한국정보전자통신기술학회논문지(jkiiect)'17-08, Vol.10 No.4
논문 17-10-04-325 http://dx.doi.org/10.17661/jkiiect.2017.10.4.325

# Comparative Analysis of Protocol Test Sequence Generation Methods for Conformance Testing

Chul Kim*

# 적합성시험을 위한 프로토콜 시험항목 생성방법의 비교분석

김 철*

**Abstract** In this paper, a survey of test sequence generation methods for testing the conformance of a protocol implementation to its specification is presented. The best known methods proposed in the literature are called transition tour, distinguishing sequence, characterizing sequence, and unique input/output sequence. Also, several variants of the above methods are introduced. Applications of these methods to the finite state machine model are discussed. Then, comparative analysis of the methods is made in terms of test sequence length. Finally, conclusions are given as follows. The T-method produces the shortest test sequence, but it has the worst fault coverage. The W-method tends to produce excessively long test sequences even though its fault coverage is complete. The problem with the DS-method is that a distinguishing sequence may not exist. The UIO-method is more widely applicable, but it does not provide the same fault coverage as the DS-method.

**요 약** 본 논문은 프로토콜 구현물이 프로토콜의 사양에 대한 적합성을 시험하기 위한 시험항목 생성방법들에 대하여 비교분석 한다. 대표적인 방법들인 천이 순회, 구별 시퀀스, 특징화 시퀀스, 유일 입출력 시퀀스와 변형된 이들 방법들을 분석하고, 유한 상태 기계 모델에 적용한 위의 방법들의 시험항목 길이를 비교 및 분석 한다. 결론에서는 프로토콜 적합성시험을 위한 시험항목 생성방법들에 대한 핵심적이고 분석적인 이슈 사안들을 다음과 같이 제시한다. 천이 순회 방법은 최단의 시험 항목을 생성하지만 최악의 오류 검출 성능을 제공한다. 특징화 시퀀스 방법은 완벽한 오류 검출 성능을 제공하지만 상대적으로 최장의 시험 항목을 생성한다. 구별 시퀀스 방법의 문제점은 이 구별 시퀀스가 항상 존재하지는 않는다는 것이다. 유일 입출력 시퀀스 방법이 비교적 폭넓게 적용될 수 있지만 구별 시퀀스 방법과 동일한 오류 검출 성능을 제공하지 못한다는 문제점이 있다.

**Key Words** : conformance testing, finite state machine, implementation under test, lower tester, test sequence, upper tester

## 1. Introduction

Protocol implementations can be tested by considering either a single-layer, or a multiple-layer entity as a whole, by simulating the entities from the layers above and below the layer being considered, and by observing the behavior of the implementation under test(IUT). The testing activity for the purpose of checking the capabilities and behavior of the IUT against the conformance requirements of the protocol standard is defined as *conformance testing*[1].

When conformance testing is performed, an IUT is viewed as a black box. In Fig. 1, the lower interface and the upper interface of the IUT are controlled and observed indirectly by

*Corresponding Author : Department of Computer Science, Yongin University (chulkim@yongin.ac.kr)
Received July 24, 2017 Revised August 14, 2017 Accepted August 16, 2017

the lower tester(LT) and directly by the upper tester(UT), respectively. The sequence of input and the expected output pairs used for testing the implementation is known as a *test sequence*. Conformance testing, in some sense, is to check whether the behavioral input/output of the implementation of a protocol is as defined by the specification.

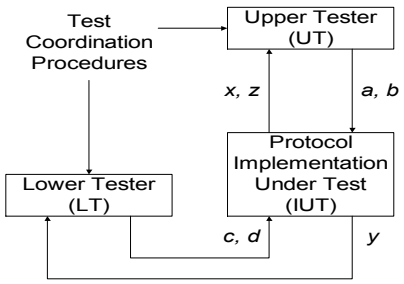


Fig. 1. An architecture for the protocol conforma nce testing

Systematic test sequence generation for communication protocols in conformance testing has been an active research area. Methods were developed to produce optimized test sequence for detecting faults in an IUT. Most of them are based on the finite state machine(FSM) model[2]. These techniques come in two classes: transition tour(*T-method*)[3] which simply includes all the transitions defined at least once; methods which require that FSM possess a special sequence/set of interactions such as distinguishing sequence (*DS-method*)[4], characterizing sequence (*W-method*)[5], and unique input/output sequence (*UIO-method*)[6]. In addition, a number of variants of the methods exist, mainly to optimize the length of the test sequence[7]. The examples are the $W_p$-*method*[8] that is a revision of the W-method and the $UIO_v$-*method*[9] that is a revision of the UIO-method. When these methods are actually applied to test a protocol implementation, they have usually a wide variety of the length of a test sequence generated. Therefore, it is important that which method should be chosen in testing a protocol where a cost of input/output interactions are taken into consideration.

The rest of the paper is organized as follows. The finite state machine models relating to the protocol conformance testing are described in Section 2. In Section 3, we discuss four major techniques for test sequence generation(T-, DS-, W-, and UIO-methods) and their variants, and their applications to the FSM model. Then, in Section 4, comparative analysis of the methods is made in terms of test sequence length. Finally, conclusions are given in Section 5.

## 2. Preliminaries

A protocol specification is typically modeled as a finite state machine(FSM)[2, 5]. A protocol can be specified as a deterministic FSM $M$ with a quintuple($S$, $I$, $O$, $f$, $g$), where:

$S$ = the set of states of $M$, including a special state $s_1$ called the initial state;
$I$ = the set of inputs, written $i$ in the following, $i_p \in I$;
$O$ = the set of outputs, written $o$ in the following, including the null output($nu$), $o_q \in O$;
$f$ = the next-state(transition) function, $S \times I \rightarrow S$;

$g$ = the output function, $S \times I \rightarrow O$.

For each state in the machine $M$, a reset transition is used to take the machine $M$ to its initial state. It takes the symbol "*ri*" as input and generates the symbol "*nu*" as output. An FSM $M$ is represented as a directed graph, $G = (V, E)$, where the set of vertices $V = \{v_1, \ldots, v_n\}$ represents the set of specified states $S = \{s_1, \ldots, s_n\}$ of $M$ and a directed edge $(T_m ; v_j , v_k ; i_p / o_q) \in E$ represents a transition from state $s_j$ to state $s_k$ in $M$. A non-negative, real-valued cost $Cost(T_{mn} ; v_j, v_k ; i_p/o_q)$ may be associated with the testing edge $(T_{mn} ; v_j , v_k ; i_p / o_q)$, where $Cost(T_{mn} ; v_j, v_k ; i_p/o_q)$ is the time required to realize the corresponding transition in $M$.

An example of a graph representation[10] and its transition table of an FSM $M$ are shown in Fig. 2 and Table 1, respectively. The initial state is assumed to be state $s_1$, reset edges are not shown for simplicity, and each edge is assumed to be of unit cost.

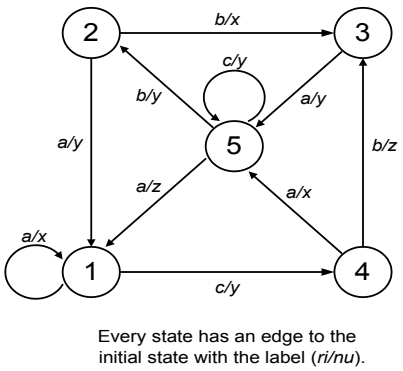


Fig. 2. A graph representation of a finite state machine $M$

Table 1. Transition table for $M$ in Fig. 2

| | Output | | | Next-State | | |
|---|---|---|---|---|---|---|
| Input / State | *a* | *b* | *c* | *a* | *b* | *c* |
| $s_1$ | *x* | *nu* | *y* | *1* | *1* | *4* |
| $s_2$ | *y* | *x* | *nu* | *1* | *3* | *2* |
| $s_3$ | *y* | *nu* | *nu* | *5* | *3* | *3* |
| $s_4$ | *x* | *z* | *nu* | *5* | *3* | *4* |
| $s_5$ | *z* | *y* | *y* | *1* | *2* | *5* |

## 3. The Practical Applications of Test Sequence Generation Methods

The methods to be described here for protocol conformance test sequence generation are based on a transition-level approach. The procedure of checking a transition from $s_j$ to $s_k$ with input/output $i_p/o_q$ consists of three basic steps:

(1) *Homing:* The FSM implementation is put into state $s_j$.

(2) *Output Verification:* Input $i_p$ is applied and the output is checked to verify that it is $o_q$, as expected.

(3) *State Recognition:* The new state of the FSM implementation is checked to verify that it is $s_k$, as expected.

The sequence of input/output pairs for testing edge $(T_{mn} ; v_j, v_k ; i_p/o_q)$ is denoted as $Test(T_{mn} ; v_j , v_k ; i_p / o_q)$ and consists of input/output $i_p/o_q$ followed by the sequence of input/output pairs necessary to realize the state recognition. The cost(or length) of each edge of $G$ is equal to the number of input/output pairs in its label. The cost(or length) of a path in $G$ is the sum of the costs(or lengths) of edges included in the path. A path with the minimum-cost(or

-length) among all paths from $v_j$ to $v_k$ is called a shortest path from $v_j$ to $v_k$.

### 3. 1 The T-Method

The *T-method*[3] is relatively simple, compared to the other three methods. A test sequence(called a *transition tour sequence*) can be generated by simply applying random inputs to an FSM $M$ until $M$ has traversed every transition at least once. The basic test process to test a state transition from $s_j$ to $s_k$ with input/output $i_p/o_q$ specified as ($T_{mn}$ ; $s_j$ , $s_k$ ; $i_p/o_q$) consists of two basic steps:

(1) The FSM implementation is put into state $s_j$;
(2) Input $i_p$ is applied and the output is checked to verify that it is $o_q$, as expected.

The algorithm finds a minimum-cost input sequence for exercising a given set of transitions of an FSM. The practical example of the *T-method* for the FSM $M$, shown in Fig. 2 and Table 1, is given in Table 2.

| *T-method* | |
|---|---|
| Assumptions | Strongly Connected, Partially Specified |
| Basic Testing Procedure of ($T_{mn}$; $s_j$, $s_k$; $i_p/o_q$) | (1) The FSM implementation is put into state $s_j$.<br>(2) Input $i_p$ is applied and the output is checked to verify that it is $o_q$, as expected. |
| Test Sequence | *a/x, c/y, b/z, a/y, a/z, c/y, a/x, c/y, b/y, b/x, a/y, b/y, a/y* |
| Total Cost of Tour | 13 (without reset edges) |

Table 2. The practical example of the *T-method*

### 3. 2 The DS-Method

In the *DS-method*[4], a distinguishing sequence(DS) is used for state identification. An input sequence $I_{ds} = i_1,\ldots,i_p$ is said to be a *distinguishing sequence* for an FSM $M$, if the output sequence produced by $M$ in response to $I_{ds}$ is distinct for each different starting state $s_j$. For each state transition defined as ($T_{mn}$ ; $s_j$ , $s_k$ ; $i_p/o_q$), the basic test process of checking the transition consists of three basic steps:

(1) The FSM implementation is put into state $s_j$, applying the synchronizing sequence and the transfer sequence which are explained in the below;
(2) Input $i_p$ is applied and the output is checked to verify that it is $o_q$, as expected;
(3) The distinguishing sequence for $s_k$ is applied and the new state after Step (2) is checked to verify that it is $s_k$, as expected.

In Step (1) of the above process, the synchronizing sequence is applied to the implementation followed by the transfer sequence to bring the implementation into state $s_j$. A *synchronizing sequence*[4] of an FSM $M$ is a sequence which takes $M$ to a specified final state, regardless of the output or the initial state. A *transfer sequence*[4], denoted as $T(s_1 , s_j)$, is defined as the shortest input sequence that takes an FSM $M$ from the initial state $s_1$ to an arbitrary state $s_j$. It is the minimum-cost input sequence of $M$ from $s_1$ to $s_j$. The practical example of the *DS-method* for the FSM $M$, shown in Fig. 2 and Table 1, is given in Table 3.

Table 3. The practical example of the *DS-method*

| *DS-method* | |
|---|---|
| Assumptions | Strongly Connected, Completely Specified, Minimal |
| Basic Testing Procedure of ($T_m$; $s_i$, $s_k$; $i_p/o_q$) | (1) The FSM implementation is put into state $s_i$, applying the synchronizing sequence and the transfer sequence.<br>(2) Input $i_p$ is applied and the output is checked to verify that it is $o_q$, as expected.<br>(3) The distinguishing sequence $I_{ds}$ for $s_k$ is applied and the new state after Step (2) is checked to verify that it is $s_k$, as expected. |
| *DS* Sequence | $I_{ds}$ = [*a, c, a*] |
| Synchronizing Sequence | [*ri*] |
| Transfer Sequences | $T(s_1, s_2)$ = [*c/y, a/x, b/y*], $T(s_1, s_3)$ = [*c/y, b/z*], $T(s_1, s_4)$ = [*c/y*], $T(s_1, s_5)$ = [*c/y, a/x*] |
| Test Sequence | *ri/nu a/x, a/x, c/y, a/x, ri/nu c/y, a/x, c/y, a/z, ri/nu c/y, a/x, b/y, a/y, a/x, c/y, a/x, ri/nu c/y, a/x, b/y, b/x, a/y, c/y, a/z, ri/nu c/y, a/x, b/y, a/x, c/y, a/x, ri/nu c/y, b/z, a/y, a/z, c/y, a/x, ri/nu c/y, b/z, a/x, c/y, a/x, ri/nu c/y, a/x, a/z, c/y, a/x, ri/nu c/y a/x, a/z, a/x, c/y, a/x, ri/nu c/y, a/x, c/y, a/z, c/y, a/x, ri/nu* |
| Total Cost of Tour | 66 |

### 3. 3 The W-Method

The *W-method*[5] is based upon deriving a *test tree* from an FSM $M$ that is defined to have the transitions as its branches and states as its nodes. The method involves the selection of two sets of input sequences:

(1) the *P-set* is a set of input sequences labeling the partial paths in the testing tree, including the empty sequence;
(2) the *W-set* is a set of input sequences differentiating each pair of states.

The W-set is called the *characterization set*. The test tree for an FSM $M$ is derived by the concatenation of its P- and W-sets. Specifically, elements of the P-set may be used as a set of preambles when concatenated with elements of the W-set to derive test sequences. Each sequence in the concatenation of $P$ and $W$ is applied starting with the initial state and followed by a transfer sequence back to the initial state to be ready for the next sequence. The practical example of the *W-method* for the FSM $M$, shown in Fig. 2 and Table 1, is given in Table 4.

Table 4. The practical example of the *W-method*

| *W-method* | |
|---|---|
| Assumptions | Strongly Connected, Minimal |
| Basic Testing Procedure of ($T_m$; $s_i$, $s_k$; $i_p/o_q$) | (1) The FSM implementation is put into state $s_i$, applying the synchronizing sequence and the transfer sequence.<br>(2) The *P*-set is applied and the set of corresponding output is checked, as expected.<br>(3) The *W*-set for $s_k$ is applied and the new state after Step (2) is checked to verify that it is $s_k$, as expected. |
| Characterization Set | *W*-set = {*a, a*} |
| Test Sequence | *ri/nu a/x, a/x, a/x, a/x, a/x, ri/nu a/x, a/x, c/y, a/x, a/z, ri/nu a/x, a/x, c/y, a/x, a/z, b/z, a/y, a/z, ri/nu a/x, a/x, c/y, a/x, a/z, b/z, a/y, a/z, a/y, a/z, a/x, ri/nu a/x, a/x, c/y, a/x, a/z, a/x, a/z, a/x, ri/nu a/x, a/x, c/y, a/x, a/z, a/x, a/z, a/x, b/y, a/y, a/x, ri/nu a/x, a/x, c/y, a/x, a/z, a/x, a/z, a/x, c/y, a/z, a/x, ri/nu a/x, a/x, c/y, a/x, a/z, a/x, a/z, a/x, a/z, a/x, a/x, ri/nu a/x, a/x, c/y, a/x, a/z, a/x, a/z, a/x, b/y, a/y, a/x, a/y, a/x, a/x, ri/nu a/x, a/x, c/y, a/x, a/z, a/x, a/z, a/x, b/y, a/y, a/x, b/x, a/y, a/z, ri/nu* |
| Total Cost of Tour | 109 |

A modified version of the *W-method*, called the $W_p$*-method* [8], was introduced so that a length of the test sequence is also reduced. The only difference between the two methods is that instead of using the complete

set $W$ to check each reached state $s_k$, only a subset of this set is used. This subset $W_k$ is called an identification set for state $s_k$. The resultant application of $W_p$-*method* is also given in Table 6.

Table 5. The practical example of the *UIO-method*

| *UIO-method* | |
|---|---|
| Assumptions | Partially Specified |
| Basic Testing Procedure of ($T_m$; $s_i$, $s_k$; $i_p/o_q$) | (1) The FSM implementation is put into state $s_i$.<br>(2) Input $i_p$ is applied and the output is checked to verify that it is $o_q$, as expected.<br>(3) The new state of the FSM implementation is checked to verify that it is $s_k$, as expected, by applying input sequence $UIO_k$ and checking that the resulting output sequence is that which is expected. |
| *UIO* Sequence, $UIO_k$ | $UIO_1$ = [c/y, a/x], $UIO_2$ = [b/x],<br>$UIO_3$ = [a/y, a/z], $UIO_4$ = [b/z],<br>$UIO_5$ = [b/y] |
| Test Sequence | ri/nu a/x, c/y, a/x, ri/nu c/y, b/z, ri/nu<br>c/y, b/z, a/y, a/z, ri/nu<br>c/y, b/z, a/y, b/y, ri/nu<br>c/y, a/x, b/y, ri/nu<br>c/y, a/x, a/z, c/y, a/x, ri/nu<br>c/y, a/x, b/y, b/x, ri/nu<br>c/y, a/x, b/y, a/y, c/y, a/x, ri/nu<br>c/y, a/x, b/y, b/x, a/y, a/z, ri/nu<br>c/y, a/x, c/y, b/y, ri/nu |
| Total Cost of Tour | 52 |

### 3. 4 The UIO-Method

The *UIO-method*[6] uses a set of unique input/output(UIO) sequences for state identification. A *UIO sequence* for a state of an FSM $M$ is an input/output behavior that is not exhibited by any other state of $M$. Formally, a UIO sequence for state $s_j$, denoted $UIO_j$, is a specified input sequence of minimum length $UIO_j = i_p, \ldots, i_l$ with initial state $s_j$ such that there is no $s_k \neq s_j$ for which the sequence of outputs produced by $UIO_k$ for initial state $s_k$ is identical to the sequence of outputs produced by $UIO_j$ for initial state $s_j$. The basic test process for realizing $Test(T_{mn}; v_j, v_k; i_p/o_q)$ for $(v_j, v_k; i_p/o_q) \in E$ is the following:

(1) The FSM implementation is put into state $s_j$;
(2) Input $i_p$ is applied and the output is checked to verify that it is $o_q$, as expected;
(3) The new state of the FSM implementation is checked to verify that it is $s_k$, as expected, by applying input sequence $UIO_k$ and checking that the resulting output sequence is that which is expected.

The practical example of the *UIO-method* for the FSM $M$, shown in Fig. 2 and Table 1, is given in Table 5.

A modified version of the *UIO-method*, called $UIO_v$-*method*[9], which contains a procedure for verifying the uniqueness of the *UIO* sequences(the *UIO*-verification procedure, denoted as $U_v$), thus detecting faults which were otherwise undetectable due to non-unique *UIO* sequences. The resultant application of $UIO_v$-*method* is also given in Table 6.

## 4. Comparative Analysis of the Test Sequence Generation Methods

In this section, we discuss the lengths of test sequences of the T-, DS-, W-, and UIO-methods discussed in Section 3 and

their variants. The nature of the different test methods implies certain relation between the lengths of the resulting test sequences. Table 6 shows a comparative test sequence length of major test generation methods. Length of the test sequence, in terms of number of input/output pairs, will determine the execution time for the test. On the average, the T-method will produce the shortest test sequence and the W-method the longest test sequence among the test sequence generation methods, while the DS- and UIO- methods generate test sequences of comparable lengths. The $W_p$-method shortens the length of W-method due to the partial W-set of the reduced length, while the $UIO_v$-method lengthens the sequence due to the UIO-verification procedure for every state. A test sequence for an FSM is said to be an optimum test sequence if it is a minimum-cost test sequence. The cost of a test sequence is usually considered as the number of inputs it contains. Thus, an optimum test sequence can also be stated as minimum length test sequence for the FSM.

Table 6. A comparative test sequence length of major test generation methods

| Methods | Test Sequence Length of FSM M in Fig. 2 | Assumptions about FSM |
|---|---|---|
| T-method | 13 input/output pairs | Strongly Connected, Partially Specified |
| DS-method | 66 input/output pairs | Strongly Connected, Completely Specified, Minimal |
| W-method | 109 input/output pairs | Strongly Connected, Minimal |
| $W_p$-method | 85 input/output pairs | Strongly Connected, Minimal |
| UIO-method | 52 input/output pairs | Partially Specified |
| $UIO_v$-method | 103 input/output pairs | Partially Specified, Minimal |

## 5. Conclusions

In conclusion, the T-method produces the shortest test sequence but it has the worst fault coverage. The W-method tends to produce excessively long test sequences even though its fault coverage is complete. Hence, the $W_p$-method which has the same fault coverage was introduced to shorten the length of W-method using the partial W-set of the reduced length. The DS- and UIO-methods produce comparable test sequences. The problem with the DS-method is that a distinguishing sequence may not exist. The UIO-method is more widely applicable, but it does not provide the same fault coverage as the DS-method. The $UIO_v$-method was henceforth introduced, and enjoys both complete fault coverage and wide applicability at the price of somewhat longer test sequences.

## REFERENCES

[1] D. Rayner, “OSI Conformance Testing,” in *Computer Networks and ISDN Systems*, vol. 14, no. 1, pp. 79-98, 1987.

[2] D. P. Sidhu and T. K. Leung, “Formal Methods for Protocol Testing: A Detailed Study,” in *IEEE Transactions on Software Engineering*, vol. 15, no. 4, pp. 413-426, 1989.

[3] G. Gonenc, “A Method for the Design of Fault Detection Experiments,” in *IEEE Transactions on Computers*, vol. C-19,

pp. 551-558, 1970.

[4] Z. Kohavi, *Switching and Finite Automata Theory*. New York: McGraw-Hill, 1978.

[5] T. S. Chow, "Testing Software Design Modeled by Finite-State Machines," in *IEEE Transactions on Software Engineering*, vol. 4, no. 3, pp. 178-187, 1978.

[6] K. K. Sabnani and A. T. Dahbura, "A Protocol Test Generation Procedure," in *Computer Networks and ISDN Systems*, vol. 15, no. 4, pp. 285-297, 1988.

[7] S. T. Chanson and J. Zhu, "A Unified Approach to Protocol Test Sequence Generation," in *Proc. IEEE INFOCOM '93*, 1993, pp. 106-114.

[8] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, "Test Selection Based on Finite State Models," in *IEEE Transactions on Software Engineering*, vol. 17, no. 6, pp. 591-603, 1991.

[9] S. T. Vuong and K. C. Ko, "A Novel Approach to Protocol Test Sequence Generation," in *Proc. IEEE GLOBECOM '90*, 1990, pp. 1880-1884.

[10] A. T. Dahbura, K. K. Sabnani, and M. U. Uyar, "Formal Methods for Generating Protocol Conformance Test Sequences," in *Proceedings of the IEEE,* vol. 78, no. 8, pp. 1317-1326, 1990.

## Author Biography

**Chul Kim** **[Regular member]**

- Feb. 1977 : Yonsei Univ., Electronics Engineering, B.S.
- Aug. 2000 : Yonsei Univ., Computer Science, Ph.D.
- Jul. 1981 ~ Jan. 1984 : Samsung LtD, Researcher
- Feb. 1984 ~ Jan. 1993 : IBM Korea, Engineer
- Mar. 1994 ~ current : Yongin Univ., Dept. of Computer Science, Professor

<Research Interests> Protocol Engineering, Computer Security