형식 기술 기법에 의한 LOTOS 프로토콜 적합성 시험

진 병 문[†] · 김 성 운[††] · 류 영 숙[††]

요 약

본 논문은 형식 기술 기법 중의 하나인 LOTOS 명세로부터 적합성 시험 시퀀스를 자동 생성하는 방법을 기술한다. 전체적인 시스템 구현을 위해 현재까지 연구된 여러가지 알고리즘들을 응용하여 적용하였고, 또 Rural Chinese Postman tour 개념을 개선한 후 적용하였다.

통신 프로토콜에 대한 LOTOS 명세로부터 CAESAR 도구의 Petri-net을 통한 시뮬레이션 기능을 이용하여 해당상태 천이 그래프를 생성하고, 얻어진 상태 천이 그래프에 대해 적합성 시험 시퀀스를 생성하기 위해 주어진 그래프의 각 상태에 대한 유일한 시퀀스인 UE sequence(Unique Event sequence)를 정의하였다. 또한 이러한 UE 시퀀스가 존재하지 않는 상태에 대해서는 부분 UE sequence(Partial UE sequence) 및 signature를 정의하였고, 또 이러한 특성 시퀀스에 대한 경험적 연구 결과를 정리하였다. 한편, 최적의 시험 시퀀스 생성을 위해 얻어진 특성 시퀀스들을 Rural Chinese Postman tour 개념에 적용하는 방법론에 대해서도 제시하였다. 또 생성된 적합성 시험 시퀀스의 오류 판단 영역 예측 방법 및 과정과 결과에 대해서도 기술하였고, 얻어진 시험 시퀀스를 표준 시험 표준기법인 TTCN으로의 변환 방법론도 제시하였다. 마지막으로 제안된 생성방법론에 대한 프로토타입은 실제 통신 프로토콜에 적용하기 위해 실질적인 많은 부분을 고려하면서 실행시험 스우트 생성을 위해 구현되었고, 이 프로토타입은 지능망이나 PCS 또는 ATM 프로토콜들을 위한 적합성 시험 목적으로 사용될 수 있다.

LOTOS Protocol Conformance Testing for Formal Description Specifications

Byoung Moon Chin[†] · Sung Un Kim[††] · Young Suk Ryu[††]

ABSTRACT

This paper presents an automated protocol conformance test sequence generation based on formal methods for LOTOS specifications by using and applying many existing related algorithms and technique, such as the testing framework, Rural Chinese Postman tour concepts. We use the state-transition graphs obtained from LOTOS specifications by means of the CAESAR tool. This tool compiles a specification written in LOTOS into an extended Petri net, from which a transition graph of a event finite-state machine(EvFSM) including data is generated. A new characterizing sequence(CS), called Unique Event sequence(UE sequence) is defined. An UE sequence for a state is a sequence of accepted gate events that is unique for this state. Some experiences about UE sequence, partial UE sequence and signature are also explained. These sequences are combined with the concept of the Rural Chinese Postman Tour to obtain an optimal test sequence which is a minimum cost tour of the

† 정 회 원:한국전자통신연구원 책임연구원
†† 정 회 원:부경대학교 정보통신공학과
논문접수:1996년 4월 10일, 심사완료:1997년 5월 19일

reference transition graph of the EvFSM. This paper also presents a fault coverage estimation experience of an automated method for optimized test sequences generation and the translation of the test sequence obtained by using our tool to TTCN notation are also given. A prototype of the proposed framework has been built with special attention to real application in order to generated the executable test cases in an automatic way. This formal method on conformance testing can be applied to the protocols related to IN, PCS and ATM for the purpose of verifying the correctness of implementation with respect to the given specification.

1. Introduction

The aim of protocol conformance testing is to check whether an implementation confirms to a given specification. Conformance testing is an important phase in the overall process of communication protocols development. Three main problems encountered in conformance testing are the automatization of the test generation, the fault coverage estimation of the selected test cases and the translation of selected test cases into the standardized test notation TTCN(Tree and Tabular Combined Notation).

The efficiency and fault coverage of conformance testing depend heavily on how test cases are selected. Automated test sequence generation is easier to adapt to specification changes, and also enables to generate more complete and consistent test sequences [1]. We have developed a protocol test sequence generation method form LOTOS specifications. LOTOS is a formal description technique for protocols and distributed systems [2], The purpose of our method is the automated generation of conformance test cases and optimized test sequences, from a formal specification of a protocol LOTOS. A signature for each state of the FSM (Finite State Machine) is defined by means of a Unique Event sequence(UE sequence) or a Partial Unique Event sequence(PUE sequence). the uniqueness of UE or PUE sequences in the implementation should first be verified, before applying an optimized test sequence based on the concept of the Chinese Postman Tour. This method has been implemented in a tool called TestGen-LOTOS, running on SUN4 (Sparce) workstation under UNIX.

On the other hand, the ability of the obtained test sequence to decide whether an implementation conforms to its specification relies upon the range of faults or errors that can be detected [3]. Ideally fault coverage should be the ratio between the number o nonequivalent EvFSMs which pass the test and the number of all possible EvFSMs which can be generated from the specification EvFSMs. In this paper, we estimate the fault coverage for the test sequence obtained by our software tool, using the Monte Carlo simulation.

Finally, we also present the methodology underlying the integration of the TTCN notation into the TestGen-LOTOS tool. Throughout this paper, we will use a simple example to illustrate both the application of the tool TestGen-LOTOS and the work accomplished on the translation of the test sequences into TTCN. Throughout this paper, our interest lies in a formal method for automated test case generation for communication protocol in LOTOS. We made use of several existing techniques to from an integrated framework for abstract test case generation from LOTOS specification. A prototype of the proposed framework has been built with special attention to real application in order to generate the executable test case in an automatic way. The main contribution ofthis paper is in giving some state-of-art ideas for the generation of test sequences from formal specifications and in the development of computer-aided test tools for practical real application. Theoretical reasearch has made significant advances in the generation of test sequences from formal specifications and in the development of computer-aided test tools. However, these methods and tools are not too industrially related and do not quite address the

problems facing testers in the industry. With the merits of test case generation from specifications based on Formal Description Techniques, the abstract test cases generated in TTCN language will be applied to the TTCN compiler in order to obtain the executable test cases which are relevant to industrial application.

The rest of the paper is organized as follows: section 2 describes some key concepts in protocol conformance testing and defines the concept of UE sequence. Some issues on the signature and PUE sequence used to generate test sequences form LOTOS specifications is then presented in section 3. A method for the automatic generation of test sequences form LOTOS specification is explained in section 4. The verification of uniquencess of the UE sequences in the implementation under test(IUT) is presented in section 5. Section 6 presents the fault coverage estimation of the test generation emthod and section 7 gives the translation of the obtained test sequences into TTCN notation. Finally, conclusions are given in section 8.

2. Preliminaries

In this section, we present some concepts and notations that we will use in the rest of this paper. Several methods have used the concept of UIO(Unique input Output) sequences [4], in roder to generate on or several test cases, when the protocol to be tested is given in the form of an Input/Output FSM. In fact, UIO sequences are used to test the resulting state after any transition of the can be formally described as T_{ij} $@UIO(s_j)$, where T_{ij} is the transition to be tested, that takes the specification I/OFSM from state s_i to state s_j, $UIO(s_j)$ is a UIO sequence for state s_j, and @ is the concatenation symbol. The UIO sequence is used to verify that the state reached by the transition to be tested is in fact s_j.

We use a kind of FSMs that we can derive from LOTOS specifications. In these FSMs, transitions are labelled only by the LOTOS rendez-vous corresponding to that transition. This is classical Moore machine studied in automata theory, that we will call Event Finite State Machine(EvFSM)[5].

Definition 1 : Event Finite State Machine
An Event Finite State Machine(EvFSM) is a 3-tuple $\sum=(S, E, \partial)$, where:
- $S=\{s_1,...,s_n\}$ is finite, non-empty set of states, with a distinguished element s_1 representing the initial state,
- $E=\{e_1,...,e_m\}$ is a finite set of events, and
- ∂. $SxE\rightarrow S$ is the state transition function.

Hence, transitions here are only labelled by an event $e \in E$. This kind of FSMs is closer to the graphs that we can generate from LOTOS specifications, since transitions are labelled by some gate events presented by LOTOS processes. We now introduce the notions of Accepted Event sequences and Unique Event secquences.

Definition 2: Accepted Event sequence
Let $\sum=(S, E, \partial)$ an EvFSM, and $s_i \in S$ a given state of $\sum$ An event $e_k \in E$ is said to be accepted for state s_i iff there exists a state $s_j \in S$ such that the EvFSM presents the following transition: $s_i - e_k \rightarrow s_j$. By extension, a sequence $e_{k1}\, e_{k2} \cdots e_{kl}$ of events is an accepted event sequence for state s_i iff there exists states $s_{i1}=s_{i1}$, $s_{i2} \cdots s_{il}$, $s_{i(l+1)}$ such that the EvFSM presents the following transitions:

$$s_{i1} - e_{k1} \rightarrow s_{i2} - e_{k2} \rightarrow s_{i3} \cdots e_{i1} - e_{k1} \rightarrow s_{i(l+1)} \quad (1)$$

Definition 3: Unique Event sequence
Let $\sum=(S, E, \partial)$ an EvFSM, and $s_i \in S$ a given state of $\sum$. A sequence $e_{k1}\, e_{k2} \cdots e_{kl}$ of events is a Unique Event sequence (UE sequence) for state s_i iff it is an accepted Event sequence for state s_i, and for no other state of the EvFSM.

Thus, we have adapted the concept of UIO sequences to the case of EvFSMs, which are generated

in our approach from LOTOS specifications. In this case, we will use UE sequences instead of UIO sequences, in order to test the resulting state after any transition of the EvFSM. Hence, each transition test subsequence will this time be formally described as T_{ij} @$UE(s_j)$, where T_{ij} is the transition to be tested, that takes the specification EvFSM from state s_i to state s_j, and $UE(s_j)$ is an UE sequence for state s_j. The test cases that we will generate should be checked at asynchronous Points of Control and Observation (PCOs). A test sequence will be composed of successive gate events, and the corresponding rendez-vous should be checked at the corresponding PCOs.

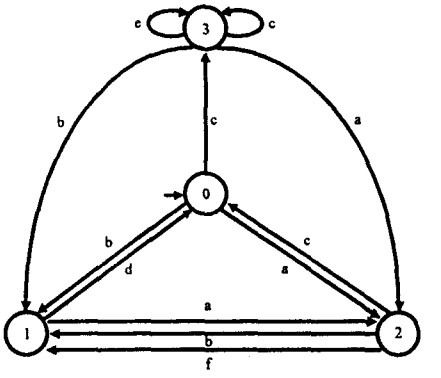
3. Signature and Partial Unique Event (PUE) sequence

A limitation of using UE sequences for test sequence generation is that some EvFSMs do not possess UE sequences for every state as provoked in the UIO approach. For a state s_i in a FSM $\sum$ which does not possess a UIO sequence, [4] defined a signature which distinguishes s_i from the remainin states one by one. We can apply this concept to the state-transition graphs obtained from LOTOS specifications.

Let the set of states for the obtained EvFSM $\sum$ be $\{s_1, s_2, ..., s_n\}$, where n is the number of states in $\sum$. Consider that state s_i does not possess an UE sequence. For each state s_j (where $j \neq i$), there exists an input event sequence IE(i, j) (origination at the state s_i) with length less than n which can distinguish s_j. The first part of the signature for s_i is an inptu event sequece IE(i, 1) which distinguishes s_i for s_1. LEt Ti (1) be a transfer sequence which brings the machine from the resultant state of IE(i, 1), denoted by s_k, back to the state s_i. This transfer sequence Ti(1) is the shortes path from s_k to s_i. Then, we apply the input event sequence IE(i, 2) which distinguishes s_i form s_2, and so on. A concatenation of sequence IE(s_i, s_j) @ Ti(j) for all states $s_i \neq s_j$ can be used as a state signature of s_i.

However, as explained in [4], a tight bound on the length of a minimum length UE sequence is still an open problem, a crude upper bound is $(n-1)n^2$ for a FSM having n states. This upper bound is meaning in practical applications. [4] demonstrates that if a state s_i has no UIO(UE) sequence of length less that $2n^2$, its signature would be used instead of the UIO(UE) sequence.

For all FSMs we have obtained from LOTOS specifications, the UE sequence for every state has less than 5 input event sequences if it exists [4] also remarked this point UIO sequence). We have also experienced that state signature for the state s_i having no UE sequence is always not unique for state s_i. Moreover, there is some cases in which state signature does not exist. For example, here is no UE sequence for state 0 in Fig. 1.



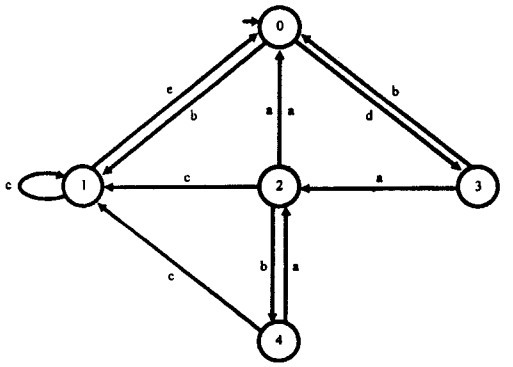
(Fig. 1) EvFSM 1

For obtaining state signature for state 0, we calculated each input event sequence IE(0, j), $1 \leq j \leq 3$ and corresponding transfer sequence T0(j) like as follow:

$$\underset{IE(0,1)}{b} @ \underset{T0(1)}{d} @ \underset{IE(0,2)}{a} @ \underset{T0(2)}{c} @ \underset{IE(0,3)}{cbd} \qquad (2)$$

Note that we have an input event sequence IE(0, 1) which distinguishes state 0 from state 1(IE(0, 2) for state 2 respectively). However, we have no IE(0, 3) with less than $n(n=4$ or forever) which can distinguish state 0 from state 3 (i.e., sequence "*cbd*" does not distinguish state 0 from state 3).

On the other hand, if a state has no UE sequence, instead of using state signature, we can use Partial Unique Event(PUE) sequences for the state. For a given state s_i, PUE sequences are a set of sequences originating at the state such that the input event behavior exhibited by this set of sequences in unique to the state. For example, input event sequences: (*c*, *ac*) in Fig. 2 constitute a set of PUE sequences for state 4.



(Fig. 2) EvFSM 2

No other state accepts both of these sequences. This set of PUE sequences {*c*, *ac*} is unique for state 4 because two PUE sequences (PUE 1: "c" and PUE 2: "ac") have disjoint exclusion sets {1, 2} and {0, 3} respectively and all other states (except state 4) of the given EvFSM are included in the exclusion sets. In this case, for testing the transition (T24: *b*) arriving at the state 4, we need two following test subsequences:

T24 @ PUE1(c),
T24 @ PUE2(ac) (3)

By testing these two test subsequences, it could be noted that this set of PUE sequences can be viewed as an UE sequence for state 4. However, in most practical situations, after generating the PUE sequences with their exclusion sets, one problem is selecting a minimal number of PUE sequences such that the intersection of their exclusion sets is empty, in the condition of all states of given EvFSM included in such exclusion sets (i.e., the states in each exclusion set have same input event behavior as corresponding PUE sequence). The optimal solution for selecting this minimal number of subsets of PUE sequences can be considered equivalent to the *set cover* problem that is known to be *NP*-complete [6]. Moreover, in our experience, the EvFSMs obained from real LOTOS specifications have many states and have no disjoint exclusion sets in most cases not like the above given example. For example, recall that the state 0 of Fig. 1 has no UE sequence. The state 0 of Fig. 1 has the following PUE sequences:

〈Table 1〉 PUE sequences for state 0

	PUE	Exclusion set
1	a	{1,3}
2	b	{2,3}
3	c	{2,3}
4	ce	{3}

When there are multiple PUE sequences with an identical minimal exclusion set, the shortest sequence is preferable as long as overall set is unique to its state(e.g., sequence PUE 1: "a" is preferable over sequence "ac" with exclusion set {1, 3}). Above table indicates that there are no disjoint exclusion sets for state 0. Consequently, state 0 has no set of PUE sequences which can be viewed as an UE sequence for the state.

In such case, we can generate a CS for state 0 by using existing PUE sequences. First of all, we must choose the shortest PUE sequence with small exclusion set among existing ones. It could be noted that sequence 4: "ce" is preferable over the sequence "a" with exclusion set {1, 2} (i.e., sequence 1: "a" would

only distinguish state 0 from state 3. It cannot distinguish state 0 from any of the other states). Consequently, we select a PUE "ce" with exclusion set {2} as optimal becasuse it is the shortest and has the smallest exclusion set elements.

for generating a CS for state 0, we need additional sequences for distinguishing given state from those states in the exclusion set. Considering the technique for generating state signature, state 0 has a following CS:

```
 ce  @  bd  @  cbd
----    -----   -------                    (4)
PUE  T0(2)  IE(0, 3)
```

Note that the sequence IE(0, 2) "cbd" does not distinguish state 0 from state 3 in the Fig. 1(i.e., there does not exist an input event sequence IE(0, 3) with less than $n(n=4$, or forever) which can distinguish state 0 from state 3). Even so, we can use this CS for test purposes. However, to attain high fault coverage of the UIOv method [7], we cannot verify uniqueness of this CS in the implementation under test becasuse this sequence is not unique in the specification (the reader should check that state 3 in Fig. 1 has the same input event behavior). This problem is also provoked in the case of state signature which in not unique in the specification. But the CS obtained by this approach is more short than state signature.

Recall that, in most practical situations in the automated test sequence generation environment, it is very difficult to find a set of PUE sequences which can be viewed as an UE sequence for a given state. We take another example for generating a CS a given state as shown in the above example. The state 4 of the Fig. 2 has a PUE "c" with exclusions set {1, 2} as optimal one because it is shortest and has smallest exclusion set (e.g., there exists some others PUE sequences; "ac" with exclusion set {0, 3} and "abc" with {0, 2, 3}). The state 4 has a following CS;

```
  c  @  eab  @  a  @  b  @  ac
------  -------  --------  ------  ------     (5)
PUE  T4(1)  IE(4, 1)  T4(2)  IE(4, 2)
```

In such cases, to attain high fault coverage of the UIOv method, we also have the same problem for verifying this CS in the implementation under test as provoked in the above example. For the practical application of UE(UIO) method for the EvFSMs which do not possess UE(UIO) sequences for every state, the following questions are of prime importance:

- Which do we choose for the role of UE(UIO) sequences?
- What is the corresponding fault coverage?

As shown in the above examples, we can choose three kinds of characterizing sequence for a state having no UE sequence: a set of PUE sequences, state signature and a CS obtained by using the optimal one among existing PUE sequences for the state. In most practical situations, it is very difficult to find a set of PUE sequences which can be viewed as a UE sequence for the state because the EvFSMs we have obtained from LOTOS specifications have many states and have no disjoint exclusion sets in most cases.

State signature is very long and moreover, there are some cases in which state signature does not exist. Therefore, we have chosen a CS obtained by using the optimal one among existing PUE sequences for the state. This sequence is more short than state signature and it is relatively easy to obtain this sequence in the automated test sequence generation environment. However, when this sequence is not unique in a given specification, we cannont verify the uniqueness of this CS in the implementation under test to attain high fault coverage of the UIOv method.

It is clear that, in general, if we have UE sequences for each state in the given EvFSM, we can obtain high fault coverage with small test sequence by the

verification of each UE sequence in the implementation under test. Next section is dedicated to the verification of each UE sequence (a set of PUE sequences) in the implementation under test to attain high fault coverage as done in UIOv method [7].

4. Generation of an optimal test sequence

This section presents the generation of the reference FSM from the LOTOS specifications and the generation of an optimal length test sequence from the obtained reference EvFSM(deterministic, minimal, and strongly connected). Thus, such a test sequence will take minimal time for an external tester to check the conformance between the observed behavior of the implementation and the expected behavior of the EvFSM specification.

4.1 Generation of the reference EvFSM

We first obtain the state-transition graph of a EvFSM from a LOTOS specification by means of CAESAR tool. CAESAR belongs to the CESAR family of verification tools for concurrent systems. This tool translates a LOTOS specification into an intermediate model, an extended Petri net, form which a state-transition graph is generated by using reachability analysis [8].

Some restrictions on LOTOS specifications are necessary in order to obtain state-transition graphs: no recursive process instantiation on the left side of operations such as parallel operator, disable operator and enable operator, and no recursive process instantiation through a parallel operator is allowed.

Frin tgese cibstraubed LOTOS specifications, translation process is accomplished by the following four phases:

Data phase. Abstract data types definitions are first analyzed and complied in order to create intermediate tables, which will be used in the simulation phase for the generation of a FSM.

Analysis phase. This phase aims at building an abstract tree from a LOTOS specification according to the lexical and syntactical definition of LOTOS. An obtained abstract syntax tree is then explored to check the static semantics of the LOTOS specification.

Generation phase. In this phase, the LOTOS abstract tree is expanded into a sub-LOTOS abstract tree, which describes a finite and statically fixed set of concurrent processes interacting through a fixed set of communicating gates. The behavior of each process is determined by sub-LOTOS algebraic terms and by the values of a finite, fixed set of state variables. Ths obtained sub-LOTOS abstract tree is hen translated into an extended Petri net. This Petri net is reduced by using a suitable set of reduction rules.

Simulation phase. This phase exhaustively explores all possible behaviors defined by the Petri net to produce a state-transition graph. It must be noted that in state-transition graphs produced by CAESAR, all data are taken into account in this phases, and represented by the gate events labeling the transitions of this EvFSM.

4.2 Generation of an optimal test sequence

From the obtained reference EvFSM, which is deterministic, minimal and strongly connected, an optimal (minimum-cost) test sequence is desired for effective testing.

In order to obtain this optimal test sequence, UE sequences, combined with an optimization technique based on the Rural Chinese Postman Tour allow the generation of a optimal test sequence [9]. The optimization echnique focuse on how to connect the test subsequence to minimize the length of overall test sequences. The reset capabilities are not required.

A Chinese Postman Tour of a directed, strongly connected graph $G=(V, E)$ is an optimal (minimum-cost) tour which contains every edge of E at least once. If G contains an Euler Tour, then this Euler

Tour is also a Chinese Postman tour [10].

For a subset E_C of E, a Rural Chinese Postman Tour is an optimal tour such that each edge in E_C is traversed at least once. This is a generalization of the Chinese Postman Tour, where $E_C = E$. Computing such a tour is known to be *NP*-complete in the most general case. However, graphs modeling protocols present some structural propoerties that allow us obtain a polynomial-time algorithm for generating an optimal test sequence. In the following, some relevant points for the implementation of this technique are discussed.

Generation of Unique Event sequences. We have adapted a procedure given in [3] for Unique Event sequences generation. In the case when a state does not possess any UE sequence, this state is identified by partial UE sequences.

Test subsequence generation. We caluculated TSSs using the formula T_{ij} @UE(s_j) for each specified edge (reset edge included) form vertex v_i to vertex v_j labelled with e_l, where e_l is a suitable event label associated with the transition in the reference graph $G=(V, E)$.

The UE sequence for state v_i is indicated y UE(v_i), and the last vertex of this UE sequence is denoted TAIL(UE(v_i)). A new directed graph $G'=(V', E')$ is defined, such that $V'=V$ and $E'=E \cup E_c$, where $E_c=\{(v_i, v_k; e_l)$ @UE(v_j): $(v_i, v_j; e_l) \in E$ and TAIL $(v_j))=v_k\}$.

Rural symmetric augmentation. The cost associated with an edge $((v_i, v_k; e_l)$ @UE(v_j)) $\in E_c$ is the sum of the cost of the edge labelled e_l and the costs of all edges in UE(v_j). In the following, we shall consider that all edges have cost 1.

In G', traversing an edge $((v_i, v_k; e_l)$ @UE(v_j)) $\in E_c$ corresponds to realizing the procedure for testing $(v_i, v_k; e_l)$ in G. Therefore, the minimum-cost test sequence that contains all test subsequences such that no two subsequences overlap corresponds to the minimum-cost tour of G', such that each edge in E_c is traversed at least once.

To find such a tour, we must replicate edges $(v_i, v_k; e_l) \in E$. This problem is reduced to that of finding a symmetric augmentation of G'.

From the directed graph $G'=(V', E')$, where $E'=E \cup E_c$, a symmetric directed graph $G''=(V'', E'')$ is constructed as follows. Let $V''=V'$. Each edge in E is included in E'' zero or more times and each edge in E_c is included in E'' at least once, such that the total cost of edges in E'' is minimized. Graph G'' is called a rural symmetric augmentation of G'.

To determine the number of replications of some edges $(v_i, v_k; e_l) \in E$ in G'', we used a minimum-cost augmentation [11], in which each augmentation is obtained by Dijkstra's shortest path algorithm.

Optimal test sequence. From the graph G'' (rural symmetric augmentation of G'), an optimal test sequence is an Euler tour (which starts from the initial state and also ends in the initial state), such that each edge in E_c is traversed at least once and the edges $(v_i, v_k; e_l) \in E$, which have non-zero replications, are traversed the given number of times. We used the algorithm of Edmonds that constructs such an Euler Tour in a linear time complexity [10].

The tour shown in Table 2 is an optimal test sequence for the EvFSM specification (shown in Fig. 4) which was obtained from the LOTOS specification of the alternating bit protocol (shown in Fig. 3)

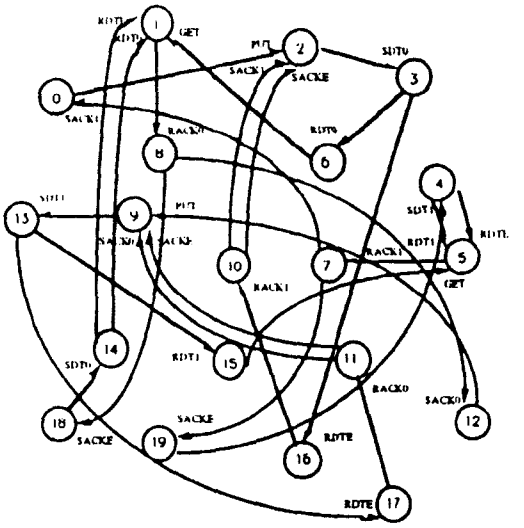
5. Verification of uniquencess of the UE sequences in the IUT

If the number of states in the IUT is the same as that in the specification, test sequences generated by the UIO approach were claimed to detect all faults in the IUT[4]. However, [7] showed that certain test sequences generated by the UIO approach are not

```
specification ALTERNATING_BIT_PROTOCOL [PUT, GET, SDT0, SDT1,
        RDT0, RDT1, RDTe, RACK0, RACK1, SACK0, SACK1, SACKe] :noexit behaviour
        (
                TRANSMITTER [PUT, SDT0, SDT1, SACK0, SACK1, SACKe]
                |||
                RECEIVER [GET, RDT0, RDT1, RDTe, RACK0, RACK1]
        )
        |[SDT0, SDT1, RDT0, RDT1, RDTe, RACK0, RACK1, SACK0, SACK1, SACKe]|
        (
                MEDIUM1 [SDT0, SDT1, RDT0, RDT1, RDTe]
                |||
                MEDIUM2 [RACK0, RACK1, SACK0, SACK1, SACKe]
        )

where


process MEDIUM1 [SDT0, SDT1, RDT0, RDT1, RDTe] : noexit :=
        SDT0;
                (
                RDT0;
                        MEDIUM1 [SDT0, SDT1, RDT0, RDT1, RDTe]
                []
                RDTe;
                        MEDIUM1 [SDT0, SDT1, RDT0, RDT1, RDTe]
                )
          []
          MEDIUM1 [SDT1, SDT0, RDT1, RDT0, RDTe]
endproc


process MEDIUM2 [RACK0, RACK1, SACK0, SACK1, SACKe] : noexit :=

          RACK0;

                (
                SACK0;
                        MEDIUM2 [RACK0, RACK1, SACK0, SACK1,
                                SACKe]
                []
                SACKe;
                        MEDIUM2 [RACK0, RACK1, SACK0, SACK1,
                                SACKe]
                )
          []

          MEDIUM2 [RACK1, RACK0, SACK1, SACK0, SACKe]

endproc
```

```
process TRANSMITTER [PUT, SDT0, SDT1, SACK0, SACK1, SACKe] : noexit :=

  PUT;
        TRANSMIT [PUT, SDT0, SDT1, SACK0, SACK1, SACKe]

  where
        process TRANSMIT [PUT, SDT0, SDT1, SACK0, SACK1, SACKe] :noexit :=

              SDT0;

                    (

                    SACK0;
                          TRANSMIT [PUT, SDT1, SDT0, SACK1, SACK0,
                             SACKe]
                    [ ]
                    SACK1;
                          TRANSMIT [PUT, SDT0, SDT1, SACK0, SACK1,
                             SACKe]
                    [ ]
                    SACKe;
                          TRANSMIT [PUT, SDT0, SDT1, SACK0, SACK1,
                             SACKe]

                    )

           endproc

  endproc

  process RECEIVER [GET, RDT0, RDT1, RDTe, RACK0, RACK1] : noexit :=

              RDT0;
                    GET;
                          RACK0;

                             RECEIVER [GET, RDT1, RDT0, RDTe, RACK1, RACK0]
              [ ]

              RDT1;
                    RACK1;

                             RECEIVER [GET, RDT0, RDT1, RDTe, RACK0, RACK1]
              [ ]

              RDTe;
                    RACK1:

                             RECEIVER [GET, RDT0, RDT1, RDTe, RACK0, RACK1]

endproc
endspec
```

(Fig. 3) LOTOS specification of the alternation bit protocol



(Fig. 4) The reference EvFSM obtained from the LOTOS specification of the alternating bit protocol

〈Table 2〉 An optimal test sequence for the EvFSM shown in Fig.4
(read from left to right and "ri" represents a special reset transition)

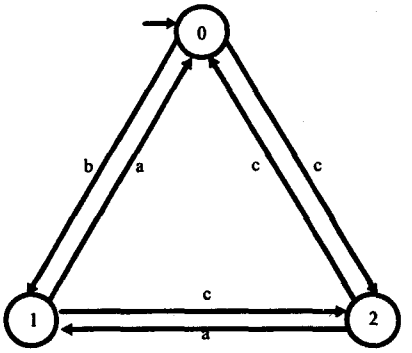
ri	PUT @ UE2	SACK1 @ UE2	SACKE @ UE2	SACKE	SDT0 @ UE3
RACK0 @ UE8	SDT1 @ UE13	RACK1 @ UE7	SDT0	RDTE	RACK1 @ UE10
RDT0 @ UE6	SACKE	SDT0 @ UE14	SACKE @ UE18	SACK0 @ UE12	RDTE
RACK0 @ UE11	RDTE @ UE17	RDT1 @ UE15	SACKE	SDT1 @ UE4	SACKE @ UE19
SACK1 @ UE0	RDTE @ UE16	RDT0	GET @ UE1	RDT0 @ UE1	RDTE @ UE1
RDTE	RACK0	SACK0	PUT @ UE9	SACKE @ UE9	SACK0 @ UE9
SACKE	SDT1	RDT1	GET @ UE5	RDTE @ UE5	RDT1 @ UE5
RDTE	RACK1	SACK1			

States	UE sequences
0	PUT SDT0
1	RACK0 SACKE SDT0
2	SDT0 RDTE RACK1
3	RDT0 GET
4	RDT1 RACK1
5	RACK1 SACKE SDT1
6	GET RACK0
7	SACK1 PUT
8	SACK0 PUT
9	SDT1 RDTE RACK0
10	SACK1 SDT0
11	SACK0 SDT1
12	PUT SDT1
13	RDT1 GET
14	RDT0 RACK0
15	GET RACK1
16	RACK1 SACKE SDT0
17	RACK0 SACKE SDT1
18	SDT0 RDTE RACK0
19	SDT1 RDTE RACK1

(Fig. 5) UE sequences for the EvFSM shown in Fig. 4.

capable of detecting all faults under the above given condition. Specifically, when UIOs and signatures are not unique in an IUT, they may not detect erroneous final states in the IUT. We have also experienced

such point in case of UE sequences. An optimal test sequence generated by our approach checks the IUT for all the transitions in the specification. However, if the IUT is faulty and an UE sequence from the specification is produced by more than one state in the IUT, then that UE sequence is not unique to the IUT and it is unable to identify a state in such IUT. An example EvFSM is shown in Fig. 6.



(Fig. 6) An EvFSM specification

We have generated the following test sequence and UE sequences by our tool for the given specification:

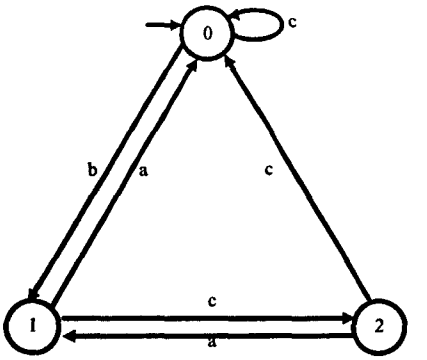
Recall that state 2 in Fig. 6 has two UE sequences "a" and "cb". If the first UE sequence "aa" was chosen for state 2, we have experienced that the test sequence obtained using this UE sequence by our tool could not pass the test with sucess in the fault IUT shown in Fig. 7.

〈Table 3〉 An optimal test sequence obtained for the EvFSM shown in Fig.6(read from left right)

ri	c @ UE2	c	a @ UE1
c	c @ UE0	c @ UE2	a @ UE0
a	b @ UE1	a	

〈Table 4〉 UE sequences for the EvFSM shown in Fig.6

States	UE sequences
0	b
1	ab
2	aa, cb



(Fig. 7) A faulty IUT

However, if we choose the second UE sequence "cb", the test sequence shown in Table 3(i.e., this sequence is obtained by using the UE sequence "cb") pass the test with success in the faulty IUT in Fig. 7. It arises from the fact that the uniquencess of UE sequence "cb" does not exist in the faulty ITU in Fig. 7(i.e., state 0 of the fault IUT has same UE sequence "cb"). IN fact, we use UE sequences on the assumption that each UE sequence is uniuqe in the IUT. However, they may not be. When we choose UE sequence according to a specification, we do not know whether they are actually unique in the IUT.

The problem of the UE(UIO) approach is that the uniqueness of the UI(UIO) sequences may not hold in a faulty IUT. This can be corrected by verifying UE (UIO) sequences to ensure they are indeed unique in an IUT prior to their use during testing [7]. For example, before applying the test sequence shown in Table 3, we verify the uniqueness of each UE sequence in the faulty IUT of Fig. 7. In such case, we can say that the IUT shown in Fig. 7 is faulty before applying the test sequence. More formally, the uniquencess verification of each UE sequence is carried out by checking the existence of the following sequences in an IUT:

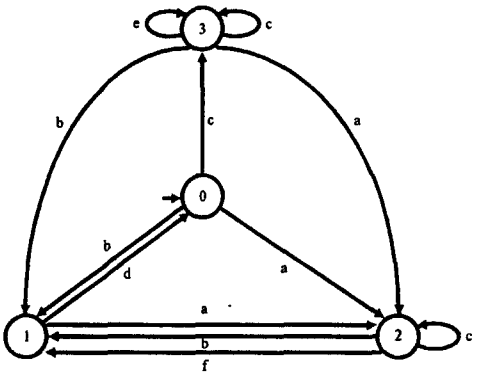
$$ri \; @ \sum_{k=0}^{n} \mathrm{UE}(k), \; (i=0)$$

$$\mathrm{UE}^{\sim}(\mathrm{si}) = \{ \qquad (6)$$

$$ri\ @\mathrm{SH}(s_o, s_i)\ @\sum_{k=0}^{n} \mathrm{UE}(k),\ (k \neq \mathrm{i},\ i \neq 0)$$

where ri represents reset transition which brings the machine to the start state s_o and $\mathrm{SH}(s_o, s_i)$ is the shortest path from s_o to s_i. Each state has to be checked for the absence of $(n-1)$ Ues in $\mathrm{UE}^{\sim}$. This requires a maximum$(n+(2n^2))(n-1)$ input event sequences for each state [7]. The presence of its own UEs is checked by our test sequence obtained by a tool based on the concept of the UE sequences combined with the concept of the Rural Chinese Postman Tour.

For the EvFSMs which do not possess UE sequences for every state, when we have a set of PUE sequences for the state having no UE sequences, we can verify the uniquencess of such set in the IUT. However, when state signature or a CS obtained using the optimal one among existing PUE sequences is not unique in the given specification, we have no method for verifying the uniquencess of these seqences in the IUT. For example, take a faulty IUT in Fig. 8 for the EvFSM specification shown in Fig. 1



(Fig. 8) A fault IUT for the EvFSM shown in Fig. 1

For testing purposes, even if we use, for state 0, either state signature ("bdaccbd") or a CS("cebdacbd") obtained by using the optimal one among existing PUE sequences, the faulty EvFSM given in figure 8 pass with success the test sequence obtained by approach. In such cases, we cannot verify the uniqueness of these sequences in the given IUT because these sequences are not unique in the specification.

In the automated test sequence generation, we have chosen two strategies. One hand, for the EvFSMs(we have generated from LOTOS specifications) which have UE sequences for each state, we add the uniqueness checking part $\mathrm{UE}^{\sim}$ to the sequence test obtained by our tool to attain high fault coverage as done UIOv method. On the other hand, for the FSMs which have no UE sequences for each state, we use only the test sequence generated by our tool for testing purposes. In such case, we may not detect erroneous final states in IUT. Moreover, it could be noted that DS sequences do not exist for the Mealy machine FSM in this situation. Consequently, it is an open problem.

6. Fault Coverage Estimation Experience

The ability of a test sequence to decide whether an IUT conforms to its specification heavily relies upon the range of faults or errors that it can detect. To evaluate the fault coverage of a given test sequence, we must generate the class of EvFSM's which are not equivalent to the specification EvFSM but will accept the test sequence. Ideally, fault coverage should be a ratio of number of nonequivalence EvFSMs which pass the test to the number of all possible EvFSMs which can be generated from the specification EvFSM [12].

However, estimation of fault coverage of a test sequence is a difficult task because the number of EvFSMs that must be examined is very large. For example, a specification EvFSM with n states and m inputs can have $(n)^{(mn)}$ possible implementations. It is obviously impossible to examine all of these machines. Instead, we sample the set of EvFSMs which are marginally different from the specification EvFSM. These EvFSMs are generated by changing the tail state of one or more transitions of the specification EvFSM

or by adding one or more possible transitions to the specificiation. We categorize these EvFSMs into the following classes:

Class 1: The tail state of one random transition in the specification EvFSM is changed to obtain this class of EvFSMs. Its new value is taken from an independent pseudo-random sequence to ensure fairness [3].

Class 2: The tail states of two random transitions in the specification EvFSM is changed to obtain this class of EvFSMs. Their new values are taken from independent pseudo-random sequence to ensure fairness.

Class 3: Anew random transition is added in the specification EvFSM to obtain this class of EvFSMs. The tail state, start and transition label are taken from an independent pseudo-random sequence to ensure fairness.

Class 4: Two new random transitions are added in the specification EvFSM to obtain this class of EvFSM to obtain this class of EvFSMs. The tail state, start state and transition label are taken from an independent pseudorandom sequence to ensure fairness.

To determine whether the EvFSMs pass the test sequence test (i.e., accept the given test sequence with success) actually conform to the specification EvFSM, the algorithm given in [13] is employed for this purpose.

We have taken two specification EvFSMs to estimate fault coverage of the test sequence generated by our approach, denoted UEop method, which combines UE sequences with the concept of Rural Chinese Postman Tour for obtaining an optimal test sequence. The first EvFSM is obtained from the LOTOS specification of the bit alternating protocol shown in Fig. 3 and another one is the specification EvFSM shown in Fig. 6 from which we can generate some faulty EvFSMs having more than one state which produce an UE sequence from the specification.

We have also defined two procedures of estimating fault coverage; the one is to estimate fault coverage without verification of the uniquencess of each UE sequence prior to the application of the test sequence and the other one is with verification of the uniqueness of each UE sequence in the IUT. First of all, we give a procedure of fault coverage estimation without UE verification as follow:

1) The specification EvFSM is read in.
2) The test sequence is generated by our test sequence generation tool.
3) Random EvFSMs which are marginally different from the specification EvFSM are generated as described in the above 4 classes.
4) The test sequence is applied to each of the machines generated in step 3) to check if they accept the given test sequence
5) EvFSMs that passed the test in step 4) are checked if they actually conform to the specification EvFSM.

For estimating the fault coverage of this sequence, 4 classes of random EvFSMs are constructed as described previously. For each class, one million random EvFSMs to examine are generated by our software tool.

As an example, take the EvFSM shown in Fig. 4 and its test sequence given in Table 2. The results of its fault coverage are given in Table 5. The entries 1 and 2 in Table 5 show that the test sequence generated by (UEop) method is able to detect one or more faults in tail state of transitions in this kind of EvFSM. Note that the EvFSM given in Fig. 4 is not fully specified and it is strongly connected. The entries 3 and 4 in Table 5 show that the given test sequence is able to use only for the weak conformance test (i.e., we constructed these classes of EvFSMs by adding one or two new transitions). In the following we will use the notation:

c : class of EvFSMs
n : number of generated EvFSMs
p : number of EvFSMs that pass the test sequence
e : number of EvFSMs equivalent to the reference EvFSM

〈Table 5〉 Estimation of the faults coverage for the test sequence given in Table 2

c	n	p	e
1	1000000	50082	50082
2	1000000	3963	3963
3	1000000	1000000	0
4	1000000	1000000	0

As another example, take the EvFSM shown in Fig. 6 and its test sequence given in Table 3. By the results of its fault coverage given in Table 6, the entries 1 and 2 in this table show that the test sequence generated by UEop method is not able to detect one or more faults in tail state of transitions in this EvFSM model. This arises from the fact that if an UE sequence from the specification EvFSM is produced by more than one state in the IUT, the test sequence obtained by UEop method can not detect one or more faults in tail state of transitions.

〈Table 6〉 Estimation of the faults coverage for the test sequence given in Table 3

c	n	p	e
1	1000000	490014	374364
2	1000000	298113	230123
3	1000000	1000000	0
4	1000000	1000000	0

We have already noted that this problem can be corrected by verifying UE sequences to ensure they are indeed unique in an IUT prior to their use during test. The following fault coverage estimation procedure is the same as presented above except UE˜ verification prior to the application of the test sequence.

1) The specification EvFSM is read in.
2) The test sequence is generated by our test sequence generation tool.
3) Random EvFSMs which are marginally different from the specification EvFSM are generated as described in the above 4 calsses.
4) The uniqueness of each UE sequence is checked in each of the EvFSMs generated in step 3). If each UE sequence is unique, we do step 5).
5) The test sequence is applied to each of the machines passed UE˜ verification in step 4) to check if they accept the given test sequence
6) EvFSMs that passed the test in step 5) are checked if they actually conform ot the specification EvFSM.

By this procedure, for the EvFSM shown in Fig. 6, the test sequence obtained by UEop method can datect perfectly one or more faults in tail state of transitions by UE˜ verification(i.e., the entries 1 and 2 in Table 7 demonstrate this point).

〈Table 7〉 Another estimation of the faults coverage for the test sequence given in Table 3

c	n	p	e
1	1000000	374364	374364
2	1000000	230123	230123
3	1000000	755024	0
4	1000000	575432	0

Based on the results in the tables above, we can conclude that the fault detection capabilities of test sequences for UEop method with UE˜ verification can perfectly detect one or more faults in tail state of transitions in an IUT. Also, this sequence is able to be used only for the weak conformance test. However, if we use state signature or a CS obtained by using the optimal one among existing PUE sequence instead of UE sequence, we can not guarantee this fault coverage.

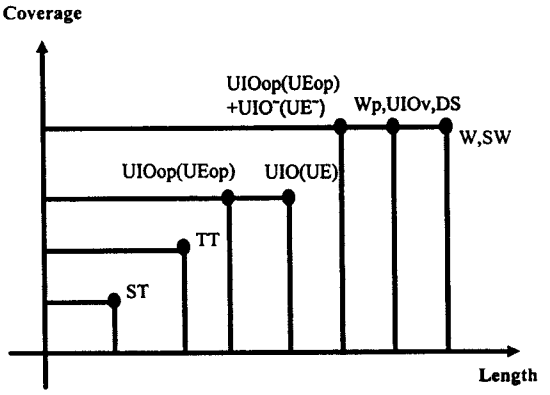
In conclusion, many test selection methods have

been developed for the case of the specification of the protocol being tested is given in the form of a FSM [14][7][1]. The test sequences derived by each of the above methods will detect any output error of the IUT. However, transfer errors (i.e., errors in the next state reached by a transition) will not always be found. The nature of the different test methods implies certain relations between the length of the resulting test sequence as shown in Fig. 9.

The figure also shows the relation for the theoretical fault coverage based on a model of output and transfer faults in the assumption of a limited number of states in the IUT.

By test sequence length and coverage relation between various methods, we analyze the results of each test sequence generation method as follows:

State-tour(ST) method: this method covers all states. However, it does not even check all transitions.



(Fig. 9) Relation between la length of the test sequences and its faults coverage

Transition-tour(TT) method: this method executes all transitions of the specification at least once, but does not make any effort to identify the target states.

UE(UIO) method: this method applies the UE (UIO) sequence to the target state of each transition; however, there is no guarantee that the UE(UIO) sequences have the identification power in the case of a faulty IUT which has more than one state having a same UE(UIO) sequence from specification.

UIOop(UEop) method [9]: by combining UIO(UE) sequences with the concept of the rural Chinese postman tour, it optimizes the resulting test sequence as against UE(UIO) method. However, this method has same problem in tail states identification as shown in UE(UIO) method.

UIOop(UEop) +UIO~(UE~) method: the test sequence obtained by this method perfectly detect one or more faults in tail state of transitions in an IUT. Its length is more short than the sequence obtained by UIOv method because the transition checking part is optimized as against UIOv method.

Wp, UIOv and DS methods: This method guarantees the same fault coverage as done in the UIOop (UEop) +UIO~(UE~). However, in the case of UIO sequence absence, if state signature or a CS obtained by PUE sequences is not unique in the IUT, we can not guarantee complete coverage in tail states errors. In such case, DS sequence does not exist.

W and SW methods: guarantees complete coverage even for the case where the number of states of IUT may be larger than that of the specification by fixed bound [14]. However, it has some difficulties for automated test sequence generation. In the case where the IUT would have a nondeterministic, it is impossible to have any guarantee for error detection.

7. TTCN translation

In this section, we present the methodology underlying the integration of the TTCN notation into the Test-Gen-LOTOS. Throughout this section, we will use a simple example to illustrate both the application of the tool TestGen-LOTOS and the work accomplished on the translation of the sequences into TTCN.

7.1 Test architecture

The IUT is modeled by an EvFSM whose transitions are labeled by LOTOS rendez-vous. It communicates with the environment in an asynchronous way.

In our example, this IUT should accept all the rendez-vous that compose the given test environment. The EvFSM that models the IUT changes from state to state, along with the establishment of the various rendez-vous, using the tested transitions at each rendez-vous. If the behaviour of the IUT is in conformance with its specification, then all the transitions of the EvFSM that model its behaviour can be accepted in the order specified by the given test sequence. We will now indicate the test entity that is used to communicate with the IUT.

The tester does communicate directly with the IUT. The tester's request for the acceptance of the gate event, one after the other one according to the given test sequence, will be transmitted to the communication points composed of an input queue and an output queue. If the gate event is accepted before expiration of given time-out, it indicates to the tester the acceptance of given gate event; otherwise, it notifies the failure of the acceptance.

7.2 Translation

Given the LOTOS specification of a protocol, the application of our conformance test generation technique results in:

- a set of test cases, one for each transition of the EvFSM representing the specification, or equivalently, an optimal test sequence which is the minimum cost combination of these test cases: and also
- UE sequences (if they exist, otherwise PUE sequences) to identify the states of the reference EvFSM.

The global test suite is composed of a set of test cases, each of them corresponding to the test of a single transition in the EvFSM. These test cases will in turn be decomposed into intended to bring the implementation into the object state necessary to test the given EvFSM transition. The body, test phase contains the rendez-vous that the tester wants the IUT to accept. Finally, the state identification phase contains the UE sequence that is needed to verify the final state of the tested transition.

Each transition in the reference EvFSM, labeled by a LOTOS rendez-vous expression will correspond to one test case in TTCN. The Preamble, made up of a sequence of LOTOS rendez-vous, will be specified by a succession of attached test steps and with as many steps as rendez-vous in the Preamble subsequence. The Body phase of a test case is also composed of test steps. Finally, the State Identification phase, corresponding to a UE sequence, will also be a succession of test steps.

Semantic incompatibility between TTCN and LOTOS has lead us to adopt the following description to specify a test of a LOTOS rendez-vous in TTCN: a test step is parameterized by means of the gate where the rendez-vous occurs and by a list of

Test Step Dynamic Behavior			
Test Step Name : Test_accepted_Input (a_gate : input) Group- Objective : Test whether an input is accepted or not Default : Comments:			
Nr	Behaviour	Constrains reference	Verdict
1	a_gate! test_input		
2	a_gate? input_accepted	input_accepted	pass
3	a_gate? input_rejected	input_rejected	fail

data exchanged during the rendez-vous. The structure of this test step is fixed, which means that it is independent from any input test sequence. It has following form:

Line 1: To test the acceptance of a LOTOS input through the gate a_gate, the tester sends through a PCO, which has the same name as the gate, a message test_input to indicate the beginning of the test of the input.
Line 2: If the tester receives an answer (indicating no time-out) from the IUT indicating the acceptance of the input proposed in line 1, the test ends successfully (PASS), which means that the input taken.
Line 3: On the contrary, if the tester receives an answer from the IUT indicating the refusal of the acceptance (indicating time-out), this means that the input proposed did not be accepted during the imparted time interval. The test then ends in failure(FAIL).

This test step will be called through an attachment mechanism, allowing a compact description of the test tree. Note that it is not necessary to continue the execution of a given test after any FAIL terminal branch. If such a test branch is executed, it means that the test ends in failure, so that it becomes useless to attach the next test step to it.

The variable TTCN tables are composed of a set of tests specific to each sequence to be translated, they are the followings:

- Test Case Dynamic Behaviour tabels
- Test Step Dynamic Behaviour tables, which describe the Preamble (the Postamble) and Identification phases.

The following table gives an example of a translation.

Note that, for the sake of simplicity, we decided

LOTOS sequence	TTCN Test Case	Preamble & Postamble Test Step
PUT @ UE2	Test_case1: +Test_accepted_input(put) +Check_UE2	
SACK1 @ UE2	Test_case2: +Test_accepted_input(SACK1) +Check_UE2	
SACKE @ UE2	Test_case3: +Test_accepted_input(SACKE) +Check_UE2	
SACKE SDT0 @ UE3	Test_case4: +Preamble_1 +Test_accepted_input(SDT0) +Check_UE3	Preamble_1: +Test_accepted_input(SDT0)
• • •		
RDT1 @ UE5 RDT2 RACK1 SACK1	Test_case_28: +Test_accepted_input(RDT1) +Check_UE5 +Postamble_1 +Postamble_2 +Postamble_3	Postamble_1: +Test_accepted_input(RDTE) Postamble_2: +Test_accepted_input(RACK1) Postamble_3: +Test_accepted_input(SACK1)

not to give the TTCN translation of the subsequences corresponding to the "reset" transitions in the implementation. The given test sequence (first column) is meant to test an IUT that implements the Alternating Bit Protocol, given as an example in Table 4 of Section 5. The column TTCN test case shows only the Behaviour Description part of the Test Case Dynamic Behaviour table. The names of the test cases are prefixed with test_case followed by the number of the current case generated. The names of the test steps used as Preamble(Postamble) are prefixed with Preamble_(Postamble_) followed by the number of the test case concerned.

The test steps called Check_UE ⟨state_number⟩ are generated separately by our tool. The following table presents the UE sequences for identification of the twenty states of the EvFSM modelling the Alternating Bit Protocol.

UE name	LOTOS Sequence	TTCN Test Step
UE0	PUT SDT0	Check_UE0: +Test_accepted_input(PUT) +Test_accepted_input(SDT0)
UE1	RACK0 SACKE SDT0	Check_UE1: +Test_accepted_input(RAC K0) +Test_accepted_input(SACKE) +Test_accepted_input(SDT0)
...		
UE18	SDT0 RDTE RACK0	Check_UE18: +Test_accepted_input(SDT0) +Test_accepted_input(RDTE) +Test_accepted_input(RACK0)
UE19	SDT1 RDTE RACK1	Check_UE19: +Test_accepted_input(SDT1) +Test_accepted_input(RDT0) +Test_accepted_input(RACK1)

8. Conclusions

In this paper, we have presented an automated method for generating an optimal test sequence from a LOTOS specification. We have introduced the concept of Unique Event sequence, which is well suited to the FSM obtained from a LOTOS specfiication. This concept is then combined with optimization echnique (without reset) based on the Rural Chinese Postman tour in order to obtain an optimal test sequence.

For the practical application of the UE method for the EvFSMs which do not possess UE sequences for every state, in the automated test sequence generation environment, we proposed and analyzed three kinds of characterizing sequence: a set of PUE sequences, state signature and a CS obtained by using the optimal one among existing PUE sequences for the state. It is experienced that, if we have UE sequences for each state in the given EvFSM, we can obtain high fault coverage with small test sequence by the verification of each UE sequence in the implementation under test.

We have also estimated the fault coverage for the test sequence obtained by our software tool, based on certains assumptions:

- For the detection of transfer errors, it is assumed that the number of states of the implementation is equal to the number of states of the specification;
- We have assumed that the implementation under test are deterministic machines. In the case where the implementation is nodeterministic, it is impossible to have any guarantee for error detection.

By the estimation results obtained, the test sequence obtained by the method "UEop+UE^{-}" perfectly detect on or more faults in tail state of transitions in an IUT. Its length is more short than the sequence obtained by UIOv method because the transition checking part is optimized as against UIOv method.

Finally, we presented the methodology underlyign the integration of the TTCN notation into the TestGen-LOTOS tool. We used simple exaple to illustrate both the application of the tool TestGen-

LOTOS and the work accomplished on the translation of the test sequences obtained into TTCN.

These techniques have been implemented in a tool called TestGen-LOTOS. Thus, an optimal test sequence of minimum cost can be automatically generated from LOTOS specifications. These techniques have been applied to the several protocols and services [15-16]. And also, it is started to try to apply these techniques, for conformance testing, on the protocols related to IN, PCS, and ATM by several R&D organizations such as AT&T Bell Lab., CNET(FRANCE),and etc.

The formal method on conformance testing described in this paper can be applied to all kinds of protocols related to IN, PCS, and ATM for the purpose of verifying the correctness of implementation with respect to the given specification.

In practice, test sequence is manually generated by extracting test cases from natural language specification. These are largely based on the experience of tests who are often able to uncover bugs in the implementation. This paper described a research result on automatic generation of abstract test cases from a reference EvFSM which represents communication protocol behaviors. With the merits of test case generation from specifications based on Formal Description Techniques, the abstract test cases generated in TTCN language will be applied to the TTCN compiler in order to obtain the executable test cases which are relevant to the industrial application. Further stdy topics related to the generation of test sequences from FDT are the following areas:

- Test sequences generation from nondeterministic EvFSM
- Resolution of minimization problems resulting from the state space explosion of EvFSM
- Application of the proposed approach to the generation of ETS in a real environment of conformance testing

REFERENCES

[1] M.S.Chen, Y.Choi, A.Kershenhaum, Approaches Utilizing Segment Overlab to Minimize Test Sequence, 10th International Symposium on Protocol Specification, Testing and Verification, Ottawa Canada, June 1990.

[2] ISO-LOTOS-A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, International Stand 8807, International Organization for Standardization Information Processing Systems-Open Systems Interconnection, Geneva, September 1988.

[3] A.Dahbura and K.Sabnani, An Experience in Estimating Fault Coverage of a Protocol Test, in Proc. IEEE INFOCOM'88, pp.71-79, 1988.

[4] K.Sabnani and A. Dahbura, A Protocol Test Generation Procoedure, Computer Networks and ISDN Systems, Vol. 15, No 4, pp.285-297, 1988.

[5] Z.Kohavi, Switching and Finite Automata Theory, New York, Mc Graw-Hill, 1978.

[6] W.Chun and P.D.Amer, Improvements on UIO Sequence Generation and Partial UIO Sequences, 12th International Symposium on Protocol Specification, Testing and Verification, Lake Buena, Florida, USA, 1992.

[7] W.Y.L.Chan, S.T.Yuong, and M.R.Ito, An Improved Protocol test Generation Procedure based on UIOs, SIGCOM'89 Symposium: Communication Architecture and Protocols in Computer Comm. Review 19(4), pp.283-294, September, 1989.

[8] H.Garavel and J.Sifakis, Compliation and Verification of LOTOS Specifications, in L.Logrippo, R.L.Probert and H.Ural, Editors, Proceedings of the 10th International Symposium on Protocol Specification, Testing,and Verification, IFIP, North Holland, Amsterdam, 1990.

[9] A.V.Aho, A.Dahbura, D.Lee and M.U.Uyar, An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours, in S.A. ggarwal Editor, 8th International Symposium on

Protocol Specification, Testing and Verification, pp.75-86, North Holland, Amsterdam, 1988.

[10] J.Edmons and E.L.Johnson, Matching, Euler Tours and the Chinese Postman's Tour, Mathematical Programming, vol.5, pp.88-124, 1973.

[11] R.E.Tarjan, Data structures and Network Algorithms, Philadelphia, PA, Society for Industrial and Applied Mathematics, 1983.

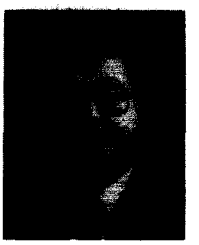
[12] D.Sidhu and T.Leung, Fault Coverage of a Protocol Test Methods, in Proc. IEEE INFOCOM'88, pp.80-85, 1988.

[13] A.V.Aho, J.E.Hopcroft, and J.D.Ullman, The Desigh and Analysis of Computer Algorithmes, Addison-Wesley, Reading, Mass. 1974.

[14] S.Fujiwara, G.Bochmann, F.Khendek, M. Amalou, and A.Ghedamsi, Test Selection Based on Finite State Models, IEEE Trans. on Soft. Eng., Vol.17, No.6, pp 591-602, June 1991.

[15] A.R.Cavalli, S.U.Kim, and P.Maigron, Automated Protocol Conformance Test Generation Based on Formal Methods for LOTOS Specifications, Protocol Test Systems V(C-11), G.V.Bochmann, R. Dssouli and A.Das editors, Elsevier Science Publishers, North Holland, Amsterdam, 1992.

[16] A.R.Cavalli, S.U.Kim, and P.Maigron, Improving Conformance Testing for LOTOS, FROTE'93, Boston, USA, pp.381-384, October 1993.



진 병 문

1976년 서울대학교 공과대학 전기공학과(학사)

1983년 서울대학교 공과대학 컴퓨터공학과(석사)

1996년 한국과학기술원 전산학과(박사)

1980년~현재 한국전자통신연구원(책임연구원, 실장)

관심분야: 프로토콜 시험, 컴퓨터 네트워크, 프로토톨 엔지니어링

김 성 운

1982년 경북대학교 전자공학과(공학사)

1990년 프랑스 국립파리 7 대학교 정보공학과(공학석사)

1993년 프랑스 국립파리 7 대학교 정보공학과(공학박사)

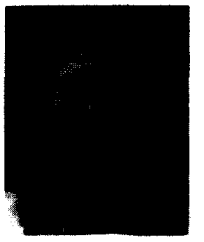
1982년~1985년 한국전자통신연구소 데이터통신연구실(연구원)

1986년~1995년 한국통신 연구개발원(선임연구원, 실장)

1990년~1993년 프랑스 정기통신기술연구소(초빙연구원)

1995년~현재 부경대학교 정보통신공학과(교수)

관심분야: 프로토콜 엔지니어링, 데이터 통신 통신프로토콜시험, 컴퓨터네트워크



류 영 숙

1977년 부경대학교 정보통신공학과(공학사)

1977년 부경대학교 대학원 정보통신공학과(석사과정)