

Test Generation for Interaction Detection Purpose

Caixia Chi Ruibing Hao

Lucent Technologies, Bell Labs

Tel: 8610-68748088, ext.8354/8581

Fax: 8610-68748226

E-mail: chic@lucent.com, rhao@research.bell-labs.com

Abstract

This paper proposes a technique to generate test sequences for checking the conformance of a system implementation to its specification, as well as to detect the interactions between the features of the system. A concept *color span* is defined to measure the extent of the interactions between different features. A modified *Chinese postman tour* algorithm is proposed to give an approximate minimum-cost and minimum *color span* tour of the transition graph of a finite-state machine.

I Introduction

A complex communication system can offer many services. Different services may interfere with each other, which results 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, but an incoming call is always rejected. The interaction between automatic call answering and call rejection occurs. Interactions of different features must be tested to ensure that a feature works correctly in a single feature environment also works as expected in an integrated multi-feature system.

Typically, an implementation of a system is tested for conformance by applying an external tester, a sequence of inputs and verifying that the corresponding sequence of outputs is that which is expected [2]. This paper describes a technique for generating optimal test sequences for an implementation of a system with an emphasis on detecting the system malfunctions resulted by interactions of different features. As stated in [2], a protocol can be specified as a deterministic finite-state machine (FSM), conformance test can be to present a method to test whether there is a discrepancy between the specification and implementation of an FSM. Lots of work has been done on generating test sequences for FSM's, but no efficient method has been proposed to generate sequences to test feature interactions in a system. With the richness of features in modern communication systems, testing the interactions of these features to guarantee the reliability and usability of whole system becomes more important.

The mechanism proposed in this paper tries to interleave the operations of different features as much as possible such that interactions between different features can be tested. A concept *color span* is defined in this paper to specify the interleaving extent of multiple features, then an optimization technique is used to find test sequences with minimum color span such that transitions from different features interleave with each other as much as possible in order to test interactions between different features.

In section II, some preliminaries knowledge on graph theory and finite automata theory is introduced. Section III describes the algorithm to generate test sequence minimal in time and with the minimum color span, section IV extends the algorithm to generate test sequence with minimum color span. Section V, the

algorithm is applied to generate test sequence for Internet telephony end systems. Section VI, a conformance test sequence is given to test Link Management Protocol [14]. The paper concludes in section VII.

II Preliminaries

II-A 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 an unique representation.

A *walk* in G is a finite, non-null sequence of consecutive edges: $W = (v_{i_1}, v_{i_2}; L_1)(v_{i_2}, v_{i_3}; L_2) \cdots (v_{i_{r-1}}, v_{i_r}; L_{r-1})$. 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[3]. 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 [4].

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*.

II-B 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 [2]. In this paper, it is assumed that G is strongly connected.

The following symbols are introduced in [2] 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 $c_e \in C$.

III Problem Statement

III-A Mathematical Model

In an implementation, features of a complex communication system are often implemented in different processes or invoked by different rules, thus concurrent operation of these features are inevitable. Interleaving the operations of different features and invoking features in different sequences may result in some intricate interaction problems. Previous research efforts have been mostly focused on the general conformance testing problem, whose purpose is to establish the confidence that a given implementation is in compliance with every function/feature description of a specification. It emphasizes on checking the compliance of individual feature of a system. However, many field observations have shown that 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. There is little work on systematically providing test sequences to test the interactions between the features.

In this work, we study the test sequence generation problem with a stress on testing the interactions between different features. We propose an algorithm to generate test sequences with requests from different features interleaving with each other as much as possible. A parameter to measure the interleaving extent of features in a test sequence is defined at first, then the test sequence generation problem is defined as an optimization problem which strives to maximize the interleaving extent of features, and finally an algorithm is proposed to solve the problem.

For a given walk W of a colored graph G , the associated *color sequence* of W is denoted as $CS(W) = c_{i_1}, c_{i_2}, \dots, c_{i_{r-1}}$, where c_{i_j} is the color assigned to edge $e_j = (v_{i_j}, v_{i_{j+1}}; L_j)$ in G .

For an edge e_j in W , its *colorspan* in W , $s_W(e_j)$, is defined as the length of the longest same color sub-walk in W starting with e_j . For example, if $W = (v_1, v_2; l_1)(v_2, v_3; l_2)(v_3, v_4; l_3)(v_4, v_5; l_4)$ and the color sequence of W is $CS(W) = c_1, c_1, c_1, c_2$, according to this definition, the *colorspan* of edge $e_1 = (v_1, v_2; l_1)$ in W is 3 and the *colorspan* of edge $e_3 = (v_3, v_4; l_3)$ in W is 1. Based on the *colorspan* definition of edges in a walk, we can also give the *colorspan* definition for a walk. The *colorspan* of a walk W is defined as the maximum of all the edge *colorspans* in the walk, $s(W) = \max\{s_W(e_i), e_i \in W\}$. If all edges in a walk are of the same color, the *colorspan* of the walk is the length of the walk. The longest same color sub-walks in a walk W are also called the *Critical sections* of W .

For example, given a walk $W = e_1, e_2, e_3, e_4, e_5, e_6, e_7$, and its color sequence $CS(W) = c_1, c_1, c_1, c_1, c_2, c_2, c_3$, the *colorspan* of W is 4 and the *Critical section* of W is e_1, e_2, e_3, e_4 . If all edges in W are of the same color, e.g., $CS(W) = c_1, c_1, c_1, c_1, c_1, c_1, c_1$, then $s(W) = 7$.

As we described earlier, for a state machine M , transitions resulted from different features are assigned with different colors. Given a test sequence consisting of the edges in M , the *colorspan* of the test sequence reflects the interleaving extent of features. The larger the *colorspan*, the less the interleaving. Thus different from the traditional test sequence generation problem, which tries to find a Chinese postman tour, we need to find a postman tour which has the minimum *colorspan*. In summary, the problems we need to solve are as follows:

Problem 1: Given a colored digraph $G = (V, E, C)$, find a postman tour T such that $|T|$ and $s(T)$ are all minimized, where $|T|$ is the number of edges in T and $s(T)$ is the *colorspan* of T .

Problem 2: Given a colored digraph $G = (V, E, C)$, find a postman tour T such that $s(T)$ is minimized.

The first problem is to find a Chinese postman tour such that $s(T)$ is minimized, while the second problem only has one object that is to minimize $s(T)$. For the optimal solution T to problem 1 and T' to problem 2,

it is obvious that $s(T') \leq s(T)$, $|T| \leq |T'|$.

III-B Algorithm for Problem 1

If the colored digraph G is an Eulerian graph, the Problem 1 is reduced to find an Eulerian tour T in G such that its *colorspan* $s(T)$ is minimized. If G is not an Eulerian graph, from [6] we know that G must have un-balanced vertex, and the number of the un-balanced vertex must be even. For any postman tour of G , some edges are traversed more than once. Suppose that a postman tour T passes edge $e_{ij} = (v_i, v_j)$, k_{ij} times, we add $k_{ij} - 1$ new edges between v_i and v_j and associate each new edge with the same color as e_{ij} , these new edges are called the augmented edges of e_{ij} . The resulted augmented graph is denoted as \tilde{G} , then \tilde{G} is an Eulerian Graph, and T is an Eulerian tour of \tilde{G} , apparently, $s(T)$ is determined by the color of the newly added edges and the way to form the tour.

To solve Problem 1, we need augment graph G to guarantee the existence of an Euler tour and then find a tour with a minimum *colorspan* from the augmented graph. The solution can be described as the following steps:

step 1. Get $E_1 \subset E$ in G with the condition that when G is augmented with only edges in E_1 , the new graph \tilde{G} has an Eulerian Tour.

step 2. On the condition that (1) is satisfied, choose E'_1 that has a minimal $|E_1|$.

step 3. In the \tilde{G} generated using E'_1 , find an Eulerian tour T , such that $s(T)$ is minimized.

When an edge set satisfies condition (1), it is referred to as the feasible augment edge set of G . When an edge set satisfies condition (1) and (2), it is referred to as optimal augment edge set of G . If a tour satisfies (3), it is called an optimal tour.

[6] gives out an algorithm to find an optimal augment edge set for G in polynomial time, so we only need to find an optimal tour on the Eulerian graph \tilde{G} . In the following we will show that to find such an optimal tour on \tilde{G} is a NP-C problem.

Theorem 1 *For a given balanced graph $G = (V, E, C)$, and an integer $k \leq |E|$, deciding if there is a tour T in G that traverses each edge once and has a colorspan $s(T) \leq k$ is a NP-C problem.*

Proof. For a given colored digraph $G = (V, E, C)$, a dual graph $\hat{G} = (U, A, \hat{C})$ can be constructed according to the following steps:

Step 1: $U = E$, that is, each edge in G corresponds to a node in \hat{G} . Let $u_i \in U$ corresponds to $e_i \in E$.

Step 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 , then $(u_i, u_j) \in A$.

Step 3: \hat{C} has an initial value NULL. For any $(u_i, u_j) \in A$, if $c(e_i) = c(e_j)$ in G , that is, e_i and e_j have the same color in G , then $\hat{c}(u_i, u_j) = 1$, $\hat{C} = \hat{C} \cup \{1\}$; otherwise, $\hat{C}(u_i, u_j) = n, n \in N^+$ and $n \notin \hat{C}$, $\hat{C} = \hat{C} \cup \{n\}$.

By above construction, if two adjacent edges in G have different colors, their corresponding nodes are connected in \hat{G} by an edge with a color different from all the other colors assigned to edges in A .

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

Finding a tour T in G that traverses each edge exactly once and has $s(T) \leq k$ is equivalent to finding a simple path p in \hat{G} that passes each node in \hat{G} exactly once and has $s(p) \leq k - 1$. As a special case of this problem, if all edges in G have the same color, the problem is equal to finding a travelling salesman path in \hat{G} , which is known to be a NP-C problem. So the original problem is also NP-Complete. ■

An optimal tour for a balanced digraph can be found in time complexity $O(2^m)$ using dynamic programming techniques. In the following we give a heuristic algorithm which has a time complexity of $O(m)$.

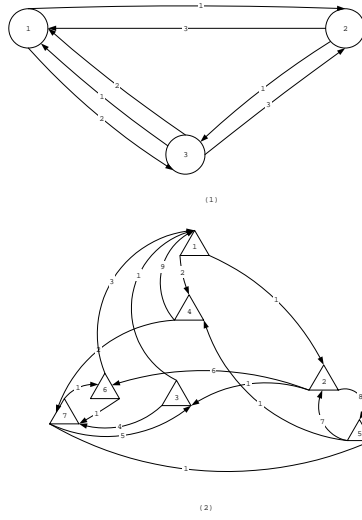


Figure 1: A Prime Graph G and its Dual Graph \hat{G}

Algorithm 1: Find an Eulerian Tour with an approximately minimal *colorspan* on a balanced digraph.

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

Input: v /* starting node for the circuit */

Output: An Eulerian Tour with an approximately minimal *colorspan*.

begin

1. $i = 0$.
2. Get an arbitrary vertex $v_1 \in V$, call $getCircuit(v_1, G)$ to get a circuit T_i with v_1 as the beginning and ending vertex.
3. $G = G - T_i$, if $G = NULL$, stop and return T_i as the optimal tour.
4. Otherwise, arbitrarily select a vertex v in T_i such that $d_G^+(v) \geq 1$
5. call $getCircuit(v, G)$ to find a circuit C starting from and ending at v ;
6. replace v in T_i with C to get T_{i+1} ;
7. $i := i + 1$; Go to step 3.

end

Procedure $getCircuit(v, G)$

Output: A circuit in G that begins with and ends at v with an approximately minimal *colorspan*.

begin

1. Let $\bar{E} = \{e_1, e_2, \dots, e_k\}$ be the set of all edges in G .
2. Arbitrarily select an edge $e = (v, v') \in \bar{E}$, $T = e$, $s = 1$;
/* variable s is used to record the *colorspan* of walk T */
3. $\bar{E} := \bar{E} \setminus e$, $v_1 = v$, $v_2 = v'$;
4. If $v_2 = v$, return T ;
5. If $\exists e' = (v_2, v'') \in \bar{E}$ such that the color of e' is different from (v_1, v_2)
6. $T = T \cdot e'$.
7. otherwise arbitrarily select an edge $e' = (v_2, v'')$ from \bar{E}
8. $T = T \cdot e'$, $s = s + 1$;

9. $e = e'$, Goto step 3;

end

Combining the algorithm given in [6] and algorithm 1, we give the algorithm for Problem 1.

Algorithm 2: Solution to Problem 1

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

Output: A shortest Postman Tour with an approximately minimal *colorspan*.

begin

1. For each $v_i \in V$, set $\sigma_i = d^-(v_i) - d^+(v_i)$.
2. If $\sigma_i = 0, i = 1 \dots n$, then set $\tilde{G} = G$ goto step 7; Otherwise,
3. Let $S = \{v_i \in V | \sigma_i > 0\}$, $T = \{v_j \in V | \sigma_j < 0\}$. $\forall v_i \in S, \forall v_j \in T$, find the shortest path from v_i to v_j ;
4. Construct a complete bi-partite graph H , with $X = \{x_{i,p} | v_i \in S, p = 1, 2, \dots, \sigma_i\}$,
 $Y = \{y_{j,q} | v_j \in T, q = 1, 2, \dots, |\sigma_j|\}$, $E(H) = \{x_{i,p}y_{j,q} | x_{i,p} \in X, y_{j,q} \in Y\}$.
 Associates each edge $x_{i,p}y_{j,q}$ in H , $p = 1, 2, \dots, \sigma_i$, $q = 1, 2, \dots, |\sigma_j|$, a weight $w(v_i, v_j)$,
 where $w(v_i, v_j)$ is the length of the shortest path between v_i and v_j in G .
5. 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.
6. For each edge $x_{i,p}y_{j,q} \in M$, suppose the shortest (originally it was minimum *colorspan*)
 path between v_i and v_j in G is $W_{i,j}$, add every edge in $W_{i,j}$ to G . Set the newly augmented
 balanced graph as \tilde{G} .
7. Call Algorithm 1 to find the Euler tour T in \tilde{G} such that $s(T)$ is minimized .
8. Return T .

end

III-C Solution to Problem 2

$s(T)$ depends on the color of the feasible edges added to graph G and the algorithm to find the Eulerian tour. Intuitively, the more the colors of the feasible edges vary, a tour with less $s(T)$ can be found. Based on such an intuition, we modify the process to find feasible edges for G such that the path formed by the feasible edges has a minimum *colorspan*.

Definition III.1 Given a graph $G = (V, E, C)$, the minimum *colorspan* path for node pair (v_i, v_j) is a path from node v_i to v_j with the minimum *colorspan*.

Algorithm 3 gives an algorithm to get the minimum *colorspan* paths from a given node s to all the other nodes in a graph. The complexity of the algorithm is $O(n^2m)$.

Algorithm 3: Find the Minimum *Colorspan* Paths.

Input: $G = (V, E, C), v$ /* a strongly connected colored digraph and a source node*/

Output: Minimal *colorspan* paths from node v to all the other nodes.

begin

1. $l(v) := 0, p(v) := v, F := \{v\}, M := V \setminus \{v\}$;
2. For any $v_j \in V$ and $v_j \neq v$
 $p(v_j) := NULL$;
 if $(v, v_j) \in E, l(v_j) := 1, a(v_j) := v$;

- otherwise $l(v_j) := \infty, a(v_j) := NULL$;
3. Select a node v_i in M such that $l(v_i) = \min_{v_j \in M} l(v_j)$.
 If $l(v_i) = \infty$, stop, no path between v_i and all the nodes in M ;
 Otherwise, $p(v_i) := p(a(v_i)) \cdot v_i, s(p(v_i)) := l(v_i)$.
 4. Set $F := F \cup \{v_i\}, M := M \setminus \{v_i\}$. If $M = \emptyset$, stop, the minimum *colorspan* paths from v to all the other nodes have been found; otherwise,
 5. For all $v_j \in M$, if $(v_i, v_j) \in E$ and $l(v_j) > s(p(v_i)) \cdot (v_i, v_j)$
 set $l(v_j) := s(p(v_i)) \cdot (v_i, v_j), a(v_j) := v_i$;
 6. Go to step 3.
- end**

In algorithm 2, when we augment the graph to find the shortest postman tour, we only search for a perfect matching in which the sum of weights of these augmented edges is minimized, and don't consider the *colorspan* of each augmented path. However, in problem 2, we care more about the *colorspan* of the final generated tour and the length of the tour is no longer an issue. An intuitive heuristic approach to problem 2 is when augmenting the original graph, we search for a matching which only uses paths with smaller *colorspan*. This is very similar to what so called the minimax matching problem.

Definition III.2 Given a bi-partite graph $H = (X, Y, E, W)$, M is a matching of H , let $\tilde{w}(M) = \max\{w_{ij} | x_i y_j \in M\}$. If H has a maximum matching M^* , whose $\tilde{w}(M^*) = \min\{\tilde{w}(M) | M \text{ is a maximum matching of } H\}$, then M^* is called the minimax matching of H .

The algorithm proposed in [5] can be modified to get the minimax matching of a bi-partite graph in time complexity $O(m^2)$.

Algorithm 4: Find the Minimax Matching.

Input: $H = (X, Y, E, W)$ /* Directed bi-partite graph H with assigned weight on each edge */

Output: Matching M of H such that $\max_{e \in M} \{w(e)\}$ is minimized.

begin

1. For any edge in H , assign a new weight $w'(e) = W - w(e)$,
 where W is a real number larger than any $w(e), e \in E(H)$;
2. Call maximin matching algorithm [5] to get the maximin matching M of H based on the new weight;
3. M is the minimax matching of original graph H .

end

If we use the *colorspan* as the weight for each edge in the bi-partite graph H , then Algorithm 4 can be used to find a matching whose edge's maximum *colorspan* is the minimal among all the matchings.

Combining algorithm 1, 3 and 4, we give a heuristic algorithm for problem 2 with complexity $O(m^2 + n^2m)$.

Algorithm 5: Heuristic Solution to Problem 2

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

Output: A Postman Tour with an approximately minimal *colorspan*.

begin

1. For each $v_i \in V$, set $\sigma_i = d^-(v_i) - d^+(v_i)$.

2. If $\sigma_i = 0, i = 1 \dots n$, then set $\tilde{G} = G$ goto step 7; Otherwise,
 3. Let $S = \{v_i \in V | \sigma_i > 0\}$, $T = \{v_j \in V | \sigma_j < 0\}$. $\forall v_i \in S, \forall v_j \in T$, find the minimal *colorspan* path from v_i to v_j using Algorithm 3;
 4. Construct a complete bi-partite graph H , with $X = \{x_{i,p} | v_i \in S, p = 1, 2, \dots, \sigma_i\}$, $Y = \{y_{j,q} | v_j \in T, q = 1, 2, \dots, |\sigma_j|\}$, $E(H) = \{x_{i,p}y_{j,q} | x_{i,p} \in X, y_{j,q} \in Y\}$. Associates each edge $x_{i,p}y_{j,q}$ in H , $p = 1, 2, \dots, \sigma_i$, $q = 1, 2, \dots, |\sigma_j|$, a weight $w(v_i, v_j)$, where $w(v_i, v_j)$ is the *colorspan* of the minimum *colorspan* path from v_i to v_j in G .
 5. Find the minimax match $M = \{e_1, e_2, \dots, e_k\}$ in H using Algorithm 4.
 6. For each edge $x_{i,p}y_{j,q} \in M$, suppose the minimum *colorspan* path between v_i and v_j in G is $P_{i,j}$, add every edge in $P_{i,j}$ to G . Set the newly augmented balanced graph as \tilde{G} .
 7. Call Algorithm 1 to find the Euler tour T in \tilde{G} such that $s(T)$ is minimized .
 8. Return T .
- end**

IV Optimal Test Sequence Generation For Internet Telephony End Systems

With supporting Session Initiation Protocol(SIP)[9], more and more services can be provided by Internet telephony end systems. Service creation in Internet telephony systems have been well discussed [10][12], and feature interactions in Internet telephony systems are also studied [1][11]. But these paper only give case-by-case study on some feature interaction problems and gives some classification of the feature interactions in Internet telephony systems. No systematic testing schemes are proposed to test the interactions between the features of Internet telephony systems. In the following, we use FSM to model an Internet telephony end system and apply our algorithm to generate test sequences for the end system.

IV-A FSM Model of Internet Telephony End Systems

An Internet telephony end system can support many services and as a case study, we suppose an end system supports the services discussed in [1], which include basic call functions such as call,accept,and disconnect, and also include call reject, redirect, transfer, hold, and mute.

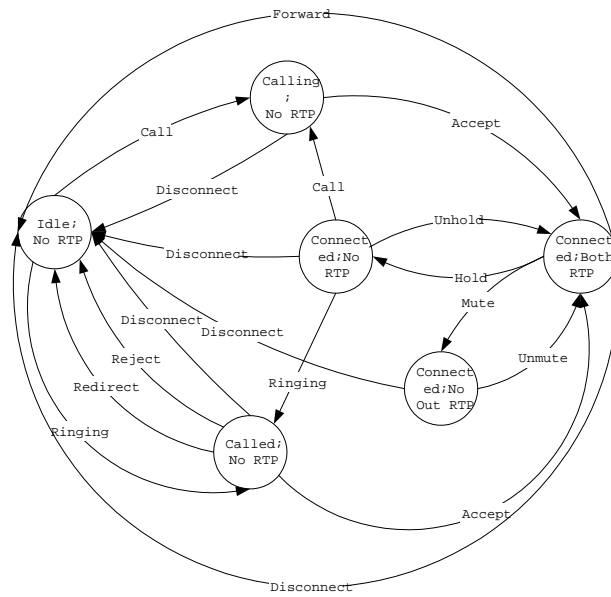
Figure 2 gives the FSM of an Internet telephony end system. With an end system being the only entity where signaling and media flows are guaranteed to converge, the state of end system should include both control state and media state. Each state of the end system is denoted as (Control State; Audio State), that is the combination of signaling state and audio state. The basic call functions include the following actions: call, ring,accept and disconnect. The actions of hold feature include: hold and unhold; the actions of mute feature include:mute and unmute.

Figure 3 gives the augmented graph of Figure 2. Label on each transition $c : a$, where c is the color assigned to the transition, a is the action that results in this transition.

From algorithm 1, we get Euler Tour T for Fig.3:

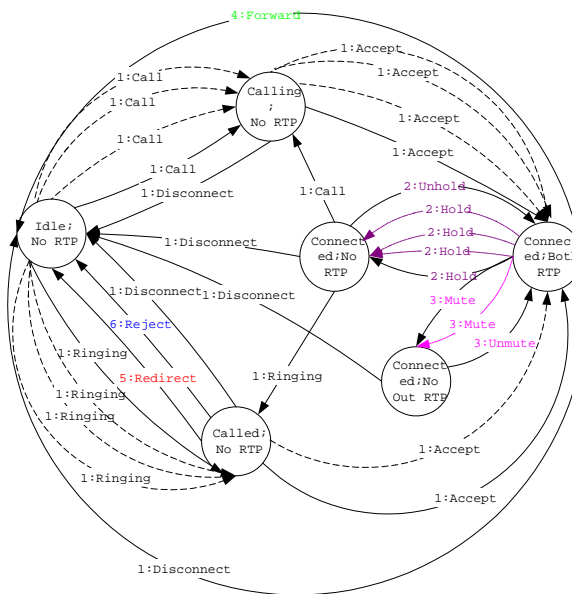
Call \rightarrow Accept \rightarrow Hold \rightarrow Disconnect \rightarrow Call \rightarrow Accept \rightarrow Hold \rightarrow Call \rightarrow Disconnect \rightarrow Call \rightarrow Accept \rightarrow Hold \rightarrow Ring \rightarrow Reject \rightarrow Call \rightarrow Accept \rightarrow Hold \rightarrow Unhold \rightarrow Mute \rightarrow Disconnect \rightarrow Ring \rightarrow Redirect \rightarrow Ring \rightarrow Accept \rightarrow Mute \rightarrow Unmute \rightarrow Forward \rightarrow Ring \rightarrow Accept \rightarrow Disconnect

Then $CS(T) = 112111211112161122311511334111$, and $s(T) = 4$.



Features: Basic Call Feature; Hold; Forward; Redirect; Reject.

Figure 2: Internet Telephony End System FSM



Color Assignment: 1: Basic Call Feature; 2: Hold; 3: Mute; 4: Forward; 5: Redirect; 6: Reject.

Figure 3: Augmented Graph for Internet Telephony End System FSM

V Optimal Test Sequence Generation For LMP

V-A 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 [13]. From the functional point of view, a GPMLS control plane can be partitioned into the following 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) [14]. Link Management Protocol (LMP), enhancements to OSPF/IS-IS, and enhancements to RSVP/CR-LDP are being standardized by the 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 [14].

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 [14]. 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 for the 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 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.

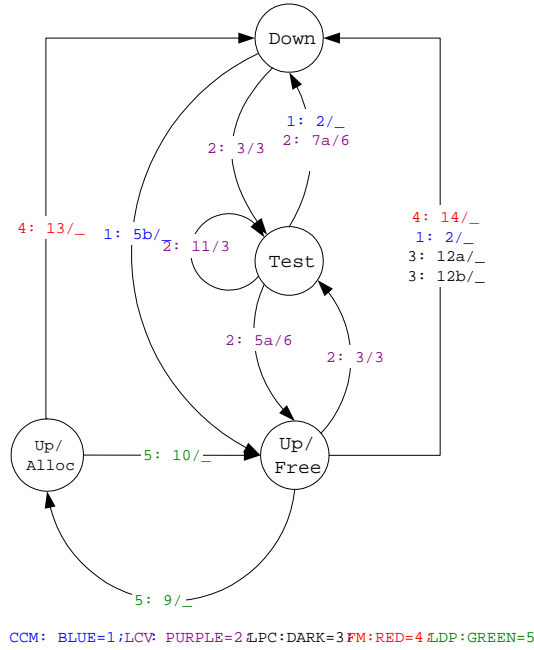


Figure 4: Active LMP Data Link FSM

In the following, we use the data link FSM of LMP as an example, applying algorithm 1 to provide a test sequence to guarantee that each transition is traversed at least once and the operations of different features are interleaved with each other as much as possible such that their interactions can be tested.

V-B Data Link Model of LMP

Fig. 4 shows the active LMP data link FSM and Fig.5 shows the passive LMP data link FSM. 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.

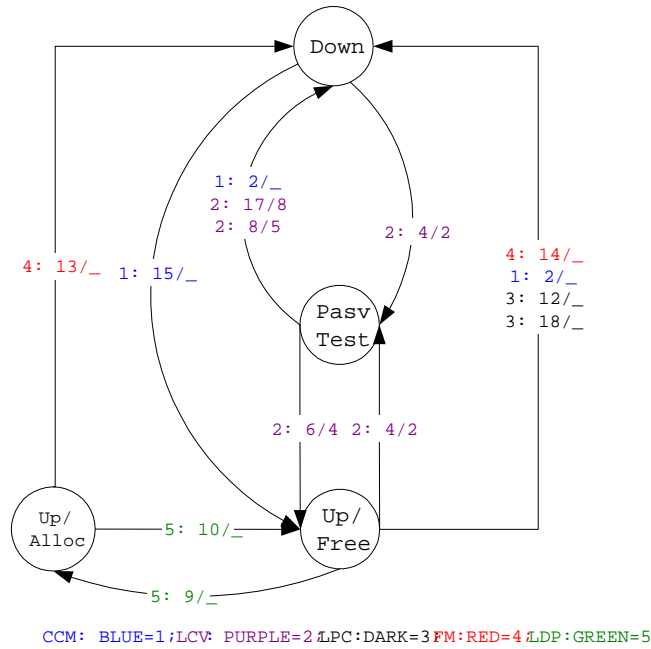


Figure 5: Passive LMP Data Link FSM

- 1 : evCCUp: Control channel has gone up.
- 2 : evCCDown: LMP neighbor connectivity is lost.
- 3 : evStartTst: ?BeginVerifyAck.
- 4 : msgBeginVerifyOK: ?BeginVerify.
- 5 : msgTstSuccess: ?TestStatusSuccess.
- 6 : evTestRcvOK: ?Test message.
- 7 : msgTstStatusFailure: ?TestStatusFailure.
- 8 : evPsvTestFail: VerifyDeadInterval has expired.
- 9 : evLnkAlloc: Allocate the data link.
- 10: evLnkDealloc: Deallocate the data link.
- 11: evTestRet: A retransmission timer expires.
- 12: msgLinkSumErr: ? error LinkSummary.
- 13: evLocalizeFail: FM localizes a Failure
- 14: evDlDown: The data link is down
- 15: inBandConfigOK: Link is ready for path establishment.
- 16 : evTstFail: Verification fails.
- 17: msgEndVerify: ?EndVerify message.
- 18: msgLinkSumNack: ? LinkSummaryNack.

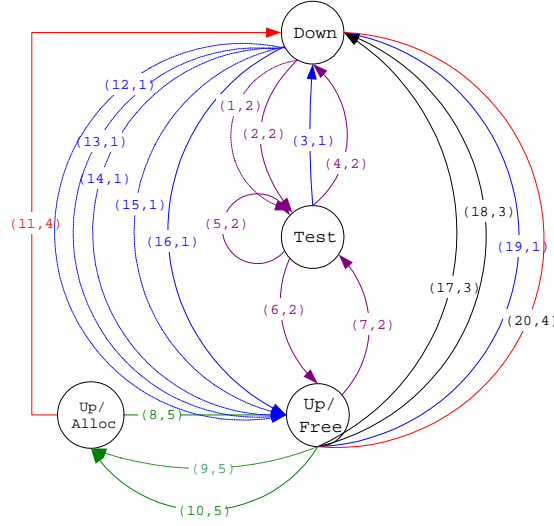


Figure 6: Symmetric Augmentation of Active LMP Data Link FSM

Output:

- 1 : !msgBeginVerify
- 2 : !msgBeginVerifyAck
- 3 : !msgTest
- 4 : !msgTestSuccess
- 5 : !msgTestFailure
- 6 : !msgTstStatusAck
- 7 : !msgEndVerify
- 8 : !msgEndVerifyAck
- 9 : !msgBeginVerifyNack
- 10: !msgLinkSumNack

V-C Optimal Test Sequence for LMP

Fig.6 shows the balanced augmentation of active LMP data link FSM, Fig.7 gives the balanced augmentation of passive LMP data link FSM, they are get from the algorithm in [6]. In Fig.6 and Fig.7, the dashed links are the optimal links added to Fig.4 and Fig.5. There is a label (l, c) on each link, l is a label assigned to the link and c is the color of the link.

From algorithm 1, we get Euler Tour T_1 for Fig.6:

$$T_1 = 1, 3, 2, 4, 12, 9, 11, 13, 10, 8, 17, 14, 18, 15, 20, 16, 7,$$

5, 6, 19. Then $CS(T_1) = 21221541553131412221$, and $s(T_1) = 3$. The following gives the test sequence get from tour T_1 .

evStartTst/!msgTest \rightarrow evCCDown/_ \rightarrow evStartTst/!msgTest \rightarrow ?msgTstStatusFailure/!msgTstStatusAck \rightarrow inBandConfigOK/_ \rightarrow evLnkAlloc/_ \rightarrow evLocalizeFail/_ \rightarrow inBandConfigOK/_ \rightarrow evLnkAlloc/_ \rightarrow evLnkDeal-
loc/_ \rightarrow ?msgLinkSumErr/!msgLinkSumNack \rightarrow inBandConfigOK/_ \rightarrow ?msgLinkSumNack/_ \rightarrow inBandCon-
figOK/_ \rightarrow evDlDown/_ \rightarrow inBandConfigOK/_ \rightarrow evStartTst/!msgTest \rightarrow evTestRet/!msgTest \rightarrow ?msgTst-
Success/!msgTstStatusAck \rightarrow evCCDown/_

The Eulerian Tour T_2 for Fig.7 is:

$$T_2 = 1, 6, 9, 2, 7, 8, 17, 10, 13, 11, 14, 12, 15, 16, 18, 3, 19,$$

4, 20, 5, 21. $CS(T_2) = 15415521222221142323$, $s(T_2) = 5$. The test sequence get from tour T_2 is:

inBandConfigOK/_ \rightarrow evLnkAlloc/_ \rightarrow evLocalizeFail/_ \rightarrow inBandConfigOK/_ \rightarrow evLnkAlloc/_ \rightarrow evLnkDeal-

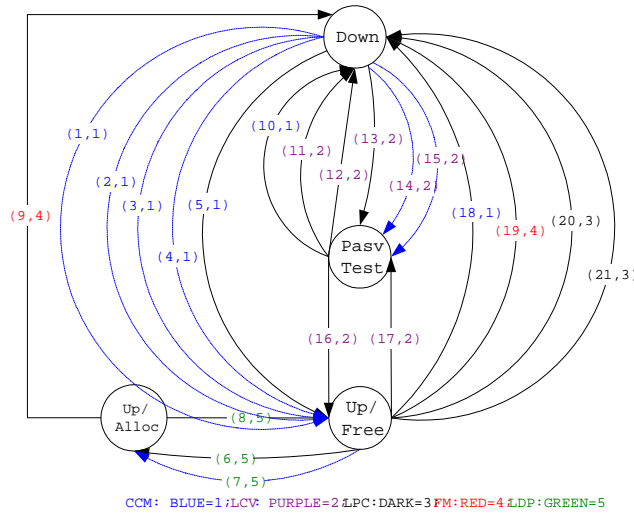


Figure 7: Symmetric Augmentation of Passive LMP Data Link FSM

loc/_ → msgBeginVerifyOK/!msgBeginVerifyAck → evCCDown/_ → msgBeginVerifyOK/!msgBeginVerifyAck
→ evPsvTestFail/!msgTestFailure → msgBeginVerifyOK/!msgBeginVerifyAck → msgEndVerify/!msgEndVerifyAck
→ msgBeginVerifyOK/!msgBeginVerifyAck → evTestRcvOK/!msgTestSuccess → evCCDown/_ → inBand-
ConfigOK/_ → evDIDown/_ → inBandConfigOK/_ → msgLinkSumErr → inBandConfigOK/_ → msgLinkSum-
Nack

VI Conclusion

In this paper, a technique is proposed to generate optimal conformance test sequences for complex systems. A complex system may include several features and these features can be implemented in multiple processes such that their operations can interleave with each other. Whether or not the implemented system can work correctly when such an interleaving occurs needs to be tested. We define a parameter *color span* to measure the extent of the interactions between different features, propose an algorithm to find test sequences with minimum length and minimum color span such that all the transitions of the system FSM are traversed at least once and the features of the system are interleaved with each other as much as possible.

With the protocol being modelled as a finite-state machine, the approach can be used for many 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.

References

- [1] Xiaotao Wu, Henning Schulzrinne, “Feature Interactions in Internet Telephony End Systems”, Technical Report, Department of Computer Science, Columbia University, January 2004.
- [2] 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.
- [3] T. H. Cormen, C. E. Leiserson and R. L. Rivest, Introduction to Algorithms. The MIT Press, 1997.
- [4] J.A.Bondy and U.S.R.Murty, Graph Theory With Applications. New York: Elsevier North Holland,1976.

- [5] Gross O. The bottleneck assignment problem: an algorithm. In: Proceedings, and Symposium on Mathematical Programming(Wolfe Ped), Rand Publication, 1960,87-88.
- [6] A.Gibbons, Algorithmic Graph Theory. Cambridge, MA:Cambridge University Press,1985.
- [7] J.Edmonds and E.L.Johnson, "Matching,Euler tours and the Chinese postman", Math.Program.,vol.5,pp.88-124,1973.
- [8] Michael R.Garey and David S.Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness,"W.H.FREEMAN AND COMPANY New York.
- [9] J. Rosenberg, Henning Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: session initiation protocol. RFC 3261, Internet Engineering Task Force, June 2002.
- [10] John de Keijzer, Douglas Tait, and Rob Goedman. "JAIN: a new approach to services in communication networks". IEEE Communications Magazine, 38(1), January 2000.
- [11] Jonathan Lennox and Henning Schulzrinne. "Feature interaction in Internet telephony". In Feature Interaction in Telecommunications and Software Systems VI, Glasgow, United Kingdom, May 2000.
- [12] J. Rosenberg, J. Lennox, and Henning Schulzrinne. "Programming Internet telephony services". IEEE Network, 13(3):42C49, May/June 1999.
- [13] 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.
- [14] Jonathan P. Lang, "Link Management Protocol (LMP)", Internet draft, draft-ietf-ccamp-lmp-10.txt, October 2003, work in progress.