Automated Test Generation from Specifications Based on Formal Description Techniques

Byoung-Moon Chin, Young-Han Choe, Sung-Un Kim, and Jae-Il Jung

CONTENTS

ABSTRACT

This paper describes a research result on automatic generation of abstract test cases from formal specifications by applying many related algorithms and techniques such as the testing framework, rural Chinese postman tour and unique input output sequence concepts. In addition, an efficient algorithm for verifying the strong connectivity of the reference finite state machine and the concept of unique event sequence are explained. We made use of several techniques to form an integrated framework for abstract test case generation from LOTOS and SDL specifications. A prototype of the proposed framework has been built with special attention to real protocol in order to generate the executable test cases in an automatic way. The abstract test cases in tree and tabular combined notation (TTCN) language will be applied to the TTCN compiler in order to obtain the executable test cases which are relevant to the industrial application.

I. INTRODUCTION

The primary objectives of standardization are to allow systems developed by different vendors to work together and to exchange and handle information successfully. In recent years, conformance testing, the assessment of a product with its specification, has been an important issue in software engineering [1]-[3], in particular for the digital communication world where most of the specifications are standardized internationally.

In order to guarantee world-wide compatibility in conformance testing, the standardization effort for this area has been conducted as a joint project of the standardization organizations, ISO and ITU-T. The effort resulted in a standard IS9646 or X.290 series [4], called "Conformance Testing Methodology and Framework (CTMF)."

CTMF is the general framework which is applicable to the wide range of protocol specifications and products. Recently, in the communications world, more and more specifications are being written in formal description technique (FDT). This makes it necessary to define the meaning of conformance in this formal context. The standardization efforts are being made under the title of "Formal Methods on Conformance Testing (FMCT)" [5].

The motivation of this paper is to develop automated tools for test generation from FDT specifications. There are FDT specifications for broadband integrated service digital network (BISDN), global system for mobile Communication (GSM), personal communication service (PCS) and other protocols in specification description language (SDL) and language of temporal ordering specification (LOTOS).

Some methods derive test cases directly from the FDT specification. Other methods consist of a transformation from the FDT to intermediate model (e.g. I/OFSM or labeled transition system (LTS)) and subsequent test generation from those models [6]-[8]. This transformation may preserve all information in the FDT specification, resulting in an intermediate model equivalent to the starting FDT description.

Research on methods for test generation from I/OFSM has received much attention in relation with conformance testing of communication protocols [9]-[10]. Automated test sequence generation is easier to adapt to specification changes, and also it enables the generation of more complete and consistent test sequences. The efficiency and fault coverage of conformance testing depend heavily on how test cases are selected.

Throughout this paper, our interest lies in a formal method for automated test case generation. We made use of several existing techniques to form an integrated framework for abstract test case generation from LOTOS and SDL specifications. A prototype of the proposed framework has been built with special attention to real application in order to generate the executable test case in

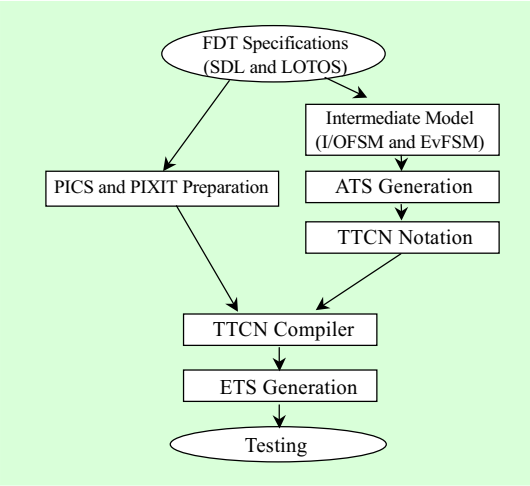


Fig. 1. Testing procedure.

an automatic way as the procedure shown in Fig. 1.

The abstract test cases generated in TTCN language are applied to the TTCN compiler in order to obtain the executable test cases which are relevant to the industrial application. This work includes

- Derivation of a theory of conformance relation for the LTS and I/OFSM
- Establishment of a framework leading to a formal testing methodology to assess the conformance of an IUT to the behaviors specified in an I/OFSM
- Presentation of a methodology for automatic test generation from FDT specifications
- Estimation of the fault coverage for the obtained test sequence
- Case study for the proposed method.

To realize this approach, we use an event finite state machine (EvFSM) of the state transition graphs obtained from LOTOS specifications and the input/output finite state machine (I/OFSM) generated from SDL specifications. Two characterizing sequences, the UIO sequence and the UE sequence are applied. These sequences are combined with the concept of the rural Chinese postman tour, without reliable reset capability, to obtain an optimal test sequence.

To develop this method, first of all, we derive a theory about conformance relation for the LTS and I/OFSM by adapting the related test theories developed by Brinksma [11] and Tretmans [12]-[13]. Then we establish a framework leading to a formal testing methodology by minor adaption of existing results of Petrenko and Bochmann [14], which can be used to assess the conformance of an implementation under test (IUT) to the behaviors specified in an I/OFSM equivalent to corresponding FDT specifications. With this framework, we present a methodology for automatic test generation from FDT specifications by applying many existing related algorithms and techniques by Aho *et al.* [15]. Finally, we also simulates a fault coverage of the proposed automated method for optimized test sequence generation by modifying the technique proposed by Sidhu and Leung [16].

This paper is organized as follows: Section II defines a theory about the conformance relation of the LTS and I/OFSM. Section III describes the establishment of

a framework leading to a formal testing methodology which can be adapted to assess the conformance of an IUT to the behavior specified in an I/OFSM specification.

Section IV gives the description of the method to generate test sequences from an intermediate model (reference FSM) that is equivalent to the corresponding FDT (SDL or LOTOS) specifications. Section V presents the fault coverage estimation of the proposed test generation method. Case study for the proposed method is explained in Section VI, and finally conclusions are given in Section VII.

II. THEORY OF CONFORMANCE

In automatic generation of optimal and efficient test cases for a given protocol specification, traditional methods for test generation are based on a specification S, which is in the form of a strongly connected, minimal and deterministic I/OFSM, or in the form of an LTS [17].

In this section, to define the conformity of an LTS implementation I to a given LTS specification S, we present the definition of trace equivalence and propose that this definition is not a precise one. For a more precise definition for the conformity of the correct implementation I, we propose an observational equivalence. Also in this section, with the assumptions that the specification S and implementation I are strongly connected, deterministic and minimal I/OFSM, and the number of states in the implementation I/OFSM is the same as that of the specification, then we show that the observable equivalence is the same as the definition of trace equivalence.

1. LTS

A. Definition of LTS

The formalism of a labeled transition system is used for modeling the behaviors of processes, systems and components. Labeled transition systems serve as a semantic model for a number of protocol specification languages, e.g. Calculus of Communication System (CCS), Communicating Sequential Processes (CSP), Asynchronous Communicating Processes (ACP) and LOTOS [18]-[20].

Definition 1: A labeled transition system is a 4-tuple $<S, L, T, s_o>$ with

- S is a (countable) non-empty set of states;
- L is a (countable) set of observable actions;
- $T \subseteq S \times (L \cup \{\tau\}) \times S$ is the transition relation;
- $s_o \in S$ is the initial state.

The labels in L represent the observable interactions of a system; the special label $\tau \notin L$ represents an unobservable internal action, and label ε represents null action.

Table 1 introduces some notation for labeled transition systems.

Table 3. The essential conventions for LTS.

Symbol	Semantic
$B-\mu 1,\ldots\mu n\rightarrow C$	$\exists B_i(1\leq i\leq n), B=B_0-\mu_1\rightarrow B_1-\mu_2\rightarrow\ldots-\mu_n\rightarrow B_n=C$
$B=\varepsilon\Rightarrow C$	$B\equiv C$ or $\exists_n\geq 1\ B-\tau^n\rightarrow C$
$B=\mu\Rightarrow C$	$\exists B_1,B_2, B=\varepsilon\Rightarrow B_1-\mu\Rightarrow B_2=\varepsilon\Rightarrow C$
$B=\mu_1,\ldots\mu_n\Rightarrow C$	$\exists B_i(1\leq i\leq n), B=B_0=\mu_1\Rightarrow B_1=\mu_2\Rightarrow\ldots=\mu_n\Rightarrow B_n=C$
$B=\mu_1,\ldots\mu_n\Rightarrow$	$\exists C\ \ B=\mu_1,\ldots\mu_n\Rightarrow C$
$B\neq\mu_1,\ldots\mu_n\Rightarrow$	$-\exists C\ \ B=\mu_1,\ldots\mu_n\Rightarrow C$
out(B)	$\{\mu\in L\mid B=\mu\Rightarrow\}$
Tr(B)	$\{\sigma\in L^*\mid B=\sigma\Rightarrow)\}$

In general, several equivalence relations were studied for the comparison of two LTSs. In this paper, we present two principal equivalence relations: observable equivalence [19] and trace equivalence [20].

B. Observable Equivalence

An observable equivalence is the matter of application of weak bisimulation in comparing two LTSs, $LTS_1=<P, L, T_1, p_o>$ and $LTS_2=<Q, L, T_2, q_o>$, where $P\cap Q=\phi$.

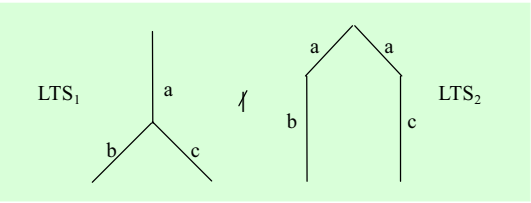
Definition 2: A binary relation $S\subset P\times Q$ between states is defined as a weak bisimulation if $(p, q)\in S$ implies that $\forall\sigma\in(L+\tau)^*$ and $\forall\sigma'\in L^*$ obtained by suppressing all the occurrence of τ in σ,

1) if $p\text{-}\sigma\rightarrow p'$ then there exists q' such as $q\text{-}\sigma'\rightarrow q'$ and $(p', q')\in S$;
2) if $q\text{-}\sigma\rightarrow q'$ then there exists p' such as $p\text{-}\sigma'\rightarrow p'$ and $(p', q')\in S$.

Definition 3: Two LTS systems P and Q are observationally equivalent if and only if there exists a bisimulation S such that every state of P is equivalent to some state of Q and every state of Q is equivalent to some state of P. That is, formally :

1) For all $p\in P$, there exists $q\in Q$, such as $(p, q)\in S$, and for all $q\in Q$, there exists $p\in Q$, such as $(p, q)\in S$.
2) An observable equivalence relation is noted as $P\approx Q$, where $\approx=\cup\{S|S$ is a weak bisimulation$\}$.

The above definition is a weak bisimulation, and the following example is said to be not observationally equivalent.



C. Trace Equivalence

This equivalence relation is not a bisimulation but a language equivalence defined in automata theory [21]. Here, we consider LTS as an I/OFSM model representing a

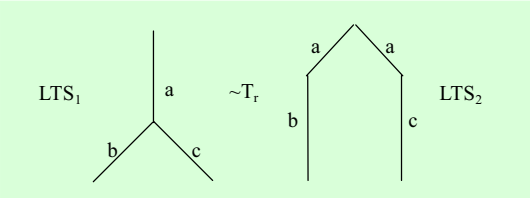
protocol specification, which has no final states.

Definition 4: Two LTSs, $LTS_1 = <P, L, T_1, p_o>$ and $LTS_2 = <Q, L, T_2, q_o>$, where $P \cap Q = \phi$, are said to be trace equivalent if the following condition is satisfied

$$LTS_1 \sim T_r\ LTS_2 \quad \text{iff } T_r(p_o) = T_r(q_o)$$

where T_r is defined in Table 1.

The following example is said to be trace equivalent.



2. I/OFSM

In general, an I/OFSM is usually used for modeling the control part of communication protocols.

Difinition 5: An I/OFSM M is a 6-tuple $<S, so, I, O, \delta, t_r>$ where

- S is a set of finite states,
- $s_o \in S$ is the initial state,
- $I = \{i_1, \ldots, i_n\}$ is a set of input symbols,
- $O = \{o_1, \ldots, o_n\}$ is a set of output symbols,
- δ is an output function $S \times I \rightarrow O$,
- t_r is a transition function, $t_r \subseteq \{s - i/o \rightarrow s' \mid s, s' \in S \wedge i \in I \wedge o \in O\}$

Definition 6: An I/OFSM M is said to be *deterministic* if, for a given input at a given state, only one output is generated and only one next state is defined.

If any given two states, are in the relation of trace equivalence, they are equivalent states.

Definition 7: An I/OFSM M is said to be minimal if it contains no equivalent states.

The machine M is minimal when the number of states of a machine M is equal to or less than the number of states of any machine M′ which is equivalent to M. The machine M has no redundant states.

3. Conformance of LTS Model

Conformance is the correctness of implementation with respect to the given specification. For the discussion of conformance of the LTS model, we consider a natural English language specification of a vending machine.

A. Vending Machine Specification

Let the natural language specification of the vending machine V be given by the following conformance requirements:

r_1: an implementation of V must be able to accept a coin;

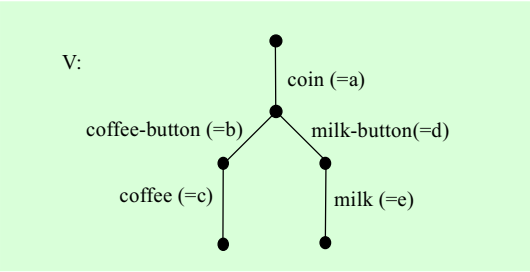
r_2: after an implementation of V has accepted a coin, there is a choice between pushing the coffee-button and the milk-button;

r_3: after the coffee-button has been pushed, the machine shall produce coffee;

r_4: after the milk-button has been pushed, the machine shall produce milk;

$T_r(V)$ =<coin>, <coin, coffee-button>, <coin, coffee-button, coffee>, <coin, milk-button>, <coin, milk-button, milk>

Figure 2 gives an example of an action tree representing the labeled transition system of the vending machine specified as follows:



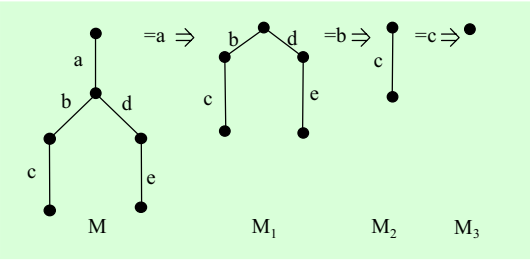


Fig. 2. Action tree transformation.

We suppose the passive action that the environment has the initiative, then specification V is transformed as follows:

B. Definition of Correct Implementation

Figure 3 represents some potential candidates for the correct implementation of V.

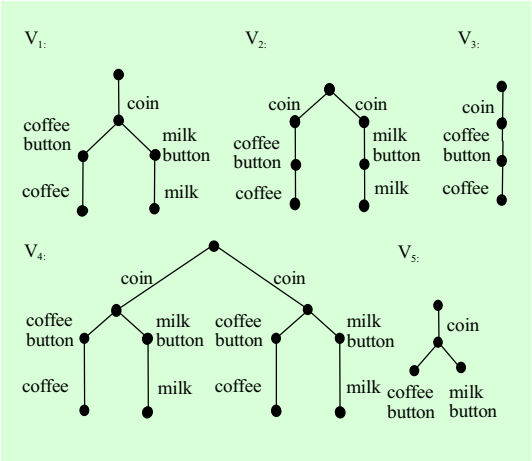


Fig. 3. Implementations for specification V.

Definition 8: I is a correct implementation of specification S when the implementation I shows all sequences of actions possible in the specification, i.e.

$$I \text{ imp } S = \text{def } T_r(I) \supseteq T_r(S).$$

In the case of Fig. 3, the correct implementations are V_1, V_2 and V_4. If we want to exclude the implementation V_2 and V_4 from the correct implementations, the next definition can be tried.

Definition 9: I is a correct implementation of S when implementation I does not show any sequence of actions impossible in the specification S, i.e.

$$I \underline{\text{imp}} \ S = \text{def } T_r(I) \subseteq T_r(S).$$

In the case of Fig. 3, the correct implementations are V_1, V_2, V_3, V_4 and V_5. And also, in order to get the advantages of Definitions 8 and 9, we can try the next definition.

Definition 10: I is a correct implementation of S when implementation I shows all sequences of actions possible in the specification S, and I does not show any sequence of actions impossible in S, i.e.

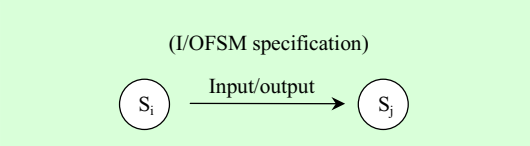
$$\mathrm{I} \underline{\mathrm{conf}}\ \mathrm{S} = \mathrm{def}\ \mathrm{T_r(I)} = \mathrm{T_r(S)}.$$

From this definition, in Figure 3 the user cannot select one of two possible choices after V_2 and V_4 accepts the coin. Nevertheless, in the cases of V_2 and V_4, Definitions 8, 9 and 10 have the disadvantage of including V_2 and V_4 as a correct implementation. So we need a more precise conformance definition which does not include V_2 and V_4 as a correct implementation. In the next subsection, we give some definitions and theorems to solve this problem.

4. Conformance of I/OFSM

In this subsection, for the discussion of more precise conformance concept, we concentrate on the specification S described in I/OFSM that is equivalent to the corresponding FDT specification.

Definition 11: Given an I/OFSM model specifying a protocol description, the definition of conformance testing consists of three steps.



1) The I/OFSM implementation is set to state s_i;
2) *input* is applied, and it is verified that *output* of the implementation I/OFSM is the same as the expected output of the specification I/OFSM.
3) The new state of the I/OFSM implementation is checked to verify that it is really s_j.

In order to generate automatically the optimal and efficient test cases for the given I/OFSM protocol specification, general methods for test generation are based on the specification S, which is in the form of strongly connected, minimal and deterministic I/OFSM. If these conditions are satisfied, the definition of conformity between specification I/OFSM and implementation I/OFSM is dedicated to the trace equivalence or observable equivalence.

The determinism shown in Definition 6 can be defined by another one as follows:

Definition 12: In the transitions set t_r of a deterministic I/OFSM, there is only one transition for any state with the same input symbol, i.e., t_r satisfies the condition:

$$\forall \mathrm{s},\ \mathrm{s_1},\ \mathrm{s_2} \in \mathrm{S},\ \mathrm{i} \in \mathrm{I},\ \mathrm{o_1},\ \mathrm{o_2} \in \mathrm{O}:$$

$$(\mathrm{s\text{-}i/o_1} \rightarrow \mathrm{s_1} \wedge \mathrm{s\text{-}i/o_2} \rightarrow \mathrm{s_2}) \rightarrow \mathrm{o_1} = \mathrm{o_2} \wedge \mathrm{s_1} = \mathrm{s_2}.$$

Definition 13: Let M be an I/OFSM as in Definition 5, then M is *n-observable* iff there is a sequence of labels $x \in (I/O)^n$ such that

$$\forall \mathrm{s},\ \mathrm{s_1}, \mathrm{s_2} \in \mathrm{S},\ (\mathrm{s\text{-}x} \rightarrow \mathrm{s_1} \wedge \mathrm{s\text{-}x} \rightarrow \mathrm{s_2}) \rightarrow \mathrm{s_1} = \mathrm{s_2}$$

Definition 14: Let M be an I/OFSM as in Definition 5. Let $v \in I^*$ and $s, s' \in S$. Then:

1) $\sigma(s, v) = \{(s', v/o) | v/o \in T_r(s)$ and $s\text{-}v/o \rightarrow s'\}$
2) $s(\mathbf{v}) = \{\mathbf{v}/o |$ there exists $s' \in S$ such that $(s', \mathbf{v}/o) \in \sigma(s, v)\}$
3) $M(\mathbf{v}) = s_0(\mathbf{v})$

A member of $M(\mathbf{v})$ is a sequence of labels and is called a trace of M for the input sequence $\mathbf{v}$.

Theorem 1: Let S and I be two I/OFSMs, S describes a protocol specification and I represents an implementation of S. Then S and I are *observationally equivalent* iff

1) S and I are *n-observable*
2) $\forall \mathbf{v} \in I^* : S(\mathbf{v}) = I(\mathbf{v})$

Proof.
1) Suppose two I/OFSMs S and I are not-n-observable. Then, by Definition 12, there exists a non-deterministic state in each I/OFSM. In conformance testing, we cannot solve the controllability of implementation I/OFSM to the traces of S, considered as a set of external behaviors. And also by Definition 3, these two I/OFSMs are not observationally equivalent.
2) $\mathbf{v}$ of $S(\mathbf{v})$ is the set of traces which starts from the initial state in the specification I/OFSM (which is minimal and deterministic). Therefore, by Definitions 13 and 14, the implementation I/OFSM has to accept this set of traces. □

Theorem 2: The observational equivalence defined in Theorem 1, is a trace equivalence relation between two I/OFSMs for S and I.

Proof. Two deterministic I/OFSMs for S and I are in fact n-observable. Therefore a trace equivalence relation is an observational equivalence relation. □

Now, we apply again Theorem 1 to the vending machine example. The specification I/OFSM of the vending machine V is as follows:

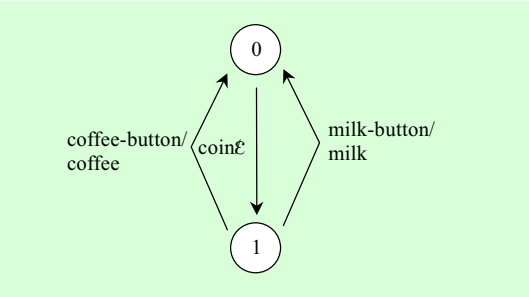


Fig. 4. Specification I/O FSM for V.

A faulty implementation I/OFSM for V (which was accepted as a correct implementation by Definitions 8, 9 and 10) is represented in Fig. 5.

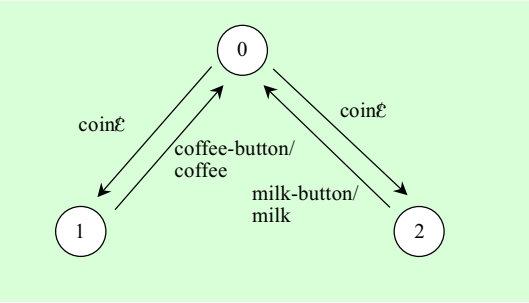


Fig. 5. A faulty implementation I/OFSM for V.

By applying Theorem 1, Fig. 5 is not n-observable equivalent to the specification I/OFSM shown in Fig. 4. Therefore it is

obviously distinguished that Fig. 5 is not a correct implementation. In conclusion, Theorem 1 can be used as a conformation relation between specification S and implementation I.

III. TESTING FRAMEWORK

The problem of conformance testing with the purpose of detecting faults in a black-box implementation is in general unsolvable unless it is dealt with in a restricted framework. We propose here a framework within which our single approach for test generation is presented.

1. Assumptions for Testing

Given an I/OFSM representation (as an intermediate form of SDL or LOTOS specification) of a specification, denoted henceforth as $I/OFSM_S$, and an implementation of this $I/OFSM_S$ denoted henceforth as $I/OFSM_I$, we are asked to determine whether $I/OFSM_I$ conforms to $I/OFSM_S$ by testing $I/OFSM_I$ as a black-box.

Solving this problem implies that we should

1) formally define the conformance relation between $I/OFSM_I$ and $I/OFSM_S$;
2) generate from $I/OFSM_S$ a sequence of test TS_I of inputs and its expected sequence of test TS_O of outputs;
3) apply TS_I to the input PCO port of $I/OFSM_I$;
4) observe a sequence TS_a of actual outputs at the output PCO port of $I/OFSM_I$;
5) compare TS_a with TS_O to determine the conformance of $I/OFSM_I$ to $I/OFSM_S$.

3) to 5) are mainly related to the actual execution of the test sequence; the analysis of test results is beyond the scope of this paper. To solve 1) and 2), we need to make certain assumptions.

Let $I/OFSM_S = < S, s_o, I, O, \delta, t_r >$ and
$I/OFSM_I = < S', s_o', I', O', \delta', t_r' >$.

Definition 15: A UIO sequence is an input/output sequence of identification which is unique for each state for the $I/OFSM_S$.

As defined in Section II, we assume throughout the rest of this section that

- $I/OFSM_S$ is strongly-connected and each of its states has at least one UIO;
- $I/OFSM_S$ is deterministic, and minimal;
- $n'=n$, that is, the number of states (n') in $I/OFSM_I$ is equal to that (n) of $I/OFSM_S$; and
- $I \subseteq I'$, that is, all the input symbols of $I/OFSM_S$ should be included in the input set of $I/OFSM_I$.

We note here in particular that we do not assume reliable reset to be available in $I/OFSM_I$ and that $I/OFSM_S$ need not be completely specified. In fact, the reference I/OFSM is said to be completely defined if, from each state of it, there exists a transition for every input symbol.

2. Conformance Relation

We use the conformance relation, defined as conf in Section II, between $I/OFSM_I$ and $I/OFSM_S$ as follows :

Definition 16: The equivalence of states in respect to a set of input sequences is defined as follows:

Let s_i' be a state of the implementation $I/OFSM_I$ and s_i a state of the specification $I/OFSM_S$. $V \subseteq T_r(s_i')$ is a set of input sequences. Then

$$s_i' \cong_V s_i \quad \text{if} \quad \delta(s_i,\ p) = \delta(s_i',\ p),\ \text{for}\ \forall p \in V.$$

Definition 17: The conformance relation can be defined as follows:

$$I/OFSM_I \ \underline{conf}\ I/OFSM_S \quad \text{iff} \quad s_o' \cong T_r\ \ s_o$$

where s_o and s_o' are the initial states of $I/OFSM_S$ and $I/OFSM_I$ respectively. We note that the conformance relation, conf, is in fact the trace extension relation; that is, an implementation $I/OFSM_I$ conforms to a specification $I/OFSM_S$ if and only if $I/OFSM_I$, when starting from its initial state s_o', can exhibit all the input/output traces specified for the initial state s_o in $I/OFSM_S$. This relation is the same as that of Theorems 1 and 2.

Definition 18: A test case is a sequence of inputs which should be of finite length and in $T_r(s_o)$

Definition 19: A test suite is a set of test cases.

Definition 20: Let TS be a test suite. We say that a given implementation $I/OFSM_I$ passes the test suite denoted by

$$I/OFSM_I \text{ pass TS iff } \delta'(s_0', p) = \delta(s_0, p) \text{ for } \forall p \in TS.$$

Now we can define the concept of a complete test suite.

Definition 21: Given a test suite TS and a specification $I/OFSM_S$, we say that TS is a complete test suite for $I/OFSM_S$ with respect to conf relation if, for any implementation $I/OFSM_I$, the following points hold:

$$I/OFSM_I \ \underline{conf}\ I/OFSM_S \quad \text{iff} \quad I/OFSM_I \text{ pass TS.}$$

IV. AUTOMATED TEST GENERATION METHOD

Several methods use the concept of UIO sequences, in order to generate one or several test cases, when the protocol to be tested is given in the form of an I/OFSM.

In fact, UIO sequences are used to test the resulting state after any transition of the I/OFSM. Hence, for each transition, test subsequence can be formally described as T_{ij}@UIO(s_i), where T_{ij} is the transition to be tested, that takes the specification $I/OFSM_S$ 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 whether the state reached by the transition to be tested is in s_j. The UIO method [10] and the UIOv method [22] assume the reliable reset to be available in

an IUT and therefore these two methods belong to the class of multiple testing approaches. The difference between these two methods is that the UIOv method can guarantee more fault coverage while the UIO method cannot.

Our single testing approach, for generating a complete test suite TS defined above, is to

1) construct a test subsequence for each transition specified in the specification I/OFSMs
2) find a single optimal test case which traverses each of the test subsequences at least once, and if possible at most once by using the rural Chinese postman tour problem [15], [23].

In this approach, we do not use reliable reset assumption. The reliable reset assumption greatly simplifies the test generation problem, and relatively efficient test suites can be obtained. However, as already pointed out by some researchers, the reliable reset may in some cases be difficult to realize, and, therefore, the multiple testing approaches cannot be applied [24].

The procedure of test sequence generation from FDT specifications (SDL or LOTOS) can be divided into six steps as shown in Fig. 6.

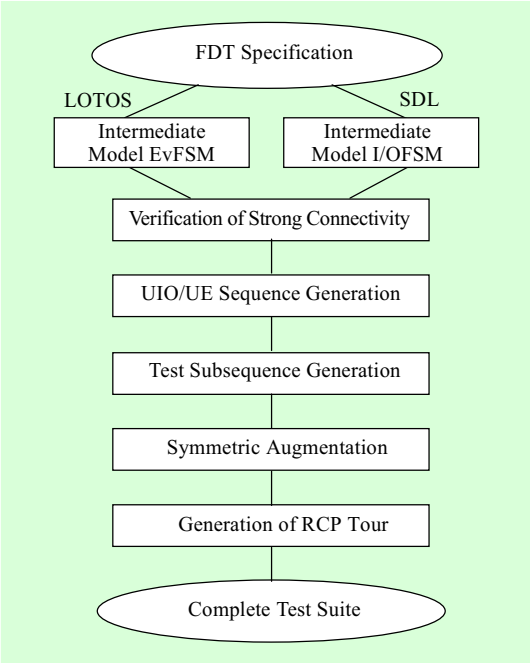


Fig. 6. Procedure of test sequence generation.

1. Intermediate Model (I/OFSM, EvFSM)

The reference I/OFSM from an SDL specifications is obtained by using the GEODE tool. The GEODE tool is used as a front end processor (FEP) in describing an SDL specification and generating the state transition graph from the SDL specification with the interactive graphic environment. GEODE is a software tool from VERILOG, which supports the ITU-T Z.100 and ITU-T Z.120 recommendations, with editing, simulation, code generation and debugging functions for the SDL and MSC descriptions [8].

We also obtain the state-transition graph of an Event Finite State Machine (EvFSM) from a LOTOS specification by means of the 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 Petrinet, from which a state-transition graph is generated by using reachability analysis. We use a kind of FSM that can be derived from LOTOS specifications. In these FSMs, transitions are labeled only by the LOTOS rendezvous corresponding to that transition. This is a classical Moore machine as studied in automata theory, that we call an Event Finite State Machine [7].

Definition 22: Event Finite State Machine
An Event Finite State Machine is a 3-tuple $\sum = (S, E, t_r)$, where

- $S = \{s_0, \ldots, s_n\}$ is a finite, non-empty set of states, with a distinguished element s_0 representing the initial state,
- $E = \{e_1, \ldots, e_m\}$ is a finite set of events, and
- $t_r = S \times E \rightarrow S$ is the state transition function.

2. Verification of Strong Connectivity

The obtained reference I/OFSM (or EvFSM) can be represented by a directed graph G=(V, E), where the set V= $\{v_1, \ldots, v_n\}$ of vertices represents the set of specified states S. A directed edge represents a transition with input and output interaction from one state to another in the FSM. Each directed edge is labeled by the input and output events (or only event) that cause the transition in the FSM.

A directed graph G is said to be strongly connected if, for every pair of vertices v_j and v_k, there exists a path from v_j to v_k. If the obtained FSM is not strongly connected, the specification is not well defined, having deadlocks or livelocks.

In general, the strong connectivity of the reference graph could be verified using the classic depth-first-search algorithm in O (MAX (n, e)) time on an n-vertices, e-edges directed graph. This algorithm can be simply implemented using a graph rewriting system in O (2(N−1)) time. This is interesting since the number of edges is, in general, much greater than the number of vertices in graphs representing protocol specifications [25].

Definition 23: Let G (V, E) and G (V′, E′) be two graphs, G′ is a subgraph of G if V′⊆V and E′⊆E. A neighbor of the subgraph G′ in G is a neighbor of a vertex of V′ in G. Two graphs G_1 (V_1, E_1) and G_2 (V_2, E_2) are isomorphic if there is a pair $\phi = (\phi_V, \phi_E)$ of one-to-one correspondence between V_1 and V_2, and between E_1 and E_2 such that :

$$\forall\{v,v'\} \in E_1, \quad \phi E(\{v,v'\}) = \{\phi V(v), \phi V(v')\}.$$

Let $C^{(V)}$ and $C^{(E)}$ be two finite sets. A labeled graph over ($C^{(V)}$, $C^{(E)}$), denoted by G(V, E, λ), is a graph G(V, E) equipped with a labeling function $\lambda = (\lambda^{(V)}, \lambda^{(E)})$ where $\lambda^{(V)}$ (resp. $\lambda^{(E)}$) is a mapping from V (resp. from E) into $C^{(V)}$ (resp. into $C^{(E)}$).

The graph G(V, E) is called the underlying graph, and the mapping λ is a labeling of it. A c-labeled vertex is a vertex v such that $\lambda^{(V)}(v) = c$.

Let $c \in C^{(V)}$ (resp. $C^{(E)}$), $|G|_c$ is the number of c-labeled vertices (resp. of c-labeled edges) in $G(V, E, \lambda)$. Let $G(V, E, \lambda)$ be a labeled graph. If $G'(V', E')$ is a subgraph of $G(V, E)$ and the restriction of λ to G' is denoted by $\lambda_{|G'}$, we have $\lambda_{|G'} = (\lambda^{(V)}_{|V'}, \lambda^{(E)}_{|E'})$.

Let $G(V, E, \lambda)$ be a labeled graph, and c an element of $C^{(V)}$, then a c-connected component is a connected maximal subgraph of G such that any vertex is labeled c.

Definition 24: Two labeled graphs $G_1(V_1, E_1, \lambda_1)$ and $G_2(V_2, E_2, \lambda_2)$ are isomorphic if there is an isomorphism $\phi=(\phi V, \phi E)$ between $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ such that

$$\lambda_2^{(V)} \circ \phi V = \lambda_1^{(E)}, \quad \lambda_2^{(E)} \circ \phi E = \lambda_1^{(E)}.$$

We will write

$$G_1(V_1,\ E_1,\ \lambda_1) \cong G_2(V_2,\ E_2,\ \lambda_2).$$

A graph rewriting rule R is a couple $R=(G_R(V_R, E_R, \lambda_R), G'_R(V_R, E_R, \lambda'_R))$ of labeled graphs with the same underlying connected graph. Indeed, it is defined by two labelings of the same graph. We define the rewriting relation $\underset{R}{\longrightarrow}$ over labeled graphs
by $G(V, E, \lambda) \underset{R}{\longrightarrow} G'(V, E, \lambda')$ if there exists a subgraph $G_S(V_S, E_S)$ of $G(V, E)$ and an isomorphism ϕ from $G_S(V_S, E_S, \lambda_{|G_S})$ into $G_R(V_R, E_R, \lambda_R)$ such that ϕ is an isomorphism from $G_S(V_S, E_S, \lambda'_{|G_S})$ into $G'_R(V_R, E_R, \lambda'_R)$, and for each vertex $v \in V|V_S$ (resp. for each edge $e \in E|E_S$) one has $\lambda^{(V)}(v)=\lambda'^{(V)}(v)$ (resp. $\lambda^{(E)}(e)=\lambda'^{(E)}(e)$). The graph $G_S(V_S, E_S)$ is the support of the rewriting step. There is a support overlapping with another rule R′ if there exists a subgraph $G'_S(V'_S, E'_S)$ with $V_S \cap V'_S \neq \phi$ which is a support for R′.

Definition 25: A graph $G(V, E, \lambda)$ is irreducible if there is no graph $G'(V, E, \lambda')$ such that

$$G(V,\ E,\ \lambda) \underset{R}{\longrightarrow} G'(V,\ E,\ \lambda').$$

We will say that a graph G is irreducible if none of the rewriting rules can be applied. Obviously, from a given graph G, one can reach one and only one irreducible graph $I_{rr}(G)$ by applying rewriting rules to obtain the desired algorithm.

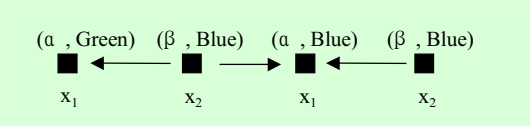
A graph rewriting system is used to verify whether the obtained graph is strongly connected. This kind of graph rewriting system does not modify the underlying graph, but only adds and changes the colors of their components (vertices and edges).

Let x be a vertex. We compute its strongly connected component in the following way : with each vertex v distinct from x, we associate the pair of colors (Green, Green). The initial vertex x is colored by the pair (Red, Blue). Such a graph is said to be a colored graph, whose vertex-color-set is given by $C_V=\{\text{Green, Red}\}\times\{\text{Green, Blue}\}$.

Then we apply the following transformations :

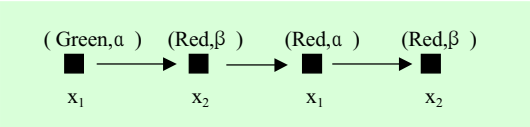
Rule 1 : Let $\alpha, \beta \in \{\text{Green, Blue, Red}\}$; if a vertex x_1 is colored (α, Green) and is linked by an arc (x_2, x_1) to a vertex x_2 colored (β, Blue), then we give x_1 the new color (α, Blue).

This transformation rule can be expressed by the following graph rewriting rule :



Rule 2 : Similarly, if a vertex x_1, colored {Green, α} is linked by an arc (x_1, x_2) to a vertex x_2 colored (Red, β), then we give x_1 the new color (Red, α).

Hence, we obtain the second graph rewriting rule :



Property 1 : We will say that a colored graph G is irreducible if none of the rewriting rules can be applied. Obviously, from a given colored graph G, one can reach one and only one irreducible graph $I_{rr}(G)$ by applying the above rewriting rules.

Proof. Each time we use one of the rewriting rules, one occurrence of the green color disappears. Hence, the number of times we can apply a rewriting rule to the graph G is bounded by the quantity $2(n-1)$ where n stands for the cardinality of set V. □

Property 2 : The strongly connected component of vertex x is the set of vertices which have color (Red, Blue) in the graph $I_{rr}(G)$.

Proof. For any vertex y of $I_{rr}(G)$ which has color (α, Blue) (resp. (Red, β)), there exists a directed path from x to y (resp. from y to x). Hence a vertex y of $I_{rr}(G)$ has color (Red, Blue) if and only if there exists a directed path from x to y and from y to x, which means that y belongs to the strongly connected component of x. □

3. UIO/UE Sequence Generation

The UIO sequence is an input and output sequence of identification which is unique for each state for the I/OFSM generated from SDL specifications. We have adapted a procedure given in [10] for UIO sequence generation. This technique is also utilized to obtain Unique Event (UE) sequences for the EvFSM generated from LOTOS specifications.

Definition 26: Let $\Sigma=(S, E, t_r)$ an EvFSM, and s_i a given state of Σ. A sequence $e_{k1} \cdot e_{k2}, \cdots, e_{kn}$ of events is a unique event sequence for state s_i iff it is an accepted event sequence for state s_i, and for no other state of the EvFSM.

4. Test Subsequence Generation

We calculated TSSs using the formula T_{ij}@UE(s_j) (resp. T_{ij}@UIO(s_j)) for each specified edge from vertex v_i to vertex v_j labeled with e_l, where e_l is a suitable event

label (resp. i/o label) associated with the transition in the reference graph G= (V, E).

The UE (resp. UIO) sequence for state v_i is indicated by UE(v_i) (resp. by UIO(v_i)), and the last vertex of this UE (resp. UIO) sequence is denoted by TAIL(UE(v_i)) (resp. TAIL(UIO(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(UE(v_j))=$v_k\}$ (resp. $E_C = \{(v_i, v_k; e_l)@$ UIO(v_j): $(v_i, v_j; e_l)\in E$ and TAIL(UIO(v_j))= $v_k\}$).

5. Symmetric Augmentation

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

In G', traversing an edge $((v_i, v_k; e_l)@UE(v_j))\in E_C$ (resp. $((v_i, v_k; e_l)@UIO(v_j))\in E_C$) corresponds to realizing the procedure for testing $(v_i, v_k; e_l)$ in G. Therefore, the mini-mum 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 some 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 [26], in which each augmentation is obtained by Dijkstra's shortest path algorithm.

6. Generation of RCP Tour

From the graph G''(rural symmetric augmentation of G'), an optimal test sequence is a 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 Edmonds algorithm that constructs such a Euler Tour in a linear time complexity [27].

The two test sequence generation methods are compared in Table 2.

V. ESTIMATION OF FAULT COVERAGE

The ability of a test sequence to decide whether an IUT conforms to its specification heavily relies upon the range of faults

Table 4. Comparison of test generation from LOTOS and SDL.

No.	Item	LOTOS	SDL
1	Model	Event FSM	I/O FSM
2	Identification sequence	UE(unique event)	UIO
3	Tool for the state transition graph	CAESAR	GEODE
4	Minimization	hiding	internal event
5	End of identification sequences	no restriction	no null output
6	Consideration of complete specification	accept / reject	the input set, input / null
7	Consideration of nondeterminism	events	inputs

or errors that it can detect. To evaluate the fault coverage of a given test sequence, we must generate the class of EvFSMs (resp. of I/OFSMs) which are not equivalent to the specification EvFSM (resp. I/OFSM) but will accept the test sequence. Ideally, fault coverage should be a ratio of the number of non-equivalence EvFSMs (resp. I/OFSMs) which pass the test to the number of all possible EvFSMs (resp. I/OFSMs) which can be generated from the specification EvFSM (resp. I/OFSM) [16].

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

Class 2 : The tail states of two random transitions in the specification EvFSM are changed to obtain this class of EvFSMs.

Class 3 : A new random transition is added in the specification EvFSM to obtain this class of EvFSMs. The tail state, start state 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 EvFSMs. The tail state, start state and transition label are taken from an

independent pseudo-random sequence to ensure fairness.

We also categorize the fault model of the I/OFSMs defined above into the following classes:

Class 1 : The tail states of two random transitions in the specification I/OFSM are changed to obtain this class of I/OFSMs.

Class 2 : The outputs of two random transitions in the specification I/OFSM are changed to generate this class of I/OFSMs.

Class 3 : The output on random transition and the tail state of another random transition in the specification I/OFSM are changed to obtain this class of I/OFSMs.

Class 4 : The tail states and the outputs of two random transitions in the specification I/OFSM are changed to generate this class of I/OFSMs.

In the above model generation, their new values are taken from an independent pseudo-random sequence to ensure fairness [16].

To determine whether the EvFSMs (resp. I/OFSMs) pass the test sequence (i.e., accept the given test sequence with success) means the test actually conform to the specification EvFSM (resp. I/OFSM) by the algorithm given in [28].

We also define two procedures of estimating fault coverage. The one is to estimate fault coverage without verification of the uniqueness of each UE (resp. UIO) sequence prior to the application of the test sequence, denoted UEop (resp. UIOop) method, which combines UE (resp. UIO) sequences with the concept of the Rural Chinese Postman Tour for obtaining an optimal test sequence.

And the other is with verification of the uniqueness of each UE (resp. UIO) sequence in the IUT denoted by UE$\sim$ (resp. UIO$\sim$) verification method. First of all, we give a procedure of fault coverage estimation without UE (resp. UIO) verification as follows :

1) The specification EvFSM (resp. I/OFSM) is read in.
2) The test sequence is generated by our test sequence generation tool.
3) Random EvFSMs (resp. I/OFSM$_S$) which are marginally different from the specification EvFSM (resp. I/OFSM) are generated as described in the above mentioned 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 (resp. I/OFSMs) that passed the test in step 4 are checked if they actually conform to the specification EvFSM (resp. I/OFSM).

To estimate the fault coverage of the obtained test sequence, 4 classes of random EvFSMs (resp. I/OFSMs) are constructed as described previously. For each class, one

million random EvFSMs (resp. I/OFSMs) are generated by our software tool.

By simulation, the test sequence generated by the UEop (resp. UIOop) method is able to detect one or more faults in transitions (resp. in outputs of transitions) and in the tail state of transitions in this kind of EvFSM (resp. I/OFSM). Moreover, the test sequence generated by the UEop (resp. UIOop) method is not able to detect one or more faults in the tail state of transitions in this EvFSM (resp. I/OFSM) model. This arises from the fact that if a UE (resp. UIO) sequence from the specification EvFSM (resp. I/OFSM) is produced by more than one state in the IUT, the test sequence obtained by the UEop (resp. UIOop) method cannot detect one or more faults in tail state transitions.

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

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

By this procedure, the test sequence obtained by the UEop (resp. UIOop) method can detect perfectly one or more faults in the tail state of transitions by UE∼ (resp. by UIO∼) verification.

Based on the results described above, we can conclude that the fault detection capabilities of test sequences for the UEop (resp. UIOop) method with UE∼ (resp. UIO∼) verification can perfectly detect one or more faults in the tail state of transitions in an IUT. Also, this sequence can be used only for the weak conformance test [29].

VI. TEST GENERATION WITH APPLICATION

In this chapter, as an example of test sequence generation by the proposed method, Q.2931 protocol of B-ISDN with some assumptions is considered.

1. Q.2931

Q.2931 protocol specifies the layer 3 call/connection states, messages, information elements, timers and procedures, used for the control of B-ISDN point-to-point on-demand calls on virtual channels [30].

This recommendation specifies the procedures for establishing, maintaining, and cleaning of network connections at the B-ISDN user-network interface. The procedures specified by this recommendation are applied at the interface between a B-ISDN customer network and a B-ISDN public network.

The system, block, process specifications of Q.2931 protocol (outgoing call connection part) in SDL are in Figs. 7, to 9 respectively.

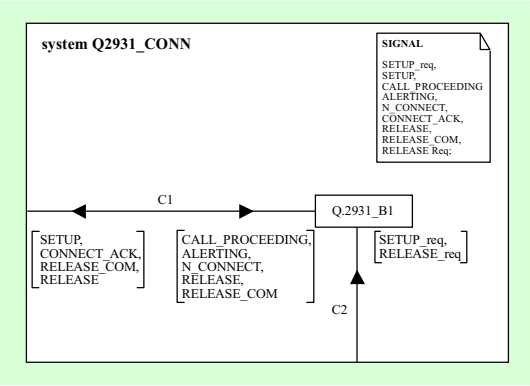


Fig. 7. System specification in SDL.

2. Intermediate Model (I/OFSM)

The reference I/OFSM from an SDL specification is obtained by applying the GEODE tool. The I/O FSM specification

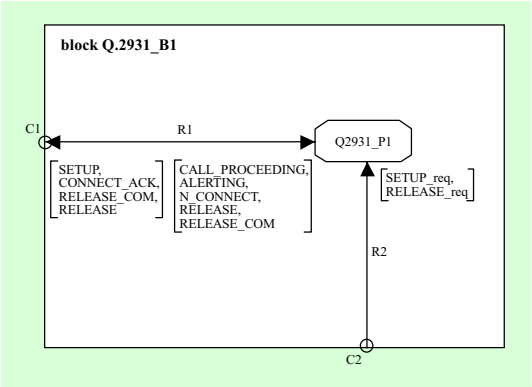


Fig. 8. Block specification in SDL.

(Fig. 10) was obtained from the SDL specification of Q.2931 protocol by the simulation function of the GEODE tool.

3. UIO Sequence Generation

UIO sequences for the I/O FSM shown in Fig. 10 are represented in Table 3.

4. Test Subsequence Generation

Figure 11 is graph G′ obtained by the method described in test subsequence generation of Section IV.

5. Symmetric Augmentation

Figure 12 represents symmetric augmented graph G* for G′, generated by the method described in symmetric augmentation of Section IV.

6. Generation of RCP Tour

We have generated the following test sequence from graph G* shown in Fig. 12 for

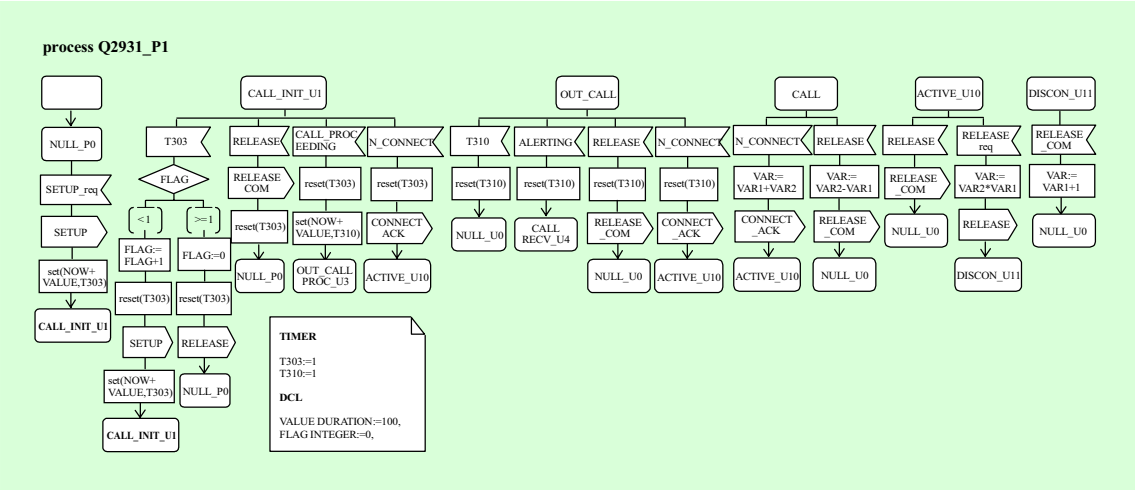


Fig. 9. Process specification in SDL.

Table 5. UIO sequences for the I/O FSM in Fig. 10.

States	UIO Sequences
Null(U0)	SETUP_req/SETUP(A)
Call Initiated(U1)	Call_Proceeding/ε(E)
Outgoing Call Proceeding(U3)	Alerting/ε(G)
Call Delivered(U4)	T301/ε(I)
Active(U10)	Release_req/Release(J)
Release Request(U11)	Release_Com/ε(K)

the given SDL specification by using our tool. The above test cases correspond to a single optimal test sequence which traverse each of the test subsequences at least once. "Preamble" represents the replicated original transition of the reference I/OFSM shown in Fig. 10 to find an RCP tour. "TUT" means each transition under test.

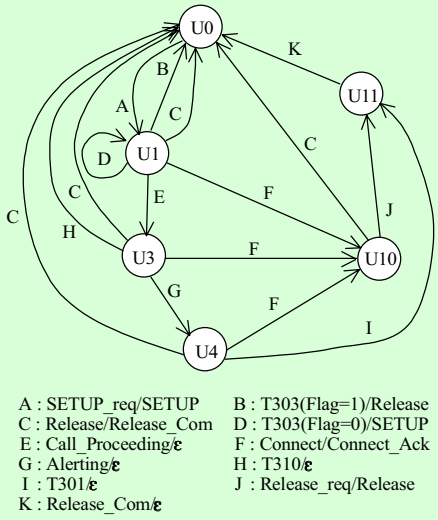


Fig. 10. The reference I/O FSM from the SDL specification of Q.2931.

For estimating the fault coverage of this sequence, 4 classes of random I/OFSMs are constructed by the method described in Section V. For each class, one million ran-

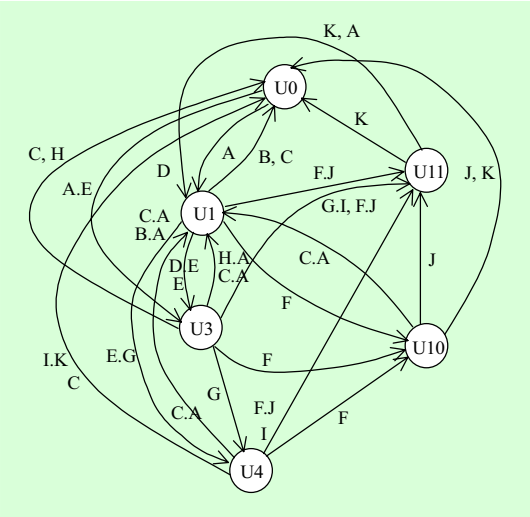


Fig. 11. Graph G′ including test subsequence.

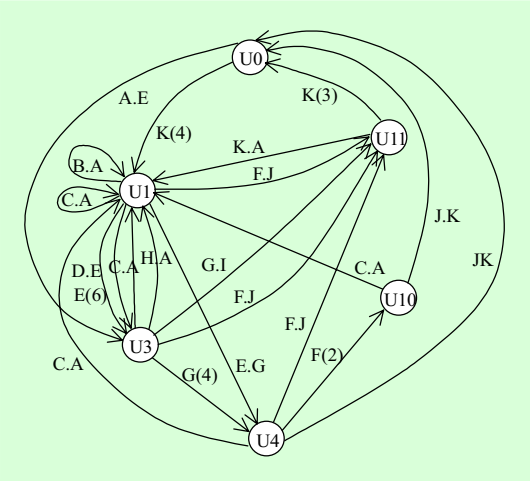


Fig. 12. Symmetric augmented graph G*.

dom I/OFSMs are generated by our software tool. By the results of simulation, we have found that this sequence can detect perfectly one or more faults in transitions and in the tail state of transitions by UIO∼ verification. This sequence can be used for the weak conformance test.

SETUP req/SETUP Preamble(U0→U1)	·	T303(Flag=1)/Release· TUT(U1→U0)
SETUP req/SETUP UIO(U0)	·	Release/Release Com· TUT(U1→U0)
SETUP req/SETUP UIO(U0)	·	Call Proceeding/ε· Preamble(U1→U3)
Release/Release Com TUT(U3→U0)	·	SETUP req/SETUP· UIO(U0)
T303(Flag=0)/SETUP TUT(U1→U1)	·	Call Proceeding/ε· UIO(U1)
T310/ε TUT(U3→U0)	·	SETUP req/SETUP· UIO(U0)
Call Proceeding/ε TUT(U1→U3)	·	Alerting/ε· UIO(U3)
Release/Release Com TUT(U4→U0)	·	SETUP req/SETUP· UIO(U0)
Call Proceeding/ε Preamble(U1→U3)	·	Alerting/ε· TUT(U3)
T301/ε UIO(U4)	·	Release Com/ε· TUT(U11→U0)
SETUP req/SETUP UIO(U0)	·	Call Proceeding/ε· Preamble(U1→U3)
Connect/Connect Ack TUT(U3→U10)	·	Release req/Release· UIO(U10)
Release Com/ε Preamble(U11→U0)	·	SETUP req/SETUP· TUT(U0→U1)
Call Proceeding/ε UIO(U1)	·	Alerting/ε· Preamble(U3→U4)
Connect/Connect Ack TUT(U4→U10)	·	Release req/Release· UIO(U10)

Release Com/ε·SETUP req/SETUP·
Preamble(U11→U4)

Call Proceeding/ε·Alerting/ε
·T301/ε·
TUT(U4→U0)

SETUP req/SETUP·
UIO(U0)

Call Proceeding/ε·Alerting/ε·
Preamble(U1→U4)

Connect/Connect Ack · Release req/Release·
TUT(U4) UIO(U10)

Release Com/ε·
Preamble(U11→U0)

SETUP req/SETUP· Call Proceeding/ε·
Preamble(U0→U3)

Connect/Connect Ack · Release/Release Com·
TUT(U3→U10) UIO(U10)

Release Com/ε · SETUP req/SETUP·
Preamble(U11→U0) Preamble(U0→U1)

Connect/Connect Ack·
TUT(U1→U10)

Release req/Release · Release Com/ε
UIO(U10) Preamble(U11→U0)

VII. CONCLUSIONS

In this paper, we have described an automated method for generating an optimal test sequence from LOTOS and SDL specifications by using and applying many existing techniques.

In development of the proposed method, we have defined a theory concerned with the conformance relation of the LTS and I/OFSM. By using this conformance relation, we have also established a framework leading to a formal testing methodology which may be used to assess conformance of an IUT to behavior specified in an I/OFSM which is equivalent to the FDT specification. To verify the strong connectivity of the reference FSM, we have proposed an algorithm which is easily implemented using a graph rewriting system in O(2(N−1)) time. By applying this automated test generation method to two FDTs (LOTOS and SDL) specifications of communication protocols, we have found that this method of automated test sequence generation is more adaptable to specification changes and also enables us to generate more complete and consistent test sequences.

We have also estimated the fault coverage for the test sequence obtained by our software tool, based on certain assumptions. By the results of fault coverage estimation, the test sequence obtained by the method "UEop + UE∼" (resp. "UIOop + UIO∼) perfectly detect one or more faults in the tail state of transitions in an IUT. Its length is shorter than the sequence obtained by the UIOv method because the transition checking part is optimized against the UIOv method.

Many test selection methods have been developed for the specifications of the protocol being tested, which are given in the

form of a FSM. The test sequences derived by each of the above methods will detect any event (or output) error of the IUT. However, transfer errors (i.e., errors in the next state reached by a transition) will not always be found. But the optimal test sequences generated by the proposed method have perfectly detected all of these kinds of faults.

This formal method on conformance testing has been applied to the protocols related to PCS, B-ISDN, and AIN, partially to generate a complete test suite.

In practice, test sequence is manually generated by extracting test cases from natural language specification. These are largely based on the expert's experiences. who are often unable to cover bugs in the implementation.

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 study topics related to the generation of test sequences from FDT are in the following areas:

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

ACKNOWLEDGMENTS

The authors are very grateful to Dr. J. W. Lee of ETRI and Professor A. Cavalli and other colleagues of the National Institute of Telecommunications of France for many helpful suggestions for this paper. This research was supported partly by the Ministry of Information and Communications in Korea.

REFERENCES

[1] R. J. Linn, "Conformance testing for OSI protocols," *Computer Network and ISDN Systems*, vol. 14, pp. 79-98, 1989.

[2] Samuel T. Chanson *et al.*, "On tools supporting the use of formal description techniques in protocol development," *Computer Network and ISDN Systems* vol. 25, pp. 723-739, 1993.

[3] O. Rafiq, "Le test de conformite des protocols," *Reseaux et Informatique Repartie*, vol. 1, no. 1, pp. 101-142 , 1991.

[4] ISO, "Information processing systems – open system interconnection – OSI conformance testing methodology and framework," *IS 9646*, 1995.

[5] Ana R. Cavalli, J. P. Favreau, and M. Phalippou, "Formal methods for conformance testing: results and perspectives," in *Protocol Test Systems VI, North Holland: Elsevier Science Publishers B. V.*, pp. 3-19, Sep. 1992.

[6] Do Y. Lee and Jai Y. Lee, "A well defined Estelle specification for the automatic test generation," *IEEE Trans. on Computer*, vol. 40, no. 4, pp. 526-542, 1991.

[7] A. Cavalli, S. Kim, and P. Maigron, "Improving conformance testing for LOTOS," *Proc. FORTE'93*, Boston, USA, pp. 371-386, Oct. 1993.

[8] A. Cavalli, Byoung-Moon Chin, and Kilnam Chon, "Testing methods for SDL systems," *Computer Networks and ISDN Systems*, vol. 28, no. 12, pp. 1669-1683, 1996.

[9] Samuel T. Chanson and Jinjong Zhu, "A unified approach to protocol test sequence generation," *IEEE INFOCOM'93*, pp. 106-114, 1993.

[10] K. K. Sabnani and A. T. Dahbura, "A protocol test generation procedure," *Computer Networks and ISDN Systems*, vol. 15, no. 4, pp. 285-297, 1988.

[11] E. Brinksma, "A theory for the derivation of tests," *Protocol Specification Testing and Verification, volume VIII*. North-Holland: Elsevier Science Publishers B.V., pp. 63-74, 1988.

[12] E. Brinksma, J. Tretmans, and L. Verhaard, "A framework for test selection," *in Proc. of the* 11^{th} *Symposium on Protocol Specification, Testing Specification, Testing and Verification*, Stockholm, pp. 233-248, June 17-20 1991.

[13] J. Tretmans, "Test case derivation from lotos specification," *in Proc.* 2^{nd} *Int, Conf. on Formal Description Techniques, FORTE'89*, Vancouver, Canada, pp. 345-359, 1989.

[14] Petrenko, A., Bochmann, v.G., and Dssouli, R., "Conformance relations and test derivation," *IFIP Transactions Protocol Test Systems VI*, North-Holland, pp. 157-178, 1994.

[15] A. V. Aho *et al.*, "An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours," *IEEE Trans. on Communications*, vol. 39, no. 11, pp. 1604-1615, Nov. 1989.

[16] D. Sidhu and T. Leung, "Fault coverage of a protocol test methods," *in Proc. IEEE INFOCOM'88*, pp. 80-85, 1988.

[17] Young-Han Choe, Sung-Un Kim, and Byoung-Moon Chin, "Communication protocol conformance testing," *ITC-CSCC'96*, Seoul, pp. 131-134, June 1996.

[18] D. Park, "Concurrency and automata on infinite sequence," In Peter Deussen, Ed., *Theoretical Computer Science*, vol. 104, *Lecture Notes in Computer Science,* Berlin: Springer-Verlag, pp. 167-183, 1981.

[19] R. Milner, *Communication and Concurrency*, Prentice Hall International, 1989.

[20] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice Hall International, 1985.

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

[22] W. Y. L. Chan, S. T. Vuong, and M. R. Ito, "An improved protocol test generation procedure based on UIOv," *SIGCOM'89 Symposium: Communication Architecture and Protocols in Computer Comm. Review 19(4)*, pp. 283-294, Sep. 1989.

[23] Y. N. Shen *et al.*, "Protocol conformance testing using multiple UIO sequences," *Proc. of PSTV'89*, pp. 131-143 , 1989.

[24] S. Fujiwara *et al.*, "Test selection based on finite state models," *IEEE Trans.on Software Eng., Se-17*, no. 6, pp. 591-603, June 1991.

[25] J. Rhee, S. Kim and Y. Koo, "Strong connectivety decision method using graphs rewriting system in conformance testing," the *Trans. of the Korea Information Processing Society*, vol. 4. no. 5, pp. 1327∼1336, May 1997.

[26] R. E. Tarjan, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, USA, 1983.

[27] J. Edmons and E. L. Johnson, "Matching, Euler tours and the Chinese postman's tour," *Mathematical Programming*, vol. 5, pp. 88-124, 1973.

[28] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Reading, Mass.: Addison-Wesley, 1974.

[29] Byoung-Moon Chin, A. Cavalli, and T. Macavei, "Test generation from SDL and I/OFSMs," *Proc. of Twelfth Int'l Conf. on Computer Communications*, Seoul, Korea, pp. 447-452, Aug. 1995.

[30] ITU-T, *User-Network Interface (UNI) Layer 3 Specification for Basic Call/Connection Control*, Q.2931, Oct. 1994.

Byoung-Moon Chin was born in Seoul, South Korea in 1953. He received his B.S. degree in electrical engineering in 1976 and M.S. degree in computer engineering in 1983, both from Seoul National University. He received his Ph. D. degree in computer science in 1996 from Korea Advanced Institute of Science and Technology (KAIST). He joined ETRI in 1980, where he is currently working as Director at the Protocol Engineering Center, and a Special Rapporteur of ITU-T SG7 Q.17 on data communication protocol testing. His research interests include protocol testing, test methodology, protocol engineering and computer networks.

Young-Han Choe received his B.S. degree in electronics engineering from Kyungpook National University in 1981 and his M.S. degree in computer science from Chung-Nam National University in 1992. He joined ETRI in 1982. From 1987 to 1989 he was with AT&T Bell Labs in Holmdel, U.S.A for Channel Bridging project. He is currently working at the Protocol Engineering Center in ETRI, and an editor of ITU-T SG7 Q.17 on data communications protocol testing. His research interests include protocol engineering and development of test methodologies on telecommunication protocols.

Sung-Un Kim received his B.S. degree in electronics engineering from Kyungpook National University in 1982. He joined ETRI in 1982 and then Korea Telecom Research Labs (KTRL) in 1985. While working for the KTRL, he received his M.S. and Ph. D. degrees in computer science from the University of Paris 7 in 1990 and 1993, respectively. He is currently a Professor in the department of Telematics Engineering, Bukyung National University, Pusan, Korea. His interests include protocol engineering on telecommunications, protocol testing, computer networks, and distributed systems.

Jae-Il Jung received his B.S. degree in electronic engineering from Hanyang University, Seoul, Korea, in 1981, his M.S. degree in electrical and electronic engineering from Korea Advanced Institute of Science and Technology (KAIST), Seoul, Korea, in 1984, and his Ph.D. degree in computer science and networks from Ecole Nationale Superieure des Telecommunications (ENST), Paris, France, in 1993. After receiving M.S. degree, he was with Korea Telecom Research Labs. from 1984 to 1997. He is currently an Assistant Professor at Hanyang University. During the last period of his Ph.D. at ENST, he was a research engineer at Centre National d'Etudes des Telecommunications (CNET), Lannion, France. His research interests include broadband networks and services, especially in QoS/network performance specification in ATM/B-ISDN. He is a member of IEEE Communication Society.