

# An Optimization Technique for Feature Interaction Test Generation

Caixia Chi

*Lucent Technologies, Bell Labs*

*15/F, Aero Space Great Wall Building*

*No.30, Hai Dian Nan Lu Beijing, 100080, P.R.C.*

**Abstract.** This paper proposes a method to generate test cases for testing feature interactions of a protocol as well as for checking the conformance of a protocol implementation to its specification. An optimization technique is used to determine a minimum-cost tour of the dual transition graph of a finite state machine and test sequences that cover all feasible combinations of features with minimum number of executions can be provided. The technique is applied to Link Management Protocol (LMP) of optical networks to get test sequences for LMP feature interaction testing.

## 1 Introduction

With the convergence of 3G wireless and mobile Internet, more and more feature-rich communication systems are designed and deployed. An example is the popular MSN messenger client which provides voice call, instant messaging and video communication services. A feature-rich communication system is a system that can offer many value-added services to its users, in which different services may interfere with each other, and result in a problem known as feature interaction [1]. For example, Internet telephony end systems can offer basic call functions, as well as some value-added services including automatic call answering, call forwarding, call waiting, call redirection, etc. When a user wants to apply a feature to automatically accept an incoming call in addition to an existing call forwarding feature, the interaction between automatic call answering and call forwarding occurs. Feature interactions also occur in very low level communication protocol systems. Feature interactions between Link Management Protocol(LMP) and Label Distributed Protocol(LDP) have been identified in [2]. Interactions between features of a communication system are usually caused by many reasons such as resource sharing, requirement violation, and can be identified through various ways including protocol verification, simulation, or testing. In this paper, we focus our discussion on how to test feature interactions in a real system to establish the confidence that a system is free of feature interactions.

As a process to establish that an implementation of a service meets its specification and interacts sensibly with other service implementations, testing for feature interactions has been an important part of system integration. After an implementation is available, careful and methodical testing takes place to guarantee that feature interactions do not occur. Godskesen [3] describes formally, how tests have to be applied in order to test for the absence of interference on the implementation level using tests

for all the implementation of the individual features. About how implementations of features can be tested for absence of interaction, the approach advocates to test the combination of the feature implementations with the test for all the implementations of the individual features. However, as the number of services increases the number of potential tests can become unmanageable and uneconomic and so some degree of test selection is necessary. Kelly et al. [4], [5] present an approach using simulation of feature behavior on the network level (for Centrex services) and on the service plane (for IN services), using a SDL-based animation tool. They use the service scenarios identified during service validation which display feature interaction to form test suites for a set of interacting services. Faci [6] suggests to detect feature interactions by exploiting testing theory for services specified in LOTOS by generating test cases automatically from the formal specifications of the services.

Test sequence generation has been studied for many years and lots of work has been done on generating test sequences for finite-state machine (FSM) specifying a communication protocol system [7][8][9][10]. All these work focus on looking for conformance test sequences that meets certain coverage criteria, such as a transition tour [10], or a postman tour [11] and no work has taken feature interaction testing into consideration yet. Conformance test presents a method to test whether there is a discrepancy between the specification and the implementation of an FSM [11]. Typically, the implementation of a system is tested for conformance by applying a sequence of inputs from an external tester, and then verifying that the corresponding sequence of outputs is what is expected [11]. Generating conformance test cases that guarantee certain fault coverage criterion and emphasize on testing feature interactions is a desirable way to design test sequences for a system. This paper describes a technique to generate optimal test sequences for testing feature interactions as well as checking conformance of an implemented system. The mechanism proposed in this paper identifies the feasible combination of the operations from different features, an optimization technique is used to find test sequences that cover all these feasible combination with minimum number of executions.

In section 2, some preliminaries knowledge on graph theory and finite automata theory is introduced. Section 3 describes the algorithm to generate test sequence minimal in time, in section 4 the proposed algorithms are applied to the data link specification of Link Management Protocol [12]. The paper concludes in section 5.

## 2 Preliminaries

### 2.1 Graphs

Let  $G = (V, E)$  be a labelled directed graph with vertex set  $V$ , edge set  $E$ , where  $V = \{v_1, \dots, v_n\}$  and  $m = |E|$ .  $G$  may contain loops and parallel edges, which are distinguished from one another by different labels. An edge  $e$  from vertex  $v_i$  to  $v_j$  is represented by a triple  $(v_i, v_j; L_e)$ , where  $L_e$  is a label such that each edge in  $E$  has a unique representation.

A *walk* in  $G$  is a finite, non-null sequence of consecutive edges:  $W = (v_{i_1}, v_{i_2}; L_1) (v_{i_2}, v_{i_3}; L_2) \dots (v_{i_{r-1}}, v_{i_r}; L_{r-1})$ . Note that in a walk, a particular edge may appear more than once. Vertex  $v_{i_1}$  is called the *origin* of  $W$ , and  $v_{i_r}$  the *tail* of  $W$ . A *tour* is a walk that starts and ends at the same vertex [13]. An *Eulerian tour* of  $G$  is a tour which

contains every edge of  $E$  exactly once.

Graph  $G$  is *strongly connected* if for any pair of distinct vertices  $v_i$  and  $v_j$ , there exists a walk  $W$  in  $G$  with origin  $v_i$  and tail  $v_j$ . [15]

The in-degree and out-degree of a vertex  $v_i$  in  $G$  are denoted by  $d_G^-(v_i)$  and  $d_G^+(v_i)$ , respectively. The index  $G$  is omitted if  $G$  is obvious in the context. A directed graph is *balanced* if for every vertex  $v_i$ ,  $d^+(v_i) = d^-(v_i)$ . For each  $v_i \in V$ , set  $\sigma_i = d^-(v_i) - d^+(v_i)$ . Let  $S = \{v_i \in V | \sigma_i > 0\}$ ,  $T = \{v_i \in V | \sigma_i < 0\}$ ,  $\sigma = \sum_{v_i \in S} \sigma_i$ .

A *postman tour* of  $G$  is a tour which contains every edge of  $E$  at least once. The *Chinese postman problem* is to find an optimal (minimum-cost) postman tour of a directed, strongly connected graph  $G$ ; such a tour is called a *Chinese postman tour*.

## 2.2 Finite-State Machines

A given finite-state machine (FSM)  $M$  can be taken as a directed graph  $G = (V, E)$ , where  $V = \{v_1, \dots, v_n\}$  represents the specified states of the FSM and a directed edge represents a transition from one state to another in the FSM [11]. In this paper, it is assumed that  $G$  is strongly connected.

The following symbols are introduced in [11] and we include it here for the convenience of the reader. There is an edge in  $E$  from  $v_i$  to  $v_j$  with label  $a_k/o_l$  if and only if FSM  $M$ , in state  $s_i$  upon receiving input  $a_k$  produces output  $o_l$ , and moves into state  $s_j$ . When there are multiple transitions from state  $s_i$  to  $s_j$ , there are multiple parallel edges from vertex  $v_i$  to  $v_j$  in the corresponding graph  $G$ . Therefore, an edge in  $G$  is fully specified by a triple  $(v_i, v_j; L)$ , where  $L \equiv a_k/o_l$ ,  $L^{(i)} \equiv a_k$ , and  $L^{(o)} \equiv o_l$ . It is assumed here that  $M$  is a deterministic FSM, that is, for a vertex  $v_i \in V$  which has two outgoing edges  $(v_i, v_j; L_1) \in E$ , and  $(v_i, v_k; L_2) \in E$ ,  $L_1^{(i)} \neq L_2^{(i)}$ , although it is permissible that  $L_1^{(o)} = L_2^{(o)}$ . In this case, a walk  $W$  in  $G$  which corresponds to a sequence of state transitions is specified by its origin (the initial state) and a sequence of input operations.

For a state machine that describes the behaviors of a feature-rich communication system, different features will often trigger different transitions of the state machine. For the ease of better presentation, we assign each system feature with a distinguished color, then  $G$  can be transformed into a colored graph, in which the color associated with each edge is the same as the color of the feature that realizes the corresponding transition in  $M$ .

Let  $G = (V, E, C)$  be a colored graph with vertex set  $V$ , edge set  $E$  and color set  $C$ , where  $V = \{v_1, \dots, v_n\}$  and  $m = |E|$ ,  $C = \{c_1, \dots, c_k\}$ . Each edge  $e \in E$  is assigned a color  $c_e \in C$ .

## 3 Problem Statement

Feature interaction testing is an important part of system integration. There have been some work on testing feature interactions [3][4][5][6], but no systematic method has been provided to design test cases for feature interaction testing purpose. Previous effort on test sequence generation have been mostly focusing on the general conformance test problem with a purpose to establish the confidence that a given implementation is in compliance with every function/feature description of a specification. It stresses more

on checking the compliance of individual feature of a system. However, even if an implementation passes the tests for all individual features, it still might fail to perform a function when there are other features running in the system concurrently. In the following, test sequence generation with a stress on feature interaction testing is stated as an optimization problem, and two algorithms are proposed to generate test sequences that cover all feasible combinations of features with minimum number of executions.

Suppose a system has feature set  $\{F_1, \dots, F_k\}$ ,  $G = (V, E, C)$  is the state machine specifying the system. If a transition is resulted from feature  $F_i$ , it is assigned a color  $c_i$ , that is, by such an assignment, transitions from different feature are assigned with different colors.

**Definition 3.1** *Given a state machine  $G = (V, E, C)$ ,  $e_1 = (v_i, v_j)$ ,  $e_2 = (v_p, v_q)$  are transitions from two features  $F_1, F_2$ . If  $e_1$  and  $e_2$  are adjacent in  $G$ , that is  $v_j = v_p$ , then the feature combination  $(F_1, F_2)$  is feasible.*

Note that,  $(F_1, F_2)$  is feasible does not mean  $(F_2, F_1)$  is feasible, for there may be no transitions from  $F_2$  adjacent with transitions from  $F_1$ . Given a state machine  $G$  and feature set  $F = \{F_1, \dots, F_k\}$ , all feasible combinations of features can be get by searching  $G$ .

*Problem 1: Given a colored digraph  $G = (V, E, C)$ , find test sequences that covers all feasible feature combinations with the minimum number of executions.*

To study the complexity of problem 1, we first construct the dual graph of a given state machine  $G = (V, E, C)$ .

**Algorithm 1:** Construct Dual Graph of a Given Digraph.

Input:  $G = (V, E, C)$

Output: Dual Graph  $\hat{G} = (U, A, \hat{C})$  of  $G$ .

**begin**

1.  $U = E$ , that is, each edge in  $G$  corresponds to a node in  $\hat{G}$ .  
Let  $u_i \in U$  corresponds to  $e_i \in E$ .
2. For  $e_i = (v_r, v_s)$ ,  $e_j = (v_p, v_q) \in E$ , if  $v_s = v_p$ , that is,  $e_i$  and  $e_j$  are adjacent via node  $v_s$  in  $G$ , then  $(u_i, u_j) \in A$ .
3. For any  $(u_i, u_j) \in A$ ,  $\hat{c}(u_i, u_j) = (c(e_i), c(e_j))$ . That is, the edge color of  $(u_i, u_j)$  in  $\hat{G}$  includes two elements from the edges in  $G$  that form node  $u_i$  and  $u_j$ .

**end**

For two edges  $a_1$  and  $a_2$  in  $\hat{G}$  with colors  $\hat{c}(a_1) = (i, j)$  and  $\hat{c}(a_2) = (k, q)$ , they are said to have the same edge color if  $i = k, j = q$ .

Fig.1(1), (2) illustrate a graph  $G$  and its dual graph  $\hat{G}$ . The number on each edge is the color assigned to the edge.

By such a transformation, each color  $\hat{c} \in \hat{C}$  corresponds to a feasible feature combination in  $G$ . Problem 1 is changed to find test sequences that covers all different

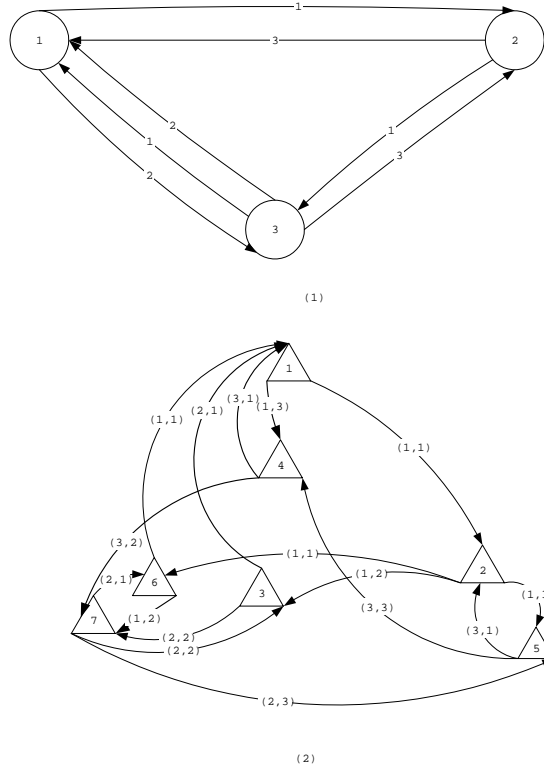


Figure 1: A Prime Graph  $G$  and its Dual Graph  $\hat{G}$

colors in  $\hat{G}$  with the minimum number of edges. In [14], it is proved that finding a test sequences that cover all different colors with minimum cardinality is NP-hard. Two heuristic algorithms are proposed to find approximate solutions to problem 1. One is modified Chinese postman tour algorithm applied to the dual graph of an FSM, the other is a greedy algorithm applied directly to the original FSM such that the complex dual graph need not be constructed.

**Algorithm 2:** A Heuristic Solution to Problem 1

Input:  $G = (V, E, C)$  /\* a strongly connected colored digraph \*/

Output: A shortest Postman Tour with all feasible color combinations included.

**begin**

1. Call Algorithm 1 to get dual graph  $\hat{G} = (U, A, \hat{C})$  of  $G$ .
2. For each  $u_i \in u$ , set  $\sigma_i := d^-(u_i) - d^+(u_i)$ .
3. If  $\sigma_i = 0, i = 1 \dots n$ , then set  $\tilde{G} := \hat{G}$  goto step 8; Otherwise,
4. Let  $S = \{u_i \in U | \sigma_i > 0\}, T = \{u_j \in U | \sigma_j < 0\}$ .  $\forall u_i \in S, \forall u_j \in T$ , find the shortest path from  $u_i$  to  $u_j$  in  $\hat{G}$ ;
5. Construct a complete bi-partite graph  $H = (X, Y, E_H, W)$ , with  $X = \{x_{i,p} | u_i \in S, p = 1, 2, \dots, \sigma_i\}, Y = \{y_{j,q} | u_j \in T, q = 1, 2, \dots, |\sigma_j|\}, E_H = \{x_{i,p}y_{j,q} | x_{i,p} \in X, y_{j,q} \in Y\}$ . Associate each edge  $x_{i,p}y_{j,q}$  in  $H$ ,  $p = 1, 2, \dots, \sigma_i, q = 1, 2, \dots, |\sigma_j|$ , a weight  $w(u_i, u_j) \in W$ , where  $w(u_i, u_j)$  is the length of the shortest path between  $u_i$  and  $u_j$  in  $G$ .

6. Find the perfect match  $M = \{e_1, e_2, \dots, e_k\}$  in  $H$ , such that  $w(M) = \sum_{e \in M} w(e)$  is minimized.
  7. For each edge  $x_{i,p}y_{j,q} \in M$ , suppose the shortest path between  $u_i$  and  $u_j$  in  $\hat{G}$  is  $P_{i,j}$  add every edge in  $P_{i,j}$  to  $\hat{G}$ . Set the newly augmented balanced graph as  $\tilde{G}$ .
  8. To find the *Euler tour*  $T$  in  $\tilde{G}$ .
  9. Return  $T$ .
- end**

The algorithm to find perfect match and Euler Tour in a balanced graph can be found in [16]. Algorithm 2 gives test sequence that not only cover all feasible feature combinations, they also cover all transitions and can be used for conformance testing. Usually an FSM has a large set of transitions such that its dual graph can be very complicated. In the following, a heuristic algorithm is proposed to solve Problem 1 directly without constructing the dual graph of the FSM.

**Algorithm 3:** A Greedy Algorithm to Problem 1

Input:  $G = (V, E, C)$

Output: Test sequences cover all feasible color combinations  $(c_i, c_j)$ , where  $c_i, c_j \in C$ .

**begin**

1. For any edge  $e_i, e_j \in E$  with color  $c(e_i), c(e_j)$ , if  $e_i$  is adjacent with  $e_j$ , then  $(c(e_i), c(e_j)) \in \hat{C}$ .
2.  $i := 1$ .
3. Get an arbitrary vertex  $v_i \in V$ , for all outgoing edges  $(v_i, v_j)$ , select a  $(v_i, v_j)$  such that there exists one outgoing edge  $(v_j, v_k)$ ,  $(c(v_i, v_j), c(v_j, v_k)) \in \hat{C}, P_i = (v_i, v_j) \cup (v_j, v_k)$ ,  $\hat{C} = \hat{C} - (c(v_i, v_j), c(v_j, v_k))$ , if  $\hat{C} = NULL$ , stop and output all  $P_i$ ; otherwise  $v_i = v_j$ , go to step 3.
4. If there is no such  $v_j$ , change  $v_i = v_j, i = i + 1$ , go to step 3.

**end**

Test sequences  $P_i$  generated by algorithm 3 may not start from the initial state  $v_0$  of the input FSM. Suppose  $P_i$  starts from node  $v_i \neq v_0$ , to get test sequences starts from  $v_0$ , the shortest path from node  $v_0$  to  $v_i$  can be prefixed to  $P_i$ .

## 4 Feature Interaction Testing For LMP

Feature interactions in optical network protocols have been classified in [2]. In the following, Link Management Protocol(LMP) is introduced briefly, then algorithms 2, and algorithm 3 are applied to generate test sequences for data link FSM of LMP.

### 4.1 Introduction to LMP

Generalized Multiprotocol Label Switching (GMPLS) is being standardized by Internet Engineering Task Force (IETF) to serve as an integral protocol for the next generation

of data networks. GMPLS provides the necessary bridge between the IP and photonic layers to allow for interoperable and scalable parallel growth in the IP and photonic dimensions [17]. From the functional point of view, a GMPLS control plane can be partitioned into three components: link management, routing, and signaling. As an extension of MPLS, GMPLS needs modify the routing and signaling protocols of MPLS, and add a new protocol to address issues related to link management in optical networks developed with photonic switches (PXC), optical cross-connects (OXC), routers, switches, DWDM systems, and add-drop multiplexors (ADM) [12]. Link Management Protocol (LMP), enhancements to OSPF/IS-IS, and enhancements to RSVP/CR-LDP are being standardized by IETF under the umbrella of GMPLS respectively.

LMP provides the fundamental functions to support GMPLS routing and signaling protocols. For example, LMP manages Traffic Engineering (TE) links composing of a number of data-bearing links between two nodes and the TE link information can be used by routing protocols to generate their Link State Advertisements (LSAs); it also maps TE links and control channels which can be used by signaling protocols to set up a Label Switched Path (LSP). Thus, LMP performs a valuable “glue” function in the control plane [12].

The features of LMP include: control channel management, link property correlation, link connectivity verification, and fault management. Control channel management and link property correlation are the primary feature of LMP and all the others are extended in different implementation environment. The features of LMP is explained in the following:

- Control Channel Management: This allows two nodes in optical network to establish and maintain control channels between adjacent nodes.
- Link Property Correlation: This allows two nodes in optical network to automatically exchange their TE link properties, verify the TE link configuration.
- Link Connectivity Verification: This allows two nodes in optical network to discover their data plane neighbor, exchange their interface ID, and verify their physical connectivity.
- Fault Management: This allows nodes in optical network to suppress downstream alarms, localize faults for protection and restoration.

All these features are specified with Control Channel FSM, Data Link FSM and TE Link FSM in LMP draft [12]. In most cases, Control Channel Management controls the state transition of control channel, Link Property Correlation controls the states of TE link, behaviors of Link Connectivity Verification and Fault Management can change the state transition of data link. On the other hand, these features are not independent, they interact with each other via the operation on the shared state machine. For example, Link Property Correlation can change data link state when it finds the data link property is not correlated in both sides, Control Channel Management can change TE link states when there is no active control channels to control the TE link. On the other hand, as a “glue” protocol in GMPLS architecture, the resource managed by LMP including control channel, TE link and data links can be accessed by signaling protocols such as LDP and routing protocols such as OSPF. For example, LDP can

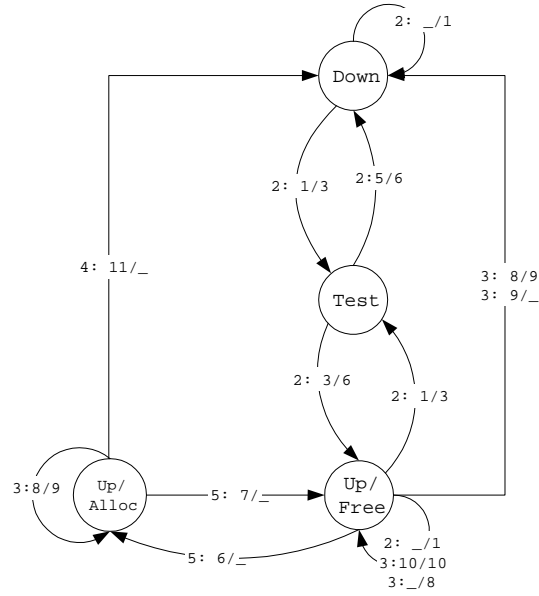
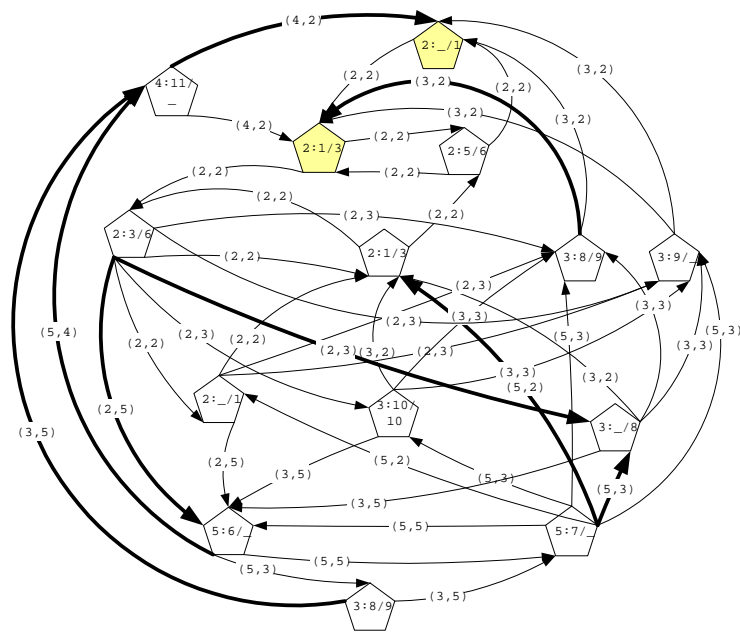


Figure 2: Active LMP Data Link FSM

change the state a data link when it needs to setup/release a label switched path. LMP draft specifies all the possible operations coming from different features of LMP and protocols that can result in the state transition of control channels, TE links and data link. Whether these operations are consistent and whether they can interact correctly or not needs to be tested. In [2], some feature interaction problems in LMP have been identified, but no systematic method is provided to detect these feature interactions. In the following, we use the data link FSM of LMP as an example, apply the algorithm proposed in previous section to generate test sequences to test feature interactions of an LMP implementation.

#### 4.2 Data Link Model of LMP

Fig. 2 shows the active LMP data link FSM and Fig.3 shows the dual of Fig.2. For Fig.2, the label on each transition is  $c : i/o$ , where  $c$  is the color assigned to the transition,  $i$  is the input event for the transition and  $o$  is the output event of the transition. Explanation of the transitions are given in the following table, in which ! represents the event of sending out a message and ? represents the event of receiving a message. For Fig.3, the label on each transition is the color pair  $(c_i, c_j)$ . The feasible feature combinations that must be covered by test sequences are highlighted, which are  $(2, 3)$ ,  $(2, 5)$ ,  $(3, 2)$ ,  $(3, 5)$ ,  $(4, 2)$ ,  $(5, 2)$ ,  $(5, 3)$ ,  $(5, 4)$ .



Color Assignment: 1: CCM; 2: LCV; 3: LPC; 4: FM; 5: LDP.

Figure 3: Dual of Active LMP Data Link FSM

Input:

- 1 : ?msgBeginVerifyAck: Receives BeginVerifyAck message.
- 2 : ?msgBeginVerifyOK: Receives correct BeginVerify message.
- 3: ?msgTstSuccess: Receives TestStatusSuccess message.
- 4 : ?msgTestOK: Receives correct Test message.
- 5 : ?msgTstStatusFailure: Receives TestStatusFailure message.
- 6 : evLnkAlloc: Allocate the data link to an LDP request.
- 7: evLnkDealloc: Deallocate the data link.
- 8: ?msgLinkSumErr: Receives error LinkSummary message.
- 9: ?msgLinkSumNack: Receives LinkSummaryNack message.
- 10: ?msgLinkSumOK: Receives correct LinkSummary message.
- 11: evLocalizeFail: FM localizes a Failure.

Output:

- 1 : !msgBeginVerify: Sends out BeginVerify message.
- 2 : !msgBeginVerifyAck: Sends out BeginVerifyAck message.
- 3 : !msgTest: Sends out Test message.
- 4 : !msgTestSuccess: Sends out TestSuccess message.
- 5 : !msgTestFailure: Sends out TestFailure message.
- 6 : !msgTstStatusAck: Sends out TestStatusAck message.
- 7: !msgBeginVerifyNack: Sends out BeginVerifyNack message.
- 8: !msgLinkSum: Sends out LinkSummary message.
- 9: !msgLinkSumNack: Sends out LinkSummaryNack message.
- 10: !msgLinkSumAck: Sends out LinkSummaryAck message.

### 4.3 Optimal Test Sequence for LMP

We apply algorithm 2 and 3 to get test sequences for LMP active data link FSM. Then we will analyze these test sequences to see how possible interactions may be detected.

The tour generated by algorithm 2 is divided into several sequences when it traverses the initial node 2 : 1/3 in the dual graph.

1. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ evLnkAlloc → evLnkDealloc/\_ → ?msgLinkSumNack/\_ → \_/!msgBeginVerify
2. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ evLnkAlloc → evLnkDealloc/\_ → \_/!msgLinkSum → ?msgLinkSumNack/\_
3. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ evLnkAlloc → evLnkDealloc/\_ → ?msgBeginVerifyAck/!msgTest  
→ ?msgTstStatusFailure/!msgTstStatusAck
4. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ evLnkAlloc → evLnkDealloc/\_ → ?msgLinkSumOK/!msgLinkSumAck  
→ ?msgLinkSumNack/\_
5. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ evLnkAlloc → evLnkDealloc/\_ → \_/!msgBeginVerify  
→ ?msgLinkSumNack/\_
6. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ \_/!msgBeginVerify → ?msgLinkSumErr/!msgLinkSumNack  
→ \_/!msgBeginVerify
7. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ \_/!msgBeginVerify → ?msgBeginVerifyAck/!msgTest  
→ ?msgTstSuccess/!msgTstStatusAck → ?msgLinkSumErr/!msgLinkSumNack
8. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ \_/!msgBeginVerify → evLnkAlloc → evLnkDealloc/\_ → evLnkAlloc  
→ ?msgLinkSumErr/!msgLinkSumNack  
→ evLnkDealloc/\_ → ?msgLinkSumErr/!msgLinkSumNack
9. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ ?msgLinkSumOK/!msgLinkSumAck → ?msgLinkSumErr/!msgLinkSumNack
10. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ ?msgLinkSumOK/!msgLinkSumAck → ?msgBeginVerifyAck/!msgTest  
→ ?msgTstSuccess/!msgTstStatusAck → ?msgLinkSumOK/!msgLinkSumAck  
→ evLnkAlloc → ?msgLinkSumErr/!msgLinkSumNack → evLocalizeFail/\_
11. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ \_/!msgLinkSum → ?msgLinkSumErr/!msgLinkSumNack
12. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ \_/!msgLinkSum → ?msgBeginVerifyAck/!msgTest  
→ ?msgTstSuccess/!msgTstStatusAck → \_/!msgLinkSum  
→ evLnkAlloc/\_ → evLocalizeFail/\_ → \_/!msgBeginVerify
13. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
→ ?msgLinkSumNack/\_
14. ?msgBeginVerifyAck/!msgTest → ?msgTstStatusFailure/!msgTstStatusAck  
→ \_/!msgBeginVerify → ?msgBeginVerifyAck/!msgTest

Prefix shortest path from the initial state “Down” to the test sequences generated

by Algorithm 3, the test sequences that cover all feasible feature combinations with approximate minimum number of executions are listed below:

1. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
 → evLnkAlloc/\_ → ?msgLinkSumErr/!msgLinkSumNack  
 → evLocalizeFail/\_ → \_/!msgBeginVerify
2. ?msgBeginVerifyAck/!msgTest → ?msgTstSuccess/!msgTstStatusAck  
 → ?msgLinkSumErr/!msgLinkSumNack → \_/!msgBeginVerify

When LDP is invoked, it is expected that a data link in “Up/Free” state can be allocated and changed to “Up/Alloc” state. When Link Property Correlation is invoked in a system, it is expected that if two ends of a data link have incompatible property, the data link can be changed to “Down” state automatically. By executing test sequence 1 generated by Algorithm 3, a user can find that the following sequence occurs:  
 “Down” → ?msgBeginVerifyAck/!msgTest → “Test” → ?msgTstSuccess/ !msgTstStatusAck → “Up/Free” → evLnkAlloc/\_ → “Up/Alloc” → ?msgLinkSumErr/ !msgLinkSumNack → “Up/Alloc”

This means that when LDP is invoked, LPC can not behave as expected, that is, it can not change a data link state to “Down” when two ends of the data link have incompatible properties. Interactions between LDP and LPC can be detected. Test sequence design and execution can not detect feature interactions in a system by itself. It must be combined with some assertion prove techniques. That is, to execute the designed test sequences to check if a certain property assertion is correct. When some assertions fail to hold, feature interactions can be detected. Using designed test sequences to check property assertions is more simple and more applicable to an implementation than only use verification techniques.

## 5 Conclusion

In this paper, an optimization technique is proposed to generate test sequences for communication protocols with an emphasis on feature interaction testing. A protocol may include several features and these features can be implemented in multiple processes such that their operations can interleave with each other. Whether the features in the implemented system can interact correctly needs to be tested. We use dual graph of an FSM to identify all feasible interleaving operations of different features, propose two algorithms to generate test sequences that cover all possible operation sequences with minimum number of executions. With the protocol modelled as a determined finite-state machine, the approach can be used for other applications such as inter-operability testing, fault detection. Some state proving techniques such as UIO sequences can be combined to make the algorithm more powerful and practical.

## Acknowledgement

We are indebted to the colleagues in Bell Labs Research China for the valuable comments and stimulating discussions.

## References

- [1] Xiaotao Wu, Henning Schulzrinne, "Feature Interactions in Internet Telephony End Systems", Technical Report, Department of Computer Science, Columbia University, January 2004.
- [2] Caixia Chi, Dong Wang, Ruibing Hao, "A Framework on Feature Interactions in Optical Network Protocols", Feature Interaction Workshop'2003, June 2003.
- [3] J.C. Godskesen, "A Formal Framework for Feature Interaction with Emphasis on Testing", Feature Interactions in Telecommunications Systems, vol. 3, K.E. Cheng and T. Ohta, eds., pp. 21C30, Amsterdam, IOS Press, Oct. 1995.
- [4] B. Kelly, M. Crowther, J. King, R. Masson, and J. DeLapeyre, "Service Validation and Testing", Feature Interactions in Telecommunications Systems, vol. 3, K.E. Cheng and T. Ohta, eds., pp. 173C 184, Amsterdam, IOS Press, Oct. 1995.
- [5] B. Kelly, M. Crowther, and J. King, "Feature Interaction Detection Using SDL Models", Proc. IEEE GLOBECOM 94, pp. 1,857C1,861, Nov. 1994.
- [6] M. Faci, Detecting Feature Interactions in Telecommunications Systems Designs, PhD thesis, Univ. of Ottawa, Canada, 1995.
- [7] David Lee, Mihalis Yannakakis, "Principles and Methods of Testing Finite State Machines - A Survey", Proceedings of the IEEE, Vol.84,No.8, August 1996.
- [8] T.S.Chow, "Testing software design modeled by finite-state machines", IEEE Trans. Software Eng. Vol.SE-4,No.3, pp.178-187, 1978.
- [9] K.K.Sabnani and A.T.Dahbura, "A protocol test generation procedure", Computer Networks and ISDN Syst. Vol.15, No.4, pp285-297, 1988.
- [10] S.Naito and M.Tsunoyama, "Fault detection for sequential machines by transitions tours", in Proc.IEEE Fault Tolerant Comput. Symp., IEEE Computer Soc.Press, pp.238-243, 1981.
- [11] Alfred V.Aho, Anton T.Dahbura, David Lee, and M.Ümit Uyar, "An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours", IEEE Tran. on Communications, Vol.39, NO.11, Nov.1991, 1604-1615.
- [12] Jonathan P. Lang, "Link Management Protocol (LMP)", Internet draft, draft-ietf-ccamp-lmp-10.txt, October 2003, work in progress.
- [13] T. H. Cormen, C. E. Leiserson and R. L. Rivest, Introduction to Algorithms. The MIT Press, 1997.
- [14] D Lee, M Yannakakis. "Pythia: a Software Tool for Testing Data Portions of Network Protocols", Technical Memo, Bell-Labs, Lucent Technologies, 1999.
- [15] J.A.Bondy and U.S.R.Murty, Graph Theory With Applications. New York: Elsevier North Holland, 1976.
- [16] A.Gibbons, Algorithmic Graph Theory. Cambridge, MA: Cambridge University Press, 1985.
- [17] Ayan Banerjee, et al. "Generalized Multiprotocol Label Switching: An Overview of Signaling Enhancements and Recovery Techniques", IEEE Communication Magazine, vol. 39, No.7, pp.144-151, July 2001.