International Journal of Applied Science and Engineering
2003. 1, 1: 45-54

An Optimization Technique for Protocol Conformance Testing Based on the Wp Method

Wen-Huei Chen*

Department of Electronic Enginering, Fu Jen Catholic University,
Taipei 242, Taiwan, R.O.C.

Abstract: In order to ensure the correct operation of a distributed system, the protocol implementation must be tested for conformance to the specification that is defined as a standard. The *UIOv* and *Wp methods* are two formal methods in generating the test sequence. In the past decade, a lot of new techniques have been proposed to optimize the test sequence resulting from the UIOv method. On the other hand, the traditional *Reset technique* is still used in the Wp method to generate a test sequence that is very long. In this paper, we propose a new technique to optimize the test sequence resulting from the Wp method. The technique involves the construction of the test segments from the Wp method, and a *Rural Chinese Postman Algorithm* which optimally connects these test segments into a test sequence. Moreover, the technique is extended to generate the synchronizable test sequence. A lot of optimization techniques used in the UIOv method can then be modified to accommodate the Wp method based on similar extensions.

Keywords: protocol engineering; conformance testing; test sequence; Wp set.

1. Introduction

A distributed system is composed of many *parties* (i.e., computers, instruments, etc.) remotely connected by communication *links* (i.e., cables, fibers, etc.) through which messages are transmitted [2]. A *protocol* is the representation of as well as the orderly exchange of these messages that must be agreed on by any party before using it, and a set of protocols is usually layered to establish a complex communicating behavior. In each party, a protocol is implemented in either software or hardware or firmware that has an upper and lower interface to the upper- and lower-layer protocol(s) [22].

The objective of *protocol conformance testing* is to see if a protocol implementation conforms to the protocol specification defined as a standard. According to the "OSI Conformance Testing Methodology and Framework" proposed by the International Standard Organization (ISO) [14], the implementation must be tested as a black box. As shown in Figure 1,

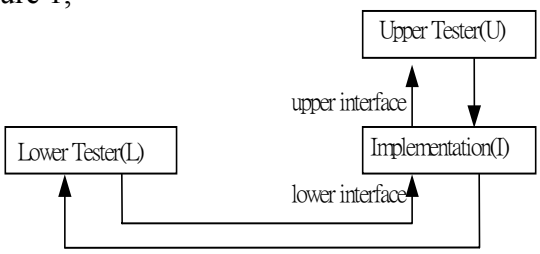


Figure 1. A protocol testing system

the upper interface of the implementation (I) is controlled and observed directly by the upper tester (U), and the lower interface of the

* Corresponding author: E-mail: whchen@ee.fju.edu.tw
Accepted for Publication: Feb. 26, 2003

implementation (I) is controlled and observed indirectly by the lower tester (L). Inputs are sent from the upper and lower testers to the implementation, and the outputs are checked to see that they are as expected. The sequence of input/output pairs is the *test sequence*, and the number of inputs is the test sequence *length* [23].

The test sequence is generated from the protocol specification which is usually decomposed into a control and a data portion [8]. The control portion determines how messages are sent and received. It can be considered a *deterministic Finite State Machine* (FSM) which contains *states* and *transitions*. Initially, the FSM is in a specific state called the *initial state*. An input (i.e., stimulus) will cause the FSM to generate output(s) (i.e., responses) and to change from the current state to a new state; this process is a *transition* [11]. The data portion specifies some supplementary parameters. It can be informally described by a set of rules among parameter values [5] or formally specified by an Extended Finite State Machine (EFSM) [15]. This paper is concerned with testing the control portion.

In testing the control portion, machine identification experiments [15] in finite state machine theory have been applied and are known as formal methods [8]. The typical aim of these methods is to generate a test sequence which guarantees that the transitions are correctly implemented. The test sequence contains a *preparatory* and a *checking part.* The preparatory part checks that the *State-Identification (SI)* sequence(s) (derived from the FSM specification) exist in the implementation. The checking part checks that each transition is correctly implemented by the test segments constructed from the SI sequence(s). Traditionally, the *Reset technique* is used to connect the test segments of the checking part. That is, the implementation is taken from the initial state to where each test segment starts, the test segment is applied, and the implementation is and the implementation is reset to the initial state.

Well-known formal methods include the *T* [19], *D* [13], *Wp* [23] *UIOv* [4][11] and method which use no sequences, *Distinguishing Sequence* [13], *Unique Input/Output (UIO) Sequence* [20] and *Characterization Set* (*W set*) [7] as the SI sequence(s) respectively. Experimentations suggest that the T method cannot detect a lot of faulty implementations [21] and only very few protocols have the Distinguishing Sequences [13]; so the T and the D methods are seldom used in reality. All protocols have the W set so the W method can be applied, however, the generated test sequence is quite long. The UIOv method produces a shorter test sequence but is not applicable to all protocols (a few protocols do not have UIO sequences.) The UIOv and Wp methods are the most and secondly popular methods respectively.

Fault detection capability of the UIOv and Wp methods have been proved theoretically. It is proved in [4][11][23] that the Reset technique can connect their test segments into a test sequence which satisfies the *existence criterion*, i.e., the generateed test sequence can detect any k-state faulty implementation, where k is not larger than the number of the states of the FSM. However, such a proof is based on the assumption that the reset transitions are correctly implemented. Unfortunately, some FSMs do not have the reset transitions. And, even if they have, the reset transitions (just like the other transitions) may be incorrectly implemented by an engineer. On the other hand, the Reset technique uses a lot of overhead sequences to take the FSM to/from the initial state. These overhead sequences make the test sequence very long.

The UIOv method based on the Reset technique has been singifically improved in the past decade. In [1], an *RCP (Rural Chinese Postman) technique* has been proposed to replace the Reset technique by connecting the test segments into a shorter test sequence.

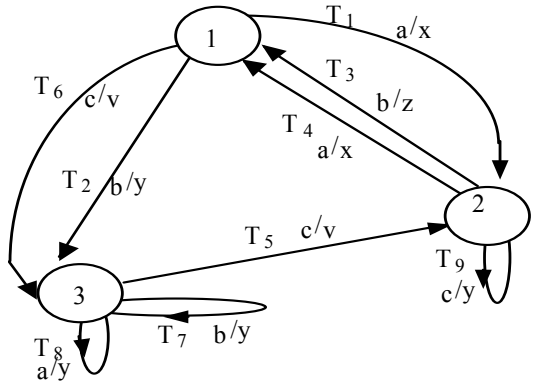
The RCP technique uses minimum-length overhead sequences to directly connect these test segments into a continuous test sequence, which doesn't need to make repeated use of reset transitions. Moreover, because the industry seeks even shorter test sequences, the preparatory part is not included in the final test sequence. A lot of papers have explored further extensions by overlapping the test segments or applying the technique to other models [8]. These results have made the approach of testing the control portion using the UIO sequences very popular.

On the other hand, the Wp method still use the Reset technique (which is the only technique to connect the test segments.) Though the Wp method in the FSM model has been applied to the Communicating Nondeterminsitc FSM model [17] and the Real-Time FSM model [9], the lengthy test sequence due to the Reset technique has made the Wp method unpoular in reality. In this paper, we are first to apply the RCP technique to the Wp method. And as in [1], the preparatory part is not included in the final test sequence. We connect test segments of the checking part into a continuous test sequence using minimum-length overhead sequences. By introducing the Rural Chinese Postman Algorithm [1], an optimal test sequence which includes all the (non overlapping) test segments can be found if the protocol satisfies either the returning or self-loop property [1]. Then, we extend the technique to generate the synchronizable test sequence [3]. It is expected that our optimization techniques can be further improved just like those techniques of [1] and can be extended to other models [9][17] other than the current FSM model.

In Section 2, the Wp method is reviewed. In Section 3, the optimization technique is proposed. In Section 4, the technique is extended to generate the synchronizable test sequence. In Section 5, other extensions of the technique are discussed.

2. Review of the checking part of the Wp method

Consider an FSM represented by the transition digraph G(V, E) of Figure 2[1]. The vertex set V = {1, 2, 3} represents the states and the edge set E = {T_1, T_2, ..., T_9} represents the transitions. Each transition T_j = (K, L; i/o) is a transition from the current state K to the next state L caused by an input "i" with an output "o". "1" is the initial state. A consecutive sequence of edges E_1, E_2, . . ., E_q is called a *path* denoted by [E_1, E_2, . . ., E_q]. A path which starts and ends at the same vertex (in this paper, specifically the initial state) is called a *tour*. In Figure 1, only transitions T_1, T_2, ..., T_6 will be checked as they represent the main behavior of the protocol.



Remarks:
a: input from U to I b, c: inputs from L to I
z: output from I to U x, y, v: outputs from I to L

Figure 2. The transition digraph G(V, E) of an FSM M

An *input sequence* will cause the FSM in a specific state to produce a corresponding *output sequence*. A set of input sequences called set *I* will cause the FSM in state J to produce a set of output sequences, namely, $OUT_j(I)$. For example, consider *I* = {a, b}. $OUT_1(I)$ = OUT_1({a, b}) = {x, y}, because input "a" will

[1] As assumed in [11], the FSM is minimal (i.e., none of its states are equivalent), completely specified (i.e., the same input can be applied to all states), and G(V, E) is strongly connected (i.e., there is a path from any vertex to any other vertex.).

cause the FSM in state 1 to output "x" (see transition T_1) and input "b" will cause the FSM in state 1 to output "y" (see transition T_2). *I* is a W set if $OUT_j(I)$ is distinct for any state J. For example, *I* = {a, b} is a W set because $Output_1(\{a, b\}) = \{x, y\}$, $Output_2(\{a, b\}) = \{x, z\}$, and $Output_3(\{a, b\}) = \{y, y\}$ are all distinct. The Wp set (i.e., the partial W set) is a subset *I'* of the W set, such that $Output_j(I')$ is different from $Output_k(I')$ when $k \neq j$. For example, consider the subset {b} of the W set {a, b}. $Output_1(\{b\}) = \{y\}$, $Output_2(\{b\}) = \{z\}$ and $Output_3(\{b\}) = \{y\}$. {b} is a Wp set for state 2, because $Output_2(\{b\})$ is different from both $Output_1(\{b\})$ and $Output_3(\{b\})$. Similarly, {a} is the Wp set for state 3, and {a, b} is the Wp set for state 1.

Assume that Wp(J) has q input sequences. In G(V, E), these q input sequences correspond to q paths starting from vertex q, namely, $Wp(J)^1$, $Wp(J)^2$, ..., $Wp(J)^q$ (or simply Wp(J) if q =1.) For example, {a, b} is a Wp set for state 1. The inputs "a" and "b" correspond to paths $Wp(1)^1 = [T_1]$ and $Wp(1)^2 = [T_2]$. Similarly, Wp(2)= $[T_3]$, Wp(3) = $[T_8]$. Every transition of the set $\{T_1, T_2, ..., T_6\}$ will be checked in the checking part of the Wp method. Each transition T_r = (J, K; i/o) is checked by the test segments $[T_r, Wp(K)^1]$, $[T_r, Wp(K)^2]$, . . . , $[T_r, Wp(K)^q]$ where T_r tests the input/output pair of the transition, and $Wp(K)^1$, $Wp(K)^2$, . . . , $Wp(K)^q$ confirm the ending state. These test segments are called $Check(T_r)^1$, $Check(T_r)^2$, …, $Check(T_r)^q$. For example, to check transition T_3 = (2, 1; b/z), we have $Check(T_3)^1 = [T_3, Wp(1)^1] = [T_3, T_1]$ and $Check(T_3)^2 = [T_3, Wp(1)^2] = [T_3, T_2]$. The other test segments are listed in Table 1.

Table 1. A set of test segments for the FSM of Figure 2

Starting State	Test Segments	Ending State
1	$Check(T_1) = [T_1, T_3]$	1
1	$Check(T_2) = [T_2, T_8]$	3
2	$Check(T_3)^1 = [T_3, T_1]$	2
2	$Check(T_3)^2 = [T_3, T_2]$	3
2	$Check(T_4)^1 = [T_4, T_1]$	2
2	$Check(T_4)^2 = [T_4, T_2]$	3
3	$Check(T_5) = [T_5, T_3]$	1
1	$Check(T_6) = [T_6, T_8]$	3

The Reset technique will connect the test segments of Table 1 into a continuous test sequence. It does not connect them directly because the test segments do not start and end at the initial state. Instead, the technique inserts an overhead sequence before each test segment to make it start from the initial state, and it inserts the reset transition (or another overhead sequence afterward to make it end at the initial state.) For example, in Table 1, consider the test segment $Check(T_4)^2 = [T_4, T_2]$ which starts at state 2 and ends at state 3. As shown in Figure 3, we insert "T_1" before $[T_4, T_2]$ so that $[T_1, T_4, T_2]$ starts from state 1, and insert "T_5, T_4" afterward so that $[T_1, T_4, T_2, T_5, T_4]$ ends at state 1. The complete test sequence is shown in Figure 3.

3. An optimization technique for checking part of the Wp method

In Section 2, the Reset technique is used to connect the test segments of the checking part of the Wp method, by extending each test segment to start and end at the initial state. In this section, the RCP technique will be used to directly connect these test segments into a minimum-length test sequence by using a minimum number of minimum-length overhead sequences. As described in Section 1, the preparatory part is not included in the fi-

nal test sequence. The technique involves two steps.

T_1, T_3 /* Check(T_1) */
T_2, T_8, T_5, T_4 /*Check (T_2) */
T_1, **T_3, T_1,** T_3 /*Check(T_3)^1 */
T_1, **T_3, T_2,** T_5, T_4 /* Check(T_3)^2 */
T_1, **T_4, T_1,** T_3 /* Check(T_4)^1 */
T_1, **T_4, T_2**, T_5, T_4 /*Check(T_4)^2 */
T_2, **T_5, T_3** /*Check(T_5) */
T_6, T_8, T_5, T_4 /*Check(T_6) */
total length = 31

Figure 3. The test sequence constructed from the test segments of Table 1 by the Reset technique (test segments are in bold face)

First, the test segments of Table 1 are embedded as bold edges into the transition digraph G(V, E) of Figure 1, resulting in a W digraph G'(V', E') of Figure 4. In Table 1, each test segment which starts at state J and ends at state K is embedded as a bold edge from vertex J to vertex K. For example, the test segment Check(T_5) = [T_5, T_3] of Table 1 is embedded as a bold edge (3, 1; Check(T_5)) in Figure 4, because the path [T_5, T_3] starts from state 3 and ends at state 1. The cost of an edge is defined as the number of input/output pairs that is associated with the edge. Hence, the cost of the bold edge (3, 1; Check(T_5)) is 2 because it has two input/outputpairs.

Second, the Rural Chinese Postman Tour [1] of the W digraph is used to generate a minimum-legnth test sequence which includes all the (non-overlapping) test segments. The Rural Chinese Postman Tour of a digraph is a minimum-cost tour which traverses every bold edge at least once. In Figure 4, the cost of an edge is defined as the number of input/output pairs associated with the edge. For example, the Rural Chinese Postman Tour of Figure 4, i.e., Check(T_1), Check(T_2), Check(T_5), Check(T_6), T_5, Check(T_3)^2, T_5, Check(T_4)^1, T_2, T_5, Check(T_4)^2, T_5, Check(T_3)^1 is used to generate the optimal test sequence of Figure 5.

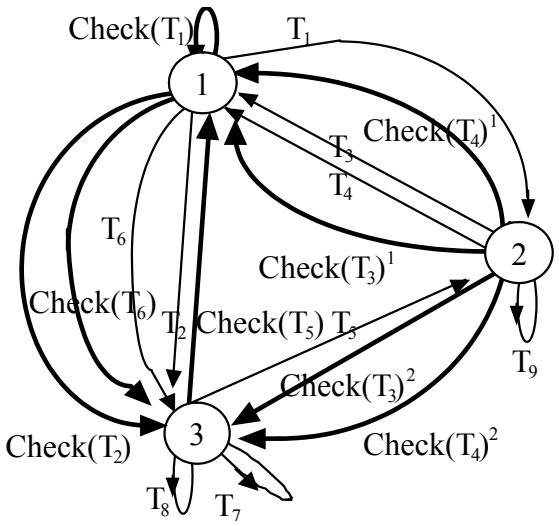


Figure 4. The W digraph G'(V', E') constructed from the transition digraph of Figure 1 by embedding the test segments of Table 1

T_1,T_3,T_2,T_8,T_5,T_3,T_6,T_8,T_5,**T_3,T_2,**T_5,**T_4, T_1**,T_2,T_5,**T_4,T_2,**T_5,**T_3,T_1**
total length = 21

Figure 5. The test sequence obtained from the test segments of Table 1 by the RCP technique

It is NP-hard to find the Rural Chinese Postman Tour of a general digraph [16]. Fortunately, a low-order polynomial-time algorithm can find the optimal tour if the bold edges form a weakly connected subgraph[2] [1]. If it is not weakly connected, a minimum number of fine edges are redrawn as bold edges until all the bold edges are weakly connected, and so that the bold edges can be weakly connected [10], in such a way that an

[2] A digraph is weakly connected if there is a path from any vertex to any other vertex disregarding the edge directions.

approximate tour can be found. If the Wp set of each state contains only an input sequence, the Wp set is in fact a UIO sequence. In such a case, the W digraph is in fact an RCP digraph of [1]. Because a Wp set can have more than one input sequences, bold edges of the W digraph are more likely to form a weakly connected subgraph than that of the RCP digraph. It has been proved in [1] that if the protocol has either the returning property (i.e., there is a transition from any state to the initial state) or the self-loop property (i.e, the FSM has at least one self-loop per state), the bold edges of the RCP digraph are weakly connected. Clearly, these two properties apply to the W digraph as well.

Formally, the process stated in this section is described by the following algorithm.

Algorithm 1.

Input: An FSM represented by a transition digraph G(V, E). A set of test segments constructed from the Wp sets.

Output: A test sequence

Step 1: /* Construct a W digraph G'(V', E') */
Copy G(V, E) into G'(V', E');
For each test segment Check(T_q) which starts from state J and ends at state K
add a bold edge labeled with (J, K; Check(T_q)) which starts from vertex J and ends at verex K to graph G'(V', E');
The edge cost is assigned as the length of test segment T_q;

Step 2: /* Graph Modification */

Check if all bold edges of G'(V', E') are weakly connected by the algorithm of [1];
If those bold edges are not weakly connected then bolden a minimum set of fine edges by the algorithm of [10];

Step 3: /* Test Sequence Generation */
Use the Rural Chinese Postman Algorithm of [1] to find the Rural Chinese Postman of G'(V, E');
Write input/output pairs of the tour into a test sequence.

4. Extending the technique to generate the synchronizable test sequence

A test sequence may encounter a synchronization problem when applying to the testing system of Figure 1. Consider the FSM of Figure 2. The test sequence [T_2, T_8] = [b/y, a/y] encounters a synchronization problem. In "b/y", the lower tester (L) sends "b" to the implementation (I) then L receives "y" from I. In "a", the upper tester (U) will send "a" to I. However, U does not know the exact time that L sends "b", so that it does not know the exact time to send "a" thereafter. By definition [3], a consecutive input/output pair "ij/oj, ij+1" encounters a synchronization problem if "ij+1" is sent from the tester that neither sends "ij" nor receives "oj". To avoid the synchronization problem, "b/y, a/y" is modified into "b/y, LtoU, a/y" where "LtoU" is an external synchronization operation which indicates that L should call U to send the next message. Similary, we need another external synchronization operation "UtoL". Such a call is usually established by the incovient telephone that is operated manually. As a result, we want to use these external synchronization operations as less as possible.

The optimization technique described in Section 3 is extended to generate the synchronizable test sequence, based on four steps. First, the transition digraph (Figure 2) is converted to the DuplexE digraph (Figure 6) [3], the path of which is used to generate the synchronizable test sequence. Each vertex J of Figure 2 is converted to vertices L_j and U_j of Figure 6. Each transition T_r = (J, K; i/o) of Figure 2 is converted to the edge (A_j, B_k; i/o) of Figure 6, if tester A sends "i" and tester B

either sends "i" or receives "o" [3]. In Figure 6, the dashed edges represent external synchronization operations. Any path of Figure 6 can be used to generated to generate the synchronizable test sequence. Any path of Figure 6 that includes no dashed edges can be used to generate the synchronizable test sequence which involves no external synchronization operations.

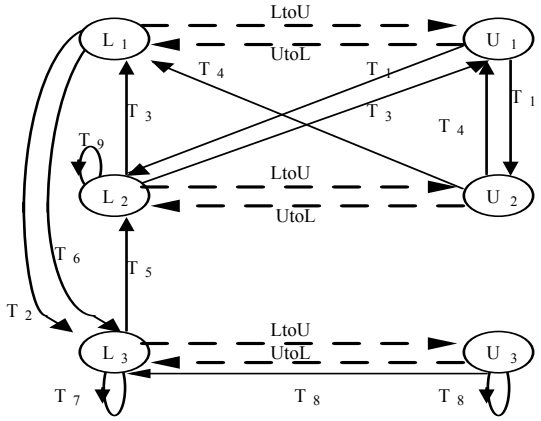


Figure 6. The DuplexE digraph constructed from the transition digraph G(V, E) of Figure 1

Second, synchronizable test segments are constructed from the Wp sets. Notice that many subsets of a W set can be the Wp set. Thus, we can explore all the possible choices to determine the Wp set which can be used to generate the synchronizable test segments. Moreover, if we cannot find the synchronizable test segments, we can enlarge the W set so that a state can have even more choices of Wp set; or we can simply construct a synchronizable test segment by introducing the external synchronization operations. For example, consider the W set {a, b} of the FSM in Figure 2. From this W set, we cannot find any Wp set that can construct the synchronizable test segments for checking transition T_2 and T_6. Thus, we enlarge the W set to be {a, b, cb} so that we determine Wp(3) = [c/v, b/z] = [T_5, T_3]. Hence, we construct the synchronizable test segments Check(T_2) = [T_2, T_5, T_3] , and Check(T_6) = [T_6, T_5, T_3].

Formally, the process stated in this section is described by the following algorithm.

Algorithm 2.

Input: An FSM represented by a transition digraph G(V, E). A W set. A set I which denotes the set of possible inputs of the FSM.

Output: a synchronizable test sequence

Step 1: /* Construct the DuplexE digraph */
Construct the duplexE digraph G"(V', E") from G(V, E) by the algorithm of [3]

Step 2: /* Find Synchronizable Test Segments */

For each state J
  Find all Wp sets of W for state J by the algorithm of [11];
Let Wp^J be the superset of all these Wp sets;
For each transition T_q which starts at state J and ends at state K.

```
k :=1;
Repeat
   Consider the kth Wp set of the
   superset Wp^J
   Let Wp(J) denote this WP set;
   If [Tq, Wp(J)] is synchronizable]
   synchronizable
   exit;
   k:= k+1;
Until all Wp sets are considered;
Check(Tq) = [Tq, Wp(J)];
```

Step 3: /* Construct the DuplexW digraph G*(V*, E*) */

For each test segment Check(T_q) which starts from state J and ends at state K
  add a bold edge labeled with (J, K; Check(T_q)) which starts from vertex J and ends at verex K to graph G"(V", E");The

edge cost is assigned as the length of test segment T_q;

Step 4: /* Sychronizable Test Sequence Generation */

Use the Selecting Chinese Postman Tour Algorithm of [5] to find the Selecting Chinese Postman Tour of G*(V*, E*);
Write input/output pairs of the tour into a test sequence.

Third, the DuplexE digraph is embedded with the synchronizable test segments so as to construct the DuplexW digraph (Figure 7). Consider a synchronizable test segment $Check(T_r)^s = [i_1/o_1, i_2/o_2, \ldots, i_p/o_p]$ which starts at state J and ends at state K. It is embedded as a bold edge (A_j, B_k; i_1/o_1, i_2/o_2, . . ., i_p/o_p) where "i_1" is sent by tester A, and either "i_p" is sent by tester B or "o_p" is sent to tester B. Moreover, we put the bold label $Check(T_r)^s$ on the edge, and each edge is assigned a cost according to the cost of operations that are associated with the edge. For example, consider $Check(T_5) = [T_5, T_3] = [c/v, b/z]$ which starts at state 3 and ends at state 1. It is embedded as two bold edges (L_3, L_1; c/v, b/z) and (L_3, U_1; c/v, b/z) because [c/v, b/z] starts with an input relating to tester L and [c/v, b/z, i] remains synchronizable for any input relating to either tester L or tester U.

Fourth, we find the Selecting Chinese Postman Tour of the DuplexW digraph so as to compute the test sequence. Notice that the edge that shares the same bold label represents the same test segment. For example, (L_2, $Check(T_3)^1$, L_2) and (L_2, $Check(T_3)^1$, U_2) represent the same test segment $Check(T_3)^1$. Thus, the Selecting Chinese Postman Tour is defined as a tour whether each bold label of the set {$Check(T_1)$, $Check(T_2)$, $Check(T_3)^1$, $Check(T_3)^2$, $Check(T_4)^1$, $Check(T_4)^2$, $Check(T_5)$, $Check(T_6)$} appears at least once. In [5], an algorithm is proposed for finding either the Selecting Chinese Postman Tour or an approximate tour of a digraph. In Figure 7, we find the tour: (L_1, L_1; $Check(T_2)$), (L_1, L_1; $Check(T_6)$), (L_1, L_3; T_2), (L_3, L_2; T_5), (L_2, L_2; $Check(T_3)^1$), (L_2, L_3; $Check(T_3)^2$), (L_3, U_1, $Check(T_5)$), (U_1, U_2; $Check(T_1)$), (U_1, U_2;T_1), (U_2, U_2; $Check(T_4)^1$), (U_2, L_3; $Check(T_4)^2$), (U_2, L_2; T_5), (L_2, L_1;T_3)), which is used to generate the synchronizable test sequence of Figure 8.

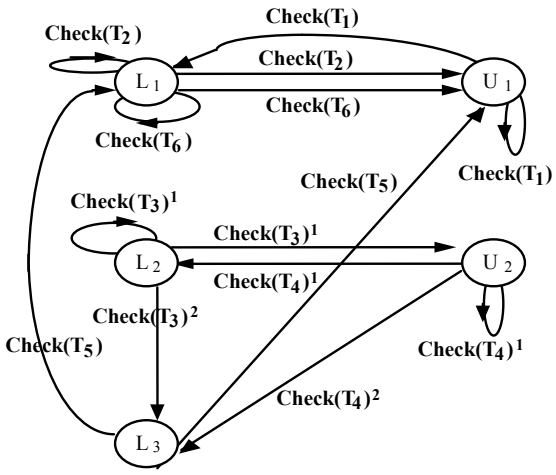


Figure 7. The DuplexW Digraph (edges and vertices of Figure 6 is a part of this digraph but is not shown here for clarity)

$T_2, T_5, T_3, T_6, T_5, T_3, T_2, T_5, T_3, T_1, T_3, T_2, T_5, T_3, T_1, T_3, T_1, T_4, T_1, T_4, T_2, T_5, T_3$

Total length = 23

Figure 8. The synchronizable test sequence for the transition digraph of Figure 1 obtained by an extension of the RCP technique

5. Conclusions

In this paper, we have proposed a technique to generate the test sequence for proto-

col conformance testing using the Wp sets and the RCP technique. An optimal test sequence can be obtained if the protocol satisfies either the returning property or the self-loop property. As stated in [1], many real protocol possess either property. However, if neither one is satisfied, a W digraph of the protocol can still be constructed to see if the bold edges are weakly connected so that an optimal test sequence can be obtained. We have also extended our technique to generate synchronizable test sequences. Other extensions are described as follows.

First, we can overlap test segments into a shorter test sequence. The technique for overlapping test segments based on the UIOv method has been widely studied [8] [18]. For the Wp method, a simple overlapping approach is performed by first checking if any pair of test segments can be overlapped and merging the two overlapped test segments into a new test segment. This merging process can be repeated k times so that at most k test segments can be overlapped. Those new test segments are then connected into a continuous test sequence by the RCP technique described in this paper. More complex overlapping techniques worth further study.

Second, we can apply the technique to a protocol modeled by other models. For example, for a protocol modeled by an Extended Finite State Machine (EFSM), a test sequence is executable if the test sequence is associated with parameter values which obey the rules defined in the EFSM. In [6], the Selecting digraph has been proposed to automatically generate the executable test sequence from an EFSM. It is expected that the current technique can be applied to the Selecting Digraph for generating an executable test sequence based on the Wp method.

References

[1] Aho, A. V., Dahbura, A. T., Lee, D., and Uyar, M. U. 1991. An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours. *IEEE Transactions on Communications,* 39, 11: 1604-1615.

[2] Ben-Ari, M. 1990. "*Principles of Concurrent and Distributed Programming".* Prentice-Hall. Englewood Cliffs. New Jersey. U. S. A.

[3] Chen, W. H. and Ural, H. 1995. Synchronizable test sequence based on multiple UIO sequences. *IEEE/ACM Transactions on Networking,* 3, 2: 152-157.

[4] Chen, W. H., Tang, C. Y., and Vuong, S. T. 1995. Improving the UIOv- method for protocol conformance testing. *Computer Communications,* 18, 9: 609-612.

[5] Chen, W. H. 1998. Test sequence generation from the protocol data portion based on the Selecting Chinese Postman Algorithm. *Information Processing Letters,* 65: 261-268.

[6] Chen, W. H. 2001. Executable test sequence for the protocol data flow property. *IFIP International Conference on Formal Techniques for Networked and Distributed Systems.* Cheju Island. Korea.

[7] Chow, T. S. 1989. Testing design modeled by finite state machines. *IEEE Transactions on Software Enginering,* 4: 178-186.

[8] Dssouli, R., Saleh, K., Aboulhamid, E., En-Nouaary, A., and Bourhfir C. 1999. Test development for communication protocols: towards automation. *Computer Networks,* 31, 17: 1835-1872.

[9] En-Nouaary, A., Dssouli, R., and Khendek, F. 2002. Timed Wp-method: testing real-time systems. *IEEE Transactions on Software Engineering,* 28, 11: 1130-1146.

[10] Eswardan, K. E. and Tarjan, R. E. 1976. Augmentation Problems. *SIAM Journal on Computing,* 5, 4: 653-665.

[11] Fujiwara, S., Bochmann, G. V., Khendek, F. Amalou, M., and Ghedamsi, A. 1991. Test selection based on finite state machine models. *IEEE Transactions on Software Engineering,* 17, 6: 591- 603.

[12] Gill, A. 1962. "*Introduction to the Theory of Finite State Machines"*. McGraw Hill. New York.

[13] Hsieh, E. P. 1971. Checking experiments for sequential machines. *IEEE Transactions on Computing,* 12, 10: 1024-1036.

[14] International Standard Organization 1989. OSI Conformance Testing Methodology and Framework. *ISO/IECJCT1/SC21 DIS 9646. Parts 4-5.*

[15] Kohavi, Z. 1978. *"Switching and Finite Automata Theory"*. MacGraw Hill. New York. U. S. A.

[16] Lee, D. and Yannakakis, M. 1996. Principles and methods of testing finite state machines. *Process of IEEE,* 84, 8: 1090-1123.

[17] Luo, G., Bochmann, G. V., and Petrenko,1994. Test selection based on communicating Nondeterministic Finite State Machines using a generalized Wp method. *IEEE Transactions on Software Engineering,* 12, 2: 120-132.

[18] Miller, R. E. and Paul, S. 1993. On the generation of minimal-length conformance tests for communication protocols. *IEEE/ACM Transactions on Networking,* 1, 1: 116-129.

[19] Naito, S. and Tsunoyama, M. 1981. Fault detection for sequential machines by transition tour. *Process of IEEE Fault Tolerant Computing Sympsium.*

[20] Sabnani, K. K. and Dahbura, A. T. 1988. A protocol test generation procedure. *Computer Networks and ISDN Systems,* 15, 4: 285-297.

[21] Sidhu, D. P. and Leung, T. K. 1988. Fault coverage of test methods. *Process of IEEE INFOCOM.*

[22] Tanebaum, A. S. 1988. "*Computer Network. Second Edition"*. Prentice-Hall. Englewood Cliffs. New Jersey. U.S.A.

[23] Vuong, S. T., Chan, W. Y. L., and Ito, M. T. 1989. The UIOv- method for protocol test sequence generation. *Process of IFIP International Workshop on Protocol Test Systems*. Berlin. Germany.