minimal length test sequence with maximal fault coverage, it can, however, be claimed that this is a test sequence which uses a minimal number of test segments.

6 Conclusion

6.1 Comparison with Previous Work:

[SSLS91] has proposed a technique to reduce the number of test segments leading to the reduction in overall length of the test sequence. It also claimed 100% fault coverage with respect to the fault categories of [SL88] based on *simulation* studies. There are no theoretical results to back this claim of guaranteed fault coverage. In contrast to their work, our claim of *guaranteed maximal fault coverage* is based on *theoretical analysis* and proofs [MP92]. In addition to that, the definition of maximal fault coverage is more general than theirs because the results of this work are not limited to the fault categories of [SL88].

[SSL92] has done some work in the direction of reducing length while maintaining a *very high* fault coverage. They use *adaptive* UIO sequences to reduce the number of tests. However, they *do not guarantee maximal fault coverage*. Their results in the area of fault coverage are based on the limited number of categories introduced in [SD88] and [SL88] and are *not for any arbitrary number of faults*.

[FSL92] has also done a similar work, the objective being reducing the length while maintaining maximal fault coverage. The approach in that paper is different from the work in this paper in the sense that they start with a given test sequence, generate all the machines that are not detected by the test sequence, add additional subsequences to the test sequence to discard the machines that do not contain the specification and eventually generate a test sequence which can detect *all implementations with erroneous behavior*. As far as we know, no comparison of these two approaches as per length of test sequences has been made.

6.2 Contribution of the paper

The main contributions of this paper may be summarized as follows:

1) This paper gives a direction to approach a fundamental problem in the generation of test sequences with near minimal length and maximal fault coverage.

2) This paper provides a rather simple mechanism to generate *non-redundant* test segments which usually involves lengthy and complex computations as in [FSL92].

As regards future research, further understanding of structural properties and fault coverage could be looked into, and simplified test sequence generation approaches might be found for special situations.

More work is needed in the area of reducing length while maximizing fault coverage. The final objective is to generate a minimal length test sequence with maximal fault coverage.

In addition, the trade-off between length and fault coverage must be studied in more detail so that questions related to the trade off can be answered *quantitatively* as opposed to *qualitatively*.

References

[ADLU88] A. Aho, A. T. Dahbura, D. Lee, and M. Umit Uyar, "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.

[ADLU91] A. Aho, A. T. Dahbura, D. Lee, and M. Umit Uyar, "An Optimization Technique For Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours," IEEE Transactions on Communication, vol. COM-39, no. 11, 1991.

[C79] T. Chow, "Testing Software Design Modelled by Finite State Machines," IEEE Transactions on Software Engineering, vol. SE-4, pp. 178-187, March 1978.

[CCK90] M. S. Chen, Y. Choi, A. Kershenbaum, "Approaches Utilizing Segment Overlap To Minimize Test Sequences," Tenth International IFIP WG 6.1 Symposium on Protocol Specification, Testing and Verification.

[CS90] Anthony Chung and Deepinder Sidhu, "Fault Coverage of Probabilistic Test Sequences," Conference Proceedings of the 3rd. International Workshop on Protocol Test Systems, Mclean, Virginia, Oct.30 - Nov.1, 1990.

[FSL92] C. Feng, Y. N. Shen and F. Lombardi, "On the Detectability of Test Sequences for Protocol Verification and Validation," private communication.

[G70] G. Gonenc, "A method for the design of fault detection experiments," IEEE Transactions on Computers, vol. C-19, pp. 551-558, June 1970.

[K78] Z. Kohavi, Switching and Finite Automata Theory. New Yo rk: McGraw-Hill, 1978.

[LS91] F. Lombardi and Y. N. Shen, "Estimation and Improvement of Fault Coverage of Conformance Testing by UIO Sequences," IEEE Transactions on Communication (Accepted).

[MP91] Raymond E. Miller and Sanjoy Paul, "Generating Minimal Length Test Sequences for Conformance Testing of Communication Protocols," IEEE INFOCOM '91.

[MP92] Raymond E. Miller and Sanjoy Paul, "Structural Analysis of a Protocol Specification and Generation of a Test Sequence with Maximal Fault Coverage for Conformance Testing of Communication Protocols", CESDIS Technical Report TR-92-89, submitted for publication.

[MP93(a)] Raymond E. Miller and Sanjoy Paul, "Generating Maximal Fault Coverage Conformance Test Sequences of Reduced Length for Communication Protocols", CESDIS Technical Report TR-93-97.

[MP93] Raymond E. Miller and Sanjoy Paul, "On the Generation of Minimal Length Conformance Tests for Communication Protocols," IEEE/ACM Transactions on Networking, February 1993.

[NT81] S. Naito and M. Tsunoyama, "Fault Detection of Sequential Machines by Transition Tours," Proc. IEEE Fault Tolerant Computing Conference, 1981.

[P92] Sanjoy Paul, "A Structural Analysis Approach for Designing Efficient and Effective Algorithms for Conformance Testing of Communication Protocols", Ph.D Dissertation, Department of Electrical Engineering, University of Maryland, College Park, MD 20742.

[SB82] B. Sarikaya and G. V. Bochmann, "Some experience with test sequence generation for protocols," Protocol Specification, Testing and Verification, C. Sunshine(ed.), North Holland Publishing Company, IFIP 1982.

[SD88] K. Sabnani and A. Dahbura, "A Protocol Test Generation Procedure and its Fault Coverage," Computer Networks and ISDN System 15, 1988, pp. 285 - 297.

[SL88] D. Sidhu and T. Leung, "Fault Coverage of Protocol Test Methods," Proc. IEEE INFOCOM '88, pp. 80 - 85, March 1988.

[SLD89] Y. N. Shen, F. Lombardi and A. T. Dahbura, "Protocol Conformance Testing Using Multiple UIO Sequences," Proc. Ninth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification, 1989.

[SSL92] X. Sun, Y. N. Shen and F. Lombardi, "Conformance Testing of Protocols by Adaptive UIO Sequences," private communication.

[SSLS91] X. Sun, Y. N. Shen, F. Lombardi and D. Sciuto, "Protocol Conformance Testing by Discriminating UIO Sequences," Proceedings of 11th. IFIP International Symposium on Protocol Specification, Testing and Verification, Stockholm, June 1991.

[YU90] Bo Yang and Hasan Ural, "Protocol Conformance Test Generation using Multiple UIO Sequences with Overlapping," Computer Communication Review, Volume 20, Number 4, September 1990.