ELSEVIER

Computer Communications 22 (1999) 1447–1456

computer communications

A complete test sequence using cyclic sequence for conformance testing

D. Nyang*, S.Y. Lim, J. Song

Department of Computer Science, Yonsei University, SeodaemunGu ShinchonDong, 120-749 Seoul, South Korea

Received 11 September 1998; received in revised form 1 June 1999; accepted 1 June 1999

Abstract

We present a problem of commonly used characterization sequences (CS) for the protocol conformance testing and propose a new test sequence to resolve the problem. The proposed test sequence can decide whether a fault arises in the edge being tested or in edges composing the CS of the tested edge. Additionally, its fault coverage is much wider than that of other test sequence generation methods. To achieve the goal, we introduce the *k-strong* FSM and the extended UIO(EUIO), and show that it can be constructed from any CS. We also illustrate our technique on Q.2931, which is the call establishing protocol in B-ISDN. To increase the probability that a given FSM might be *k-strong*, we introduce a new test sequence generation scheme using cyclic input characterization sequence (CICS). Also, we present a technique to reduce the length of the test sequence satisfying the completeness.

Keywords: Protocol testing; FSM; Characterization sequence; Completeness criterion; Cyclic sequence

1. Introduction

Conformance testing of communication protocol implementations is an area of considerable research as testing can help guarantee correct operation of the protocol. In the literature, an implementation is regarded as a black box, and its conformity is verified by observing output behaviors for input sequences.

Various automatic test sequence generation methods have been proposed such as UIO-, *W*-, DS-methods [1]. With these methods, one can test only one transition, hence a technique to constitute an overall test sequence that tests all transitions for a specification FSM is required. One is to combine the characterization sequence using reset input and the shortest path to the state to be tested [1,2]. Another is a transition tour, which is regarded to have worse fault capability [3].

Studies in the test sequence generation method have been concentrated on the fault coverage and the length of a generated test sequence [2,4–8]. For the wide fault coverage, testing methods such as fault tolerant UIO methods and pair wise distinguishing sequence have been proposed [6–9]. However, test sequences with the wide fault coverage still cannot guarantee the correctness of its decision. In this paper, we show that commonly used test sequences such as UIO and DS fail to test a correct state transition, but our scheme guarantees the completeness of the verdict of a test sequence.

Concerning the incorrect behavior of test sequences, we define a new problem instance and propose the input test sequence generating scheme to resolve the problem. To achieve the goal, *k-strong* FSM is defined. A characterization sequence that makes an FSM to be *k-strong* is proposed using extended UIOs, and its fault detection capability is examined. For the construction of an extended UIO, multiple UIO sequences are adopted [3]. We illustrate our technique on Q.2931 that is the call establishing protocol in B-ISDN. To increase the probability that a given FSM might be *k-strong*, we introduce a new test sequence generating scheme using cyclic input characterization sequence (CICS). We show that the length of the test sequence satisfying the completeness can be reduced using the proposed scheme. Conformance test procedure with the proposed test sequence reduces a tester's job, as the test sequence guarantees the validity of its verdict.

2. Background

A protocol specification is usually described as a deterministic FMS, which has a finite set of inputs $I = \{i_1, i_2, \ldots, i_k\}$, a finite set of outputs $O = \{o_1, o_2, \ldots, o_m\}$, and that of states $S = \{s_1, s_2, \ldots, s_n\}$. The next state function *NS* and the output function *Z* are given by $NS: S \times I \rightarrow S$ and $Z: S \times I \rightarrow O$, respectively. An FSM of protocol specification is represented by a directed graph $G = (V, E)$, where V

* Corresponding author. Tel.: + 82-2365-7966; fax: + 82-2365-2579.
E-mail address: nyang@emerald.yonsei.ac.kr (D. Nyang)

0140-3664/99/$ - see front matter
PII: S0140-3664(99)00102-4

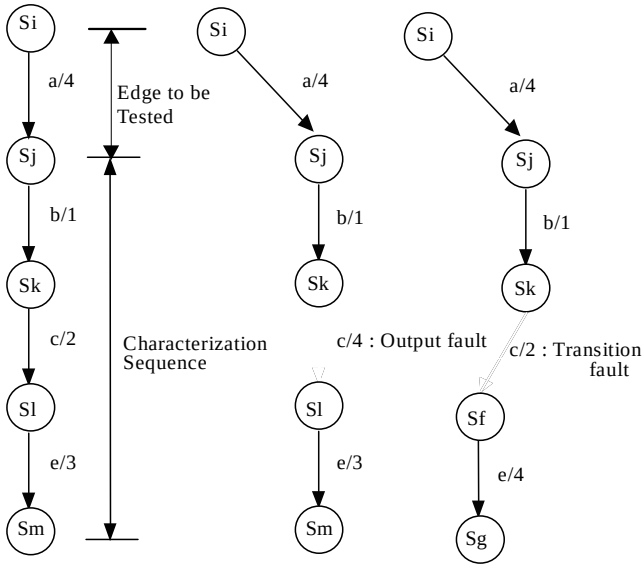

Fig. 1. Incorrect behavior of commonly used test sequences (a) without fault, (b) with output fault and (c) with transition fault.

is a set of vertices $\{v_1, v_2, \ldots, v_n\}$ corresponding to *S* and *E* is a set of edges $\{e_1, e_2, \ldots, e_m\}$ which is determined by *NS* and *Z*. The FSM is considered to be deterministic in the sense that no two edges with the same input go out from the same state. In this paper, we denote the input output pair as "i_n/o_n". An implementation of a certain protocol is assumed to be a black box and its internal states are regarded neither controllable nor observable. The internal structure of the implementation under test (IUT) can be inferred only by observing output behaviors for a certain input sequence. Conformance testing for an IUT is to examine whether the state transition and its output is correctly implemented or not. To be sure that the IUT is in a state, an input sequence is applied to the IUT and its output sequence is compared to the expected output sequence. We call the input sequence and the expected output sequence pair a *characterization sequence*, and denote a characterization sequence for a state *Sj* as CS(*Sj*).

Most famous characterization sequences include unique input/output (UIO) sequence, distinguishing sequences (DS), and *W* set. The idea behind those characterization sequences is to identify a state by applying an input sequence set whose output sequences are unique for all other states in the FSM. An UIO sequence for a state of an FSM is an input/output behavior that is not exhibited by any other state of the FSM. An input sequence *x* is said to be a DS of an FSM if the output sequence produced by the FSM in response to *x* is different for each starting state. DS can be computed by constructing the distinguishing tree inductively on tree levels. A characterization set *W* for an FSM is a set consisting of input string $a_1, \ldots, a_k$ such that the last output symbols observed from the application of these strings are different at each state of the FSM. Details on these characterization sequences can be found in Ref. [1].

3. A new problem instance

In this section, we illustrate a problem that arises in existing test sequences and define a criterion to deviate the problems, called completeness criterion. To satisfy the completeness criterion, we introduce a new concept of *k-strong* FSM and the condition of a test sequence to make an FSM *k-strong*. Also, the connection between the *k-strong* FSM and the completeness criterion is explained.

3.1. Completeness criterion

In this section, we describe the situation where the commonly used characterization sequences such as UIO and DS fail to test a correct state transition. Fig. 1 shows that the transition (*Si*, *Sj*) is considered faulty due to the output fault (Fig. 1(b)) or the transition fault (Fig. 1(c)) in CS for *Sj* even though implemented correctly. That is, an output fault or a transition fault in the characterization sequence can induce a wrong verdict. Thus, consideration on the completeness criterion should be taken into a generation of test sequence.

Definition 1 (Completeness criterion). Assume that an edge to be tested is conformed to the specification. When a testing sequence makes a verdict that the tested edge in an implementation under test is conformed to the specification FSM, then the testing sequence is said to satisfy the *completeness criterion*.

Test sequence that satisfies the completeness criterion has several advantages. First of all, it guarantees the validity of the verdict when the final verdict of the test sequence is FAIL. Second, a test sequence designed carefully to satisfy the completeness criterion has fault tolerant property in the sense that a few faults in the characterization sequence do not effect the verdict of the test sequence. Third, using the completeness property, one can substantially reduce the length of the overall test sequence, which will be explained in Section 6.

As illustrated in Fig. 1, wrong verdicts by a test sequence may be caused from output faults or transition faults, or both. In the present paper, we solve the completeness criterion problem caused only from output faults.

3.2. A test sequence considering the problem

Here, we briefly explain the test sequence that can verify the correct state transition. A key observation is that a characterization sequence whose input/output behavior is totally different from other states' can discriminate whether the fault arises in the segment being tested or in the characterization sequence. Under a few output errors in

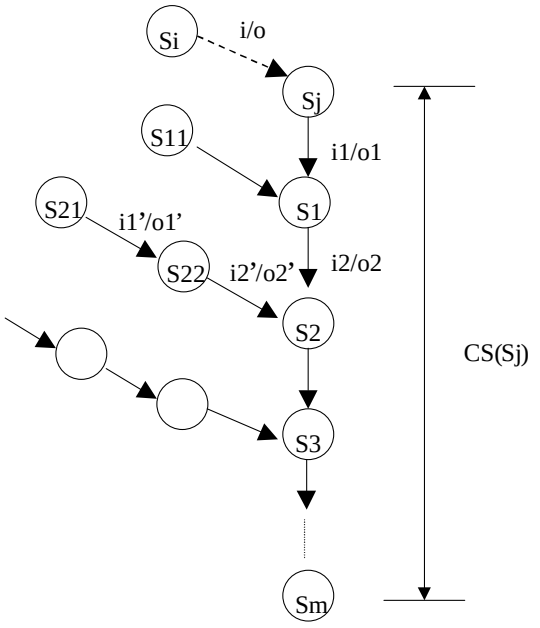

Fig. 2. Illustration of converging walks.

the characterization sequence, existing characterization sequences cannot verdict correctly, whereas the characterization sequence that satisfies the completeness criterion can say that the edge being tested is correctly implemented. Now, we describe the test sequence with more detail.

Definition 2 (Degree of difference). *Degree of Difference* between an i/o sequence A and an i/o sequence B, DoD(A,B) is defined as the number of output differences between A and B, where A and B have the same input sequences. If A and B have different input sequences, then DoD(A, B) = infinite. In counting the number of differences, previously visited transitions in loops are not counted, as they already contribute to DoD(A,B).

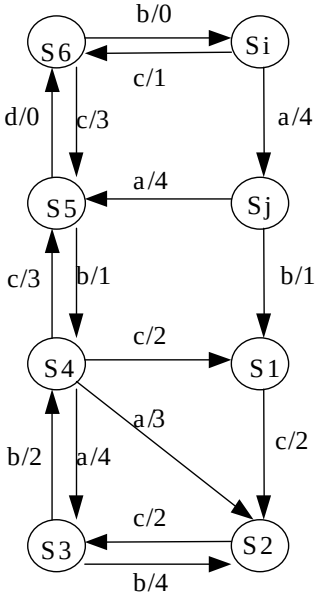

Fig. 3. Example of *k-strong* FSM.

Definition 3 (*k-strong* characterization sequence). A CS(*Si*) for a state *Si* is called *k-strong* if it satisfies

$$2k + 1 \Leftarrow \min(\mathrm{DoD}(\mathrm{CS}(Si), W)) < 2k + 3$$

for all i/o sequences *W* in the FSM, which have the same input sequence as CS(*Si*).

Definition 4 (*k-strong* FSM)). An FSM is called *k-strong* if there exists CS(*Si*) such that CS(*Si*) is more than *k-strong* for every state *Si* in the FSM in the FSM.

Intuitively, at most *k* output faults in verifying sequence do not affect the fault detection capability of the test sequence in testing *k-strong* FSM. This is because at most *k* output faults do not make a CS look like the other CS in *k-strong* FSM.

Now, we consider the connection between a *k-strong* FSM and an FSM having a test sequence satisfying the completeness criterion. There are two reasons that a CS makes a wrong verdict for the correctly implemented edge, as we have seen in Fig. 1. Ignoring the wrong verdict caused by the transition fault in a CS, the only reason that a CS makes a wrong verdict is the output fault in a CS. By definition, the *k-strong* FSM is immune against less than *k* output faults in a CS. Thus, a *k-strong* FSM is equivalent to an FSM having a test sequence satisfying the completeness criterion, if there are less than *k* output faults and no transition faults in a CS. In the sequel, if we provide a CS that makes an FSM *k-strong*, we can guarantee the completeness of the verdict. In the subsequent sections, we provide a CS generation method that makes an FSM *k-strong*.

3.3. Proper CS selection criterion for k-strong FSM

W-set, UIO, DS, and their variants can be used as the characterization sequence satisfying Definition 3. For each characterization sequence, *k-strong* FSM may exist or not.

Definition 5 (Converging walks). *Converging walks*, *W*1 and *W*2 are those i/o sequences that converge into the same state with the same input sequence without consideration of the output sequence. Assume that the edge (*Si*, *Sj*) is to be tested in Fig. 2. $(Sj, S1, S2; \mathrm{i1/o1i2/o2})$ and $(S21, S22, S2; \mathrm{i1'/o1'i2'/o2'})$ converge into the same state *S*2, if $\mathrm{i1} = \mathrm{i1'}$ and $\mathrm{i2} = \mathrm{i2'}$. Then they are called *converging walks* and state *S*2 is called a *convergent state*. Also, the state *S*1 is called a *converging state* (Fig. 3).

For a CS for a state *Sj* to be *k-strong*, following criterion should be satisfied.

For $n = 1, 2, 3, \ldots, 2k$, *n*-Head(CS(*Sj*)) does not have any converging walk, where *n*-Head(X) is a first *n* i/o sequence of X. For example, 3-Head($(a/1, b/2, a/0, c/3, b/2)$) is $(a/1, b/2, a/0)$.

The criterion explains that the *k-strong* FSM does not have any converging walks whose lengths are less than or equal to $2k$. Because there are less than $2k$ output results when k output faults occur, the number of output faults overwhelm the number of correctly implemented edges on the CS and we can neither know what the current state is, nor decide the correctness of the edge being tested.

The criterion, however, is not a sufficient condition, but a necessary condition. Also, Definition 3 says that CS that makes an FSM *k-strong* must have at least $2k + 1$ different outputs. If the CS has $2k + 3$ output differences with all other converging walks, it makes an FSM $(k + 1)$-*strong*. As there is more than $2k + 1$ output results, there should be a CS that makes it different from all other walks. Thus, we know both what the current state is and where the fault occurs, even when k faults occur.

In the information theoretical sense, this criterion is similar to the error correction capability of Hamming code. Proof of the above criterion can be constructed in a similar way of the error correction capability of the Hamming code [10].

Testing with *k-strong* FSM is performed as follows: The edge (Si, Sj) is correctly implemented when

$$\mathrm{DoD}(\mathrm{CS}(Sj),\ \text{Expected Sequence}) \Leftarrow k,$$

where CS(Sj) is the characterization sequence for a state Sj.

The above passing criterion for an edge is the generalization of all other existing test methods, where k is equal to "0". When k is equal to "0", the FSM under testing is 0-strong and cannot verdict exactly whether the tested edge is correct or not. However, when k is greater than 0, a tester can decide whether the tested edge is correctly implemented or not even when there are k output faults in verifying sequence, or the testing with *k-strong* FSM satisfies the completeness criterion explained in Definition 1.

Like most other testing methods, the overall test sequence is composed of

$$(\mathrm{ri})@(\text{Shortest Path to } Si)@(\mathrm{i/o}@\mathrm{CS}(Sj), \qquad (1)$$

where ri is a reset input and @ means concatenation. Instead of (1), test sequences using graph touring such as Rural Chinese Postman Tour (RCPT) can be used.

4. Construction of *k-strong* FSM with extended UIO

This section shows how to construct the extended UIO sequence using the plain UIO sequence, make the FSM *k-strong*, and test an IUT with the EUIO. With a simple FSM, we explain the EUIO construction and the testing procedure. Finally, EUIO for the Q.2931 that is the call/connection establishment protocol in B-ISDN is shown, and its properties are analyzed.

4.1. Extended UIO construction algorithm

As explained in the previous section, characterization sequences that make an FSM *k-strong* may be constructed from *W*-set, UIO, DS, and their variants. In this section, we propose a characterization sequence using UIO, which makes an FSM *k-strong*.

Generally, minimal UIO sequence is applied in testing an FSM, but in most cases it cannot make an FSM *k-strong*. Thus we need to extend minimal UIO so that it may make the FSM *k-strong*. For this purpose, we suggest the new testing sequences called EUIO.

Now, we present an EUIO construction algorithm. The algorithm is basically based on the breadth first search (BFS), and during the search they check the completeness criterion. Though the EUIO construction algorithm is based on BFS, it is not a brute force search as it uses heuristics to find an EUIO sequence. To examine converging walks for a state, it does not examine all paths that come into the state, but only the paths that have the same input sequence and the same input length. Thus, during the search, pruning does occur.

```
Program main
Begin
  if there is a UI sequence then
  EUIO = UI
  Print(UI), exit(Succeed!)
  else
  L: for each minimal UIO
  PreUIO = minimal UIO
  if |PreUIO| < 2k then
  (* Extend PreUIO to the length 2k, considering convergence. *)
  M: for each padding sequence pad which makes
  |PreUIO @ pad| = = 2k
  PreUIO = PreUIO@pad =
  (S1, S2, ..., Sn; i1/o1, i2/o2, ...in-1/on-1)
  preuioflag = TRUE
  for each Sn; from n = 1 to 2k
  for each i/o sequence W whose tail state is Sn
  if CheckPreUIO(W,PreUIO,n) = = FALSE then
  preuioflag = FALSE
  goto M
  endif
  endfor
  endfor
  if preuioflag = = TRUE then
  break;
  endif
  endfor
```

```
if preuioflag = = FALSE then
goto L
endif
for each outgoing edge e = i/o from tail(PreUIO)
enqueue(PreUIO @ e)
endfor
while Queue is not Empty do
P = dequeue( )
dodflag = TRUE
for each walk W in the FSM whose input sequence is the same as that of P
if DoD(P, W) < 2k + 1 then
dodflag = FALSE
break
endif
endfor
if dodflag = = FALSE then
if P is not composed of all edges in the FSM then
for each outgoing edge m = i/o from tail(P)
enqueue(P @ m)
endfor
else
exit(this machine is not k-strong)
endif
else
Print(P), exit(Succeed!)
endif
endwhile
else (* |PreUIO| > = 2k *)
(* Do the same procedure as that when |PreUIO| < 2k, except padding. *)
endif
endfor
endif
End.
```

If there exists a UI (Unique Input) sequence among UIOs of the state, then we can use it as an EUIO. If not, we choose a minimal UIO and pad some tail sequence to make the length of the padded UIO greater than or equal to $2k$. After padding the UIO, the algorithm checks whether the padded sequence satisfies the completeness criterion. If the padded sequence does not satisfy the completeness criterion, other sequence is padded.

After finding a PreUIO that satisfies the completeness criterion, an extended UIO is constructed so that it might satisfy Definition 3. The EUIO is found by adding an outgoing edge to a PreUIO using the BFS. If the extended sequence uses up all the edges in the FSM and cannot find a sequence satisfying Definition 3, the algorithm fails and stops. CheckPreUIO is the routine that checks whether two input sequences have the same input sequence or not.

4.2. An example: k-strong FSM for EUIO

Here, we illustrate the *k-strong* FSM with a sample FSM. For the FSM of Fig. 3, the edge $(Si, Sj; a/4)$ will be tested using a minimal UIO and the extended UIO, respectively. We will show that our scheme satisfies the completeness criterion.

First, minimal UIO for Sj is computed: $(b/1, c/2, c/2, b/2)$. Then the degree of difference of each state is computed by Definition 2:

$S1$

$$\mathrm{DoD}((Sj, S1),\ (S4, S1)) = \text{infinite}$$

$S2$

$$\mathrm{DoD}((Sj, S1, S2),\ (S5, S4, S2)) = \text{infinite}$$
$$\mathrm{DoD}((Sj, S1, S2),\ (S4, S3, S2)) = \text{infinite}$$
$$\mathrm{DoD}((Sj, S1, S2),\ (S4, S1, S2)) = \text{infinite}$$

$S3$

$$\mathrm{DoD}((Sj, S1, S2, S3),\ (Sj, S5, S4, S3)) = \text{infinite}$$
$$\mathrm{DoD}((Sj, S1, S2, S3),\ (S6, S5, S4, S3)) = \text{infinite}$$

$S4$

$$\mathrm{DoD}((Sj, S1, S2, S3, S4),\ (S6, Si, S6, S5, S4)) = 4$$

As seen in the above result, $\min(\mathrm{DoD}(\mathrm{UIO}(Sj), W)) = 4$ for all walks W in FSM. As $2 * 1 + 1 < 4 < 2 \times 1 + 3$, the FSM is *1-strong* for the extended UIO by Definition 3. Now, assume that the edge $(S2, S3; c/2)$ has been incorrectly implemented with an output fault as $(S2, S3; c/3)$. After evaluation of DoDs, output sequences by each method are as follows:

Using Minimal UIO,

Expected output sequence: 1 2 2 2
Observed output sequence: 1 2 3 2
$\Rightarrow$ Verdict as FAIL.

Using Extended UIO,

Expected output sequence: 1 2 2 2
Observed output sequence: 1 2 3 2
$\Rightarrow$ Since DoD $= 1 \Leftarrow 1$, Verdict as PASS.

In case of the minimal UIO scheme, test sequence could not decide whether the fault arises in the edge being tested or one of the edges in the UIO sequence, whereas our

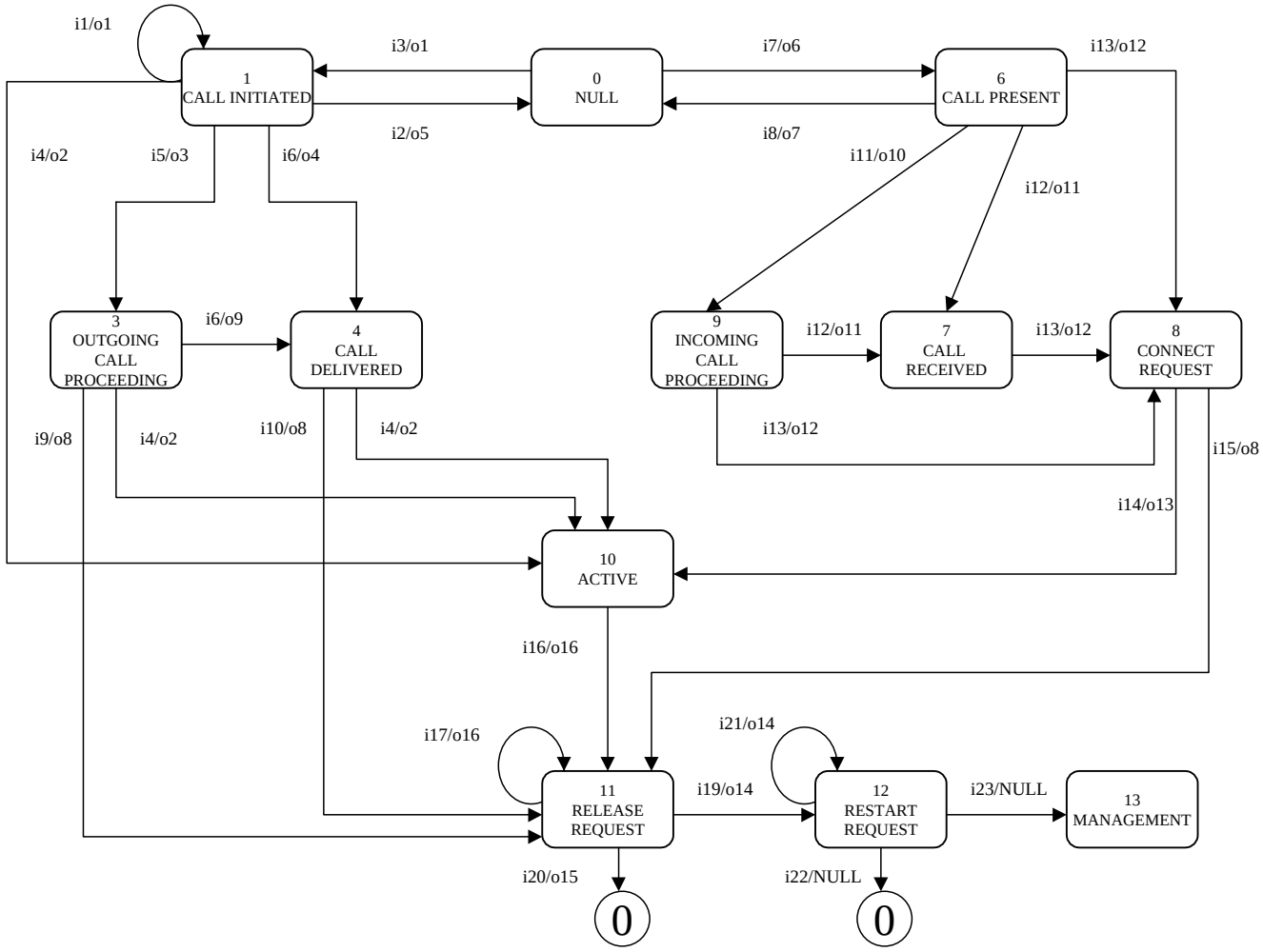

Fig. 4. Simplified transition diagram of Q.2931.

scheme can tell that the fault arises in the UIO sequence with high probability. Thus, a tester is sure that the edge being tested is conformed to the specification.

Besides the fault correction capability, considering the completeness criterion can reduce the length of the test sequence. For example, by testing the edge $(Si, Sj; a/4)$, we do not need to test the edges in verifying sequence, $(Sj, S1, S2, S3, S4; b/1, c/2, c/2, b/2)$. Because there are no converging walks in paths from Sj to $S4$, we do not need to compute $S1$, $S2$, $S3$ and $S4$s UIOs and test them either. We will explain this property in detail in Section 5. In this example, as the FSM is *1-strong* for the extended UIO, the test sequence has only one fault correction capability.

4.3. Case study: EUIO for Q.2931

Here, we introduce the *4-strong* FSM of Q.2931 and find the EUIO of each state for conformance testing. Q.2931 is a call/connection establishment protocol of B-ISDN, and we simplified its transition diagram in Fig. 4.

In the FSM, we substituted each input/output signal as follows:

Input signals of Q.2931

i1: 1st Timeout T303
i2: 2nd Timeout T303
i3: Setup.req
i4: CONNECT
i5: CALL PROCEEDING
i6: ALERTING
i7: SETUP
i8: Release.resp
i9: Timeout T310
i10: Timeout T301
i11: Proceeding.req
i12: Alerting.req
i13: Setup.resp
i14: CONNECT ACK
i15: Timeout T313
i16: release.req
i17: 1st Timeout T308
i19: 2nd Timeout T308

i20: RELEASE COMPLETE OR RELEASE
i21: 1st and 2nd Timeout T316
i22: RESTART ACK
i23: 3rd Timeout T316

Output signals of Q.2931

o1: SETUP
o2: Setup.conf + CONNECT ACK
o3: proceeding.ind
o4: Alerting.ind
o5: Release.ind
o6: Setup.ind
o7: RELEASE COMPLETE
o8: RELEASE + Release.ind
o9: Alert.req
o10: CALL PROCEEDING
o11: ALERTING
o12: CONNECT
o13: Setup.complete.ind
o14: RESTART
o15: Release.conf
o16: RELEASE

Following are the extended UIOs of each state:

*S*1: i7/o6 i8/o7 i3/o1 i1/o1 i5/o3 i9/o8 i17/o16 i19/o14 i21/o14 i23/null
*S*2: i2/o5 i7/o6 i8/o7 i3/o1 i1/o1 i5/o3 i9/o8 i17/o16 i19/o14 i21/o14 i23/null
*S*3: i9/o8 i17/o16 i19/o14 i21/o14 i22/null i7/o6 i11/o10 i12/o11 i13/o12
*S*4: i10/o8 i17/016 i19/o14 i21/o14 i22/null i7/06 i8/o7 i3/o1 i1/o1 i5/o3 i9/o8
*S*6: i8/o7 i3/o1 i1/o1 i5/o3 i9/o8 i17/o16 i19/o14 i21/o14 i23/null
*S*7: UIO does not exist
*S*8: i15/o8 i17/o16 i20/o15 i3/o1 i1/o1 i5/o3 i9/o8 i20/o15 i7/o6 i13/o12
*S*9: UIO does not exist
*S*10: i16/o16 i17/o16 i19/o14 i21/o14 i22/null i7/o6 i8/o7 i3/o1 i1/o1 i5/o3 i9/o8
*S*11: i20/o15 i3/o1 i5/o3 i9/o8 i19/o14 i22/null i7/o6 i11/o10 i13/o12
*S*12: i21/o14 i22/null i3/o1 i1/o1 i5/o3 i9/o8 i20/o15 i7/o6 i12/o11
*S*13: N/A

The extended UIO that makes an FSM *k-strong* can make verdict not only on the edge being tested but also on the edges included in the characterization sequence. In addition, the method can make verdict on edges that cannot be tested by existing methods. For example, using the EUIO of *S*3, *S*8, *S*11, *S*12, our method can make verdicts on edges, on which existing methods cannot. The EUIO of *S*3 can make verdict on edge $(S6, S9; i11/o10)$ and $(S9, S7; i12/o11)$ and $(S7, S8; i13/o12)$. The EUIO of *S*8 can make verdict on edge $(S6, S8; i13/o12)$. The EUIO of *S*11 can make verdict on edge $(S9, S8; i13/o12)$. The EUIO of *S*12 can make verdict on edge $(S6, S7; i12/o11)$. This is because the proposed method has the property that if there are faults less than *k*, the method says where faults arose. But there may be states that have no EUIO satisfying the property. Thus, on conformance testing, edges are tested with EUIO first, and edges not covered by EUIO are tested with existing schemes such as traditional UIO.

The edge can be tested as follows: for example, the edge to be tested is $(S0, S1; i3/o1)$. The characterization sequence of *S*1 is (i7/o6 i8/o7 i3/o1 i1/o1 i5/o3 i9/o8 i17/o16 i19/o14 i21/o14 i23/null). The sequence satisfies $2 \times 4 + 1 \Leftarrow \mathrm{EUIO(SI)} < 2 \times 4 + 3$. Hence we can detect and correct 4 faults. If the output result is (o1,o5,o6, o7,o2,o2,o1,o8,o16,o13,o14,null), the tester can make verdict that $(S0, S1; i3/o1)$ is correct and $(S1, S0; i2/o5)$, $(S0, S6; i7/o6)$, $(S6, S0; i8/o7)$, $(S3, S11; i9/o8)$, $(S11, S11; i17/o16)$ and $(S12, S13; i23/null)$ are correct. Moreover, there are faults on $(S0, S1; i3/o1)$, $(S1, S1; i1/o1)$, $(S1, S3; i5/o3)$ and $(S12, S12; i21/o14)$.

Here, there is no need of finding characterization sequence to test $(S1, S0; i2/o5)$, $(S0, S6; i7/o6)$, $(S6, S0; i8/o7)$, $(S3, S11; i9/o8)$, $(S11, S11; i17/o16)$, $(S12, S13; i23/null)$, $(S0, S1; i3/o1)$, $(S1, S1; i1/o1)$, $(S1, S3; i5/o3)$ and $(S12, S12; i2/o14)$. Therefore, the length of the overall test sequence is substantially reduced.

5. Cyclic input characterization sequence

Larger the *k* in *k-strong* FSM, stronger the immunity against output faults in a CS. Sometimes, we cannot make a given specification FSM *k-strong*. An improvement for these drawbacks is suggested in this section. The improvement uses the cyclic sequence of which concept was originally in Ref. [11] to make an integrated test sequence that tests both the control part and the data flow of protocols. We adapt the cyclic sequence to *k-strong* FSM.

5.1. Making an arbitrary FSM k-strong

Despite many advantages of the EUIO, we cannot make an arbitrary FSM *k-strong* with the EUIO presented in Section 3. This is clear from the fact that if there is any converging walk whose length is less than or equal to 2*k*, then the FSM cannot be *k-strong* for EUIO. Even though we can still test the other transitions satisfying the CS selection criterion, EUIO approach is very restrictive. Thus, the strength of an FSM is not a controllable attribute but a given attribute.

In this section, we propose a new characterization sequence which makes almost every FSM to be *k-strong* for arbitrary *k*. The characterization sequence is composed of cyclic input characterization sequences (CICS), a postamble sequence (PAS), and a basic characterization sequence (BCS) such as UIO and DS. This structure is shown in Fig. 5.

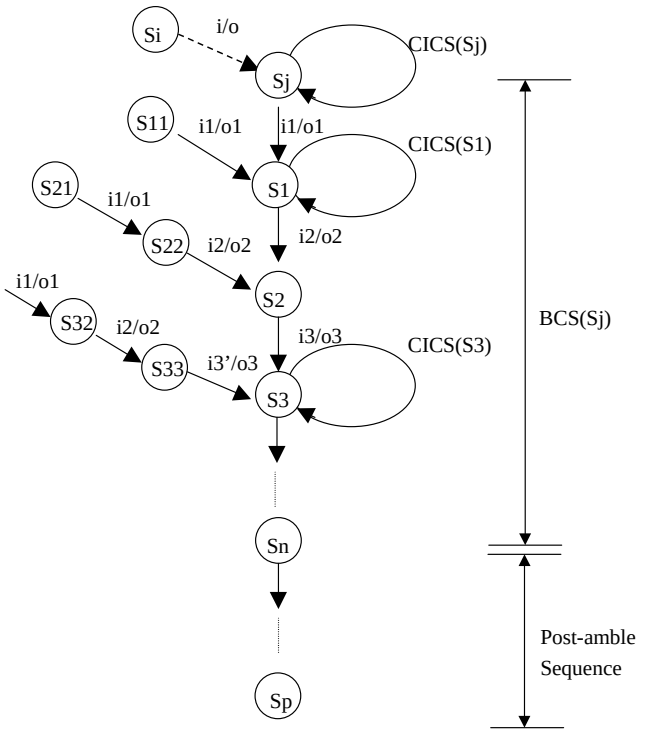

Fig. 5. Integrated characterization sequence (ICS).

First, we define the CICS (Cyclic Input Characterization Sequence), which is essential to make an FSM *k-strong*.

Definition 6 ((CICS(k, Sn, W))). For a walk $W = (Sj, S1, S2, \ldots, Sn, Sn + 1; \text{i1/o1}, \text{i2/o2}, \ldots, \text{in/on}, \text{in} + 1/\text{on} + 1)$, CICS($k, Sn, W$) is a cyclic input/output sequence which satisfies

$$\text{DoD}(n\text{-Head}(W)@\text{CICS}(k, Sn, W)@\text{in} + 1/\text{on} + 1, A) > = 2k + 1,$$

where A is any walk reaching to $Sn + 1$ with the same input sequence as that of n-Head(W)@CICS(k, Sn, W)@in + 1/on + 1.

By inserting CICS(k, Sn, W) at a converging state Sn, any converging walk into $Sn + 1$ has enough output differences with a new characterization sequence n-Head(W)@CICS(k, Sn, W)@in + 1/on + 1. It is clear that there is no need to insert any CICS when there are no convergent states in BCS. Using the CICS, we can make an FSM *k-strong* with very high probability for arbitrary k. CICS needs just slightly stronger conditions than CCS (Cyclic Characterization Sequence) as Ref. [11] does.

When the total length of the BCS including CICS is less than $2k + 1$, the FSM cannot be *k-strong* for the characterization sequence. In this case, we pad the post-amble sequence to the test sequence composed above so that the FSM might be *k-strong*.

Definition 7 (Post-amble Sequence PAS(W))). For a walk $W = (Sj, S1, S2, \ldots, Sn, Sn + 1; \text{i1/o1}, \text{i2/o2}, \ldots, \text{in/on}, \text{in} + 1/\text{on} + 1)$, if $\text{DoD}(W, A) < 2k + 1$, the *post-amble sequence PAS*(W) is the shortest input/output sequence which makes

$$\text{DoD}(W@\text{PAS}(W), A) >= 2k + 1,$$

where A is any walk reaching to Tail(PAS(W)) with the same input sequence as that of W@PAS(W).

We call the concatenation of BCS, CICS, and PAS as Integrated Characterization Sequence (ICS). If at least one CICS is added during the ICS construction, a post-amble sequence does not need to be padded since already the ICS is long enough for the FSM to be *k-strong*. The post-amble sequence is padded only if the BCS is too short to make the FSM *k-strong* and no CICS is inserted.

The simple *CICS*(k, vi, W) construction algorithm is as follows:

```
set yes
for i from 2k + 1 to |E|
  for each possible cyclic sequence CC(vi) with |CC(vi)| = i do
  for each converging walk CW with n-Head(W)@CC(vi)@in + 1/on + 1 do
  if DoD(CW, n-Head(W)@CC(vi)@in + 1/on + 1) < 2k + 1 then
  set no
  break
  endif
  endfor
  if set?yes then break
  endfor
  if set?yes then break
endfor
if set?no then error("This FSM cannot be k-strong for ICS")
CICS(k, vi, W) = CC(vi)
```

The cyclic sequence composed in this way does not have optimal length, hence, we may shorten the length of the test sequence and reduce the time complexity by using more sophisticated CICS(k, vi, W) construction algorithm. Using the BCS, CICS, and PAS, we can construct the ICS as follows:

```
(* ICS Initialization *)
Find BCS(Sj)
ICS(Sj) := BCS(Sj)
(* CICS padding to the ICS *)
for each state vi in BCS(Sj)
  if vi is a converging state then
```

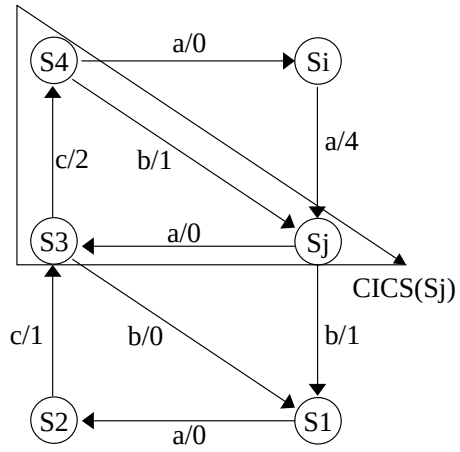

Fig. 6. Example testing with ICS.

add CICS(k, vi, ICS(Sj)) to ICS(Sj)

endif

endfor

(* Post-amble sequence padding *)

if no *CICS*(k, vi, ICS(Sj)) is added and the length of the BCS(Sj) $< 2k + 1$ then

while DoD(ICS(Sj), A) $< 2k + 1$ do

Extend ICS(Sj) by one edge

Compute DoD(ICS(Sj),A) for all walks A with the same input sequence as that of

ICS(Sj)

endwhile

endif

5.2. Testing with ICS

Testing an implementation FSM with ICS is the same as that with EUIO. As before, a test segment for the edge (Si, Sj; i/o) is

ri@SP(Si)@i/o@ICS(Sj).

In the ICS construction algorithm, BCS(Sj) may be any well-known characterization sequence such as UIO, DS, *W*-set, and their variants. The ICS is just a template to construct an instance of a characterization sequence. This property is resulted from the fact that the ICS is defined without regard to BCS. Thus, our scheme is adaptable to various environments. For example, if a test suite designer wants a characterization sequence with wider fault coverage, she can use a fault tolerant UIO or a pair wise distinguishing sequence [6,7].

Table 1
ICS for the example FSM

State	UIO	ICS	DoD
Si	(a/4)	(a/4, a/0, b/0)	Infinite
Sj	(b/1, a/0, c/1)	(a/0, c/2, b/1, b/1, a/0, c/1)	Infinite
$S1$	(a/0, c/1)	(a/0, c/1, c/2)	Infinite
$S2$	(c/1)	(c/1, c/2)	Infinite
$S3$	(c/2)	(c/2, a/0)	Infinite
$S4$	(a/0, a/4)	(a/0, a/4, a/0)	Infinite

Fig. 6 shows an FSM that has a convergent state. We will use UIO as a BCS. First, assume that the edge (Si, Sj; a/4) is under test. For the tail state Sj verification, a UIO for Sj is computed: (b/1, a/0, c/1). Here, $S1$ is a convergent state as ($S3$, $S1$; b/0) has the same input sequence as (Sj, $S1$; b/1). Thus, CICS(1,Sj,UIO) should be computed: (a/0, c/2, b/1). By definition, ICS(Sj) = (a/0, c/2, b/1)@(b/1, a/0, c/1). We do not pad a post-amble sequence because ICS(Sj) has already a CICS. By using EUIO, we cannot test the edge (Si, Sj; a/4) with completeness criterion since there is a converging walk.

Now, we present the case when there is no converging walk and the BCS is too short to construct a characterization sequence with $2k + 1$ output differences. Assume that the edge to be tested is (Sj, $S1$; b/1). BCS is (a/0, c/1). The example FSM is not *1-strong* for the BCS since there is a walk (a/0, c/2) and DoD(BCS, (a/0, c/2)) $= 1 < 2 \times 1 + 1$. In this case, no CICS is inserted to ICS, so we need to pad a post-amble sequence. A PAS padded ICS is computed: (a/0, c/1, c/2). In both the cases,

DoD(ICS, A) = infinite for all walks A in the FSM.

Table 1 shows UIO, ICS, and corresponding DoD for each state. The example FSM is infinite-strong for the ICS. This fact states that the FSM satisfies the completeness criterion for the ICS regardless of the number of output faults.

6. Test sequence optimizing technique

In this section, we present that the length of our test suite can be reduced. Our test sequence generation method has the property that it exactly verdicts the correctness of the state verifying sequence. Let us assume that the edge under test is (Si, Sj; i/o), our state verifying sequence (EUIO or ICS) of Sj is (Sj, $S1$, $S2$, $S3$, $S4$; i1/o1, i2/o2, i3/o3, i4/o4), and the FSM is *1-strong*. If the result of the test is (PASS, PASS, PASS, FAIL, PASS), then we are certain that the edge under test is correctly implemented even though there is one output fault in the verifying sequence. Here, note that the other edges do not need to be tested, since our test sequence has the completeness property. More specifically, because there are no converging walks in paths from Sj to $S4$, we do not need to compute $S1$, $S2$, $S3$ and $S4$s state verifying sequences and test their edges either. With the optimization technique, we can improve the time complexity to test an FSM with our test sequence. Edges that are not tested using the method will be tested using classical conformance testing methods.

7. Time complexity

The number of cyclic sequence whose length is about $2k + 1$ is roughly the same as the average running time of the CICS(k, vi, W) construction algorithm. Let m be the average outgoing degree of a state of an FSM. Since the number of walks whose length is $2k + 1$ is m^{2k+1}, we can estimate the average running time of the CICS(k,vi,W) construction algorithm to be $O(m^{2k+1})$. Note that the number of walks whose length is $2k + 1$ is smaller than that of cyclic sequences, hence the running time is much less than $O(m^{2k+1})$.

It takes about $O(((c/2) \times m^{2k+1})(1 - 1/2^c) + m^{2(2k+1-c/2)} \times (1/2)^c)$ for ICS construction algorithm to complete the procedure, where c is the average length of the BCS. We assume that half of states of BCS are convergent, hence $c/2$ CICSs' should be constructed. Using the CICS construction time estimation, we can estimate the ICS construction time to be $O(c/2 \times m^{2k+1})$. In case no CICS is added, only the PAS construction time is counted. As before, we assume that BCS has only $c/2$ output differences with other walks. Thus, only $(2k + 1 - c/2)$ output differences are needed to make an FSM *k-strong* and a PAS with the length $2 \times (2k + 1 - c/2)$ should be padded. If we assume that a state of the BCS is converging with the probability 1/2, then the average running time of the ICS construction is $O((c/2 \times m^{2k+1}) \times (1 - 1/2^c) + m^{2(2k+1-c/2)} \times 1/2^c)$.

Even though our estimation is not so tight to the actual running behavior of the algorithms, the proposed characterization sequence can be constructed within a reasonable time bound.

8. Conclusion

In this paper, we present a problem of commonly used characterization sequences and propose two test sequence generation schemes to resolve the problem. The first one can be applied to various test sequences such as UIO-, DS-, *W*-, and their variants, though only a construction strategy using UIO sequence is presented. Since the test sequence satisfies the completeness criterion, it could decide whether the fault arises in the edge being tested or in one of the edges in the UIO sequence. Additionally, the fault coverage is much wider than other test sequence generation methods, as the test segment for a tail state verification is longer than others [6].

However, the length of the proposed test sequence is about two times longer than the original one if the simple test sequence construction algorithm is used from the beta sequence. The other test sequence generation scheme that is constructed using CICS has much shorter length than the first one. Besides that, it greatly increases the probability that a given FSM might be *k-strong*. By definition, it also satisfies completeness criterion. The ICS is just a template to construct an instance of a characterization sequence. This property resulted from the fact that the ICS is defined without regard to a certain instance of the BCS. Thus, our scheme is adaptable to various environments.

The test sequences satisfying the completeness criterion are longer than existing test sequences. To resolve this inefficiency, we take advantage of the completeness property of the test sequence. To construct the CICS, we use a simple brute-force algorithm in this paper, but more sophisticated algorithm that generates shorter CICS has a lower time complexity could be considered.

In this paper, we proposed just a beta sequence construction scheme. Overall test sequences are made from the concatenation of test segments. With our beta sequence and the optimizing technique, we can reduce the total length of overall test sequences using RCPT. If we make an RCPT so that the edges in the BCS whose head state is already verified might not be visited during the tour, the overall test sequences have an optimal length. The proposed scheme is also applicable to software testing and the conformance testing of sequential circuits.

References

[1] D.P. Sidhu, T.-K. Leung, Formal methods for protocol testing, a detailed study, IEEE Trans. SE 15 (4) (1989) 413–426.
[2] K. Sabnani, A. Dahbura, A protocol test generation procedure, Computer Networks and ISDN Systems 15 (1988) 285–297.
[3] Y.N. Shen, F. Lombardi, A.T. Dahbura, Protocol conformance testing using multiple UIO sequences, IEEE Trans. Comm. 40 (8) (1992) 1282–1287.
[4] A.V. Aho, T. Dahbura, D. Lee, M. Umit Uyar, An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours, IEEE Trans. Comm. 39 (11) (1991) 1604–1615.
[5] H. Ural, K. Zhu, Optimal length test sequence generation using distinguishing sequences, IEEE Trans. Networking 1 (3) (1993) 358–371.
[6] K. Naik, Fault-tolerant UIO sequences in finite state machines, IWPTS'95 (1995) 207–220.
[7] R.E. Miller, S. Paul, Structural analysis of protocol specifications and generation of maximal fault coverage conformance test sequences, IEEE/ACM Trans. Networking 2 (5) (1994) 457–470.
[8] F. Lombardi, Y.N. Shen, Evaluation and improvement on fault coverage of conformance testing by UIO sequences, IEEE Trans. Comm. 40 (8) (1992) 1288–1293.
[9] D. Rayner, OSI conformance testing, Computer Networks and ISDN Systems 14 (1987) 79–98.
[10] R.B. Ash, Information Theory, Dover, New York, 1990, pp. 89–90.
[11] S.T. Chanson, J. Zhu, A unified approach to protocol test sequence generation, Proc. In IEEE INFOCOM'93 (1993) 106–114.