Test Generation for Interaction Detection in Feature-rich Communication Systems

Caixia Chi and Ruibing Hao

Bell Labs Research China, Lucent Technologies, Beijing, China, 100080,
chic@lucent.com, rhao@research.bell-labs.com

Abstract. This paper proposes a technique to generate test sequences to check the conformance of an implementation of a feature-rich communication system to its specification, as well as to detect the interactions between the features of the system. A concept called *color span* is introduced to measure the extent of the interactions between different features. A modified *Chinese postman tour* algorithm is proposed to produce an approximate minimum-cost and minimum *color span* tour of the transition graph of a finite-state machine. Test generation using the proposed algorithm for the SIP-based Internet telephony end system and for the Link Management Protocol are reported.

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. In [2], we have identified some feature interactions in the protocols for core optical networks. Interactions between features of a communication system are usually caused by different reasons such as resource sharing or 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 identify feature interactions in a real system by means of testing.

As stated in [3], a communication protocol system 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 the implementation of an FSM. 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[3]. Lots of work has been done on generating test sequences for FSM's [4][5][6][7], and all these work focus on finding a tour of the transition graph of the FSM that meets certain coverage criteria, such as a transition tour [7], or a postman tour [3].

With the richness of features in modern communication systems, it is important to make sure that a feature that works correctly in a stand alone mode also works as expected in an integrated multi-feature system. Thus to guarantee the reliability and usability of whole system, identifying the interactions between features becomes more important. Generating conformance test cases that at the same time can detect feature interactions in a feature-rich communication system can be an efficient way to achieve this goal. Unfortunately we have not seen any previous work on this aspect.

This paper describes a technique for generating optimal test sequences for an implementation of a feature-rich communication system with an emphasis on detecting the system's malfunctions resulted by interactions between different features. 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 called *color span* is introduced 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 preliminary 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* only. In Section V, the algorithm is applied to generate test sequence for the SIP-based Internet telephony end systems. In Section VI, we report the conformance test sequence generated for the Link Management Protocol [15]. The paper concludes in section VII.

2 Preliminaries

2.1 Graphs

Let $G = (V, E)$ be a labelled directed graph with vertex set V, edge set E, where $V = \{v_1, \cdots, v_n\}$ 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, L_e, v_j), 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:
$W = (v_{i_1}, L_1, v_{i_2})(v_{i_2}, L_2, v_{i_3}) \cdots (v_{i_{r-1}}, L_{r-1}, v_{i_r})$. 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[8]. 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 v_i as the origin and v_j as the tail[9].

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 $V = \{v_1, \cdots, v_n\}$ represents the specified states of the FSM and a directed edge represents a transition from one state to another in the FSM [3]. In this paper, it is assumed that G is strongly connected.

The following symbols are introduced in [3] and we include them 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 the 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, L, v_j), 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 $(v_i, L_1, v_j) \in E$, and $(v_i, L_2, v_k) \in E$, $L_1^{(i)} \neq L_2^{(i)}$, 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 $V = \{v_1, \cdots, v_n\}$ and $n = |V|$,$m = |E|$, $C = \{c_1, \cdots, c_k\}$. Each edge $e \in E$ is assigned a color $c_e \in C$.

3 Problem Statement

3.1 Mathematical Model

In an implementation, features of a complex communication system are often implemented in different processes or invoked according to different rules, thus concurrent operation of these features are inevitable. Interleaving the operations of different features and invoking features in different orders 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 generating 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 stated 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 $CS(W) = c_{i_1}, c_{i_2}, \cdots, c_{i_{r-1}}$, where c_{i_j} is the color assigned to edge $e_j = (v_{i_j}, L_j, v_{i_{j+1}})$ in G.

For an edge e_j in W, its *color span* 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 $W = (v_1, l_1, v_2)(v_2, l_2, v_3)(v_3, l_3, v_4)(v_4, l_4, v_5)$ and the color sequence of W is $CS(W) = c1, c1, c1, c2$, according to this definition, the *color span* of edge $e_1 = (v_1, l_1, v_2)$ in W is 3 and the *color span* of edge $e_3 = (v_3, l_3, v_4)$ in W is 1. Based on the *color span* definition of edges in a walk, we can also give the *color span* definition for a walk. The *color span* 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 *color span* 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 have 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 *color span* of the test sequence reflects the

interleaving extent of features. The larger the *color span*, 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 *color span*. 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 both minimized, where $|T|$ is the number of edges in T and $s(T)$ is the *color span* 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 optimization object that is to minimize $s(T)$. For the optimal solution T to problem 1 and the optimal solution T' to problem 2, it is easy to prove that $s(T') \leq s(T)$, $|T| \leq |T'|$.

3.2 Algorithm for Problem 1

If a colored digraph G is an Eulerian graph, Problem 1 is reduced to find an *Euler tour* T in G such that its *color span* $s(T)$ is minimized. If G is not an Eulerian graph, from [11] we know that G must have un-balanced vertex, and the number of such un-balanced vertex is even. For any *postman tour* of G, some edges are traversed more than once. Suppose that a *postman tour* T passes edge $e_{ij} = (v_i, v_j)$ for 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 *Euler 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 from the augmented graph find such a tour with the minimum *color span*. The solution can be summarized 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 *Euler tour*.
Step 2. On the condition that step 1 is satisfied, choose E_1' that has the minimum number of edges.
Step 3. For $\tilde{G}$, augmented from G using only edges in E_1', find an *Euler tour* T, such that $s(T)$ is minimized.

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

In [11], the author gives an algorithm to find the optimal augment edge set for G in polynomial time, so we only need to find an optimal tour on the augmented

Eulerian graph $\tilde{G}$. In the following we will show that to find such an optimal tour on $\tilde{G}$ is a NP-Complete 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 color span $s(T) \leq k$ is a NP-Complete 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 in G, then $(u_i, u_j) \in A$, in $\hat{G}$.
Step 3: $\hat{C}$ has an initial value of 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 label on edges of Fig.1(1) is (l, c), where l is the the edge index and the c is the color assigned to the edge. The label edges of Fig.1(2) is the color assigned to the edge. Each edge index in Fig.1(1) corresponds to a vertex in Fig.1(2).

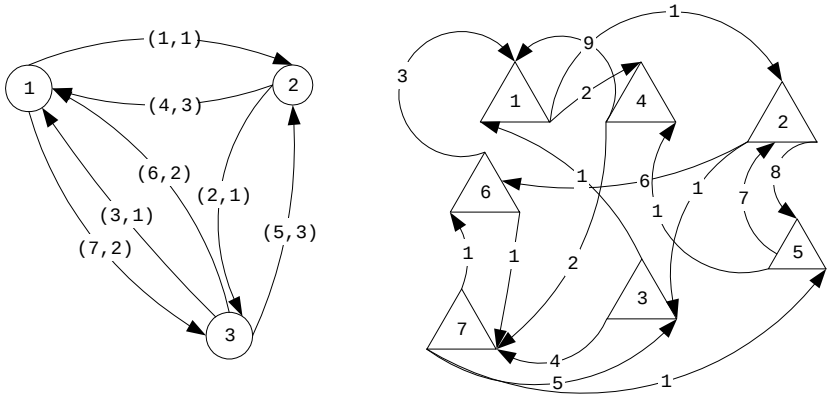


Fig. 1. A Prime Graph G (1) and its Dual Graph $\hat{G}$(2)

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 equivalent 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.

In the following we give a heuristic algorithm to find an *Euler tour* in G with an approximately minimum *color span* in time complexity $O(m)$, where m is the number of edges of G.

Algorithm 1: Find an *Euler tour* with an approximately minimal *color span* on a balanced digraph.
Input: $G = (V, E, C)$ /* a balanced colored digraph */
Output: An *Euler tour* with an approximately minimal *color span*.

begin
1. $i := 0$.
2. Get an arbitrary vertex $v_1 \in V$, call subroutine $getCircuit(v_1, G)$ 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 subroutine $getCircuit(v, G)$ to find a circuit C starting from and ending at v; Replace v in T_i with C to get T_{i+1};
6. $i := i + 1$; 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 *color span*.

begin
1. Let $\bar{E} := \{e_1, e_2, \cdots, e_k\}$ be the set of all edges in G.
2. Arbitrarily select an edge $e = (v, v') \in \bar{E}$, $T := e$;
3. $\bar{E} := \bar{E} \backslash e$, $v_1 := v, v_2 := v'$;
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. $T := T \cdot e'$; /* append edge e' to T */
7. otherwise arbitrarily select an edge $e' = (v_2, v'')$ from $\bar{E}$
8. $T := T \cdot e'$;
9. $e := e'$, Goto step 3;

end

Combining the algorithm to find an optimal augment edge set for G given in [11] and algorithm 1, we give a heuristic solution to Problem 1 in Algorithm 2.

Algorithm 2: A Heuristic Solution to Problem 1
Input: $G = (V, E, C)$ /* a strongly connected colored digraph */
Output: A shortest Postman Tour with an approximately minimal *color span*.

begin
1. For each $v_i \in V$, set $\sigma_i := d^-(v_i) - d^+(v_i)$.
2. If $\sigma_i = 0, i = 1 \ldots n$, then set $\tilde{G} := G$ goto step 7; Otherwise,
3. Let $S = \{v_i \in V | \sigma_i > 0\}$, $T = \{v_j \in V | \sigma_j < 0\}$. For $\forall v_i \in S, \forall v_j \in T$, find the shortest path from v_i to v_j;
4. Construct a complete bi-partite graph $H = (X, Y, E_H, W)$, with $X = \{x_{i,p} | v_i \in S, p = 1, 2, \cdots, \sigma_i\}$, $Y = \{y_{j,q} | v_j \in T, q = 1, 2, \cdots, |\sigma_j|\}$, $E_H = \{x_{i,p}y_{j,q} | x_{i,p} \in X, y_{j,q} \in Y\}$. Associates each edge $x_{i,p}y_{j,q}$ in H, $p = 1, 2, \cdots, \sigma_i$, $q = 1, 2, \cdots, |\sigma_j|$, a weight $w(v_i, v_j) \in W$, 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 $M = \{e_1, e_2, \cdots, e_k\}$ 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 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

3.3 A Heuristic Solution to Problem 2

The *color span*, $s(T)$, of an *Euler tour* T depends largely on the color of the feasible edges added to graph G and also the algorithm to find the *Euler tour*. Intuitively, a tour with less $s(T)$ can be found when the colors of the feasible edges become more. Based on such an intuition, we modify the process to find the feasible edges for G in a way such that the path formed by the feasible edges has a minimum *color span*.

Definition 1. *Given a graph $G = (V, E, C)$, the minimum color span path C_{ij} for node pair (v_i, v_j) is the one with the minimum color span value among all the paths from node v_i to v_j.*

Following we present an algorithm to find the minimum *color span* path from a given node s to every other node in a graph. The complexity of the algorithm is $O(n^2 m)$, where m, n is the number of edges and vertices of G.

Algorithm 3: Find the Minimum *color span* Paths.
Input: $G = (V, E, C), v$ /* a strongly connected colored digraph and a source node*/
Output: Minimum *color span* path from node v to every other node in G.

Notations: $l(v_i)$ - the current minimum *color span* value for all the paths from v to v_i;
$p(v_i)$ - the current minimum *color span* path from v to v_i;
$a(v_i)$ - the immediate ancestor for v_i on the current minimum *color span* path from v to v_i;
F - visited node set; M - unvisited node set.

begin
1. $l(v) := 0$, $p(v) := v$, $F := \{v\}$, $M := V \backslash \{v\}$;
2. For any $v_j \in V$ and $v_j \neq v$ /* initialization */
   $p(v_j) := NULL$;
   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, $p(v_i) := p(a(v_i)) \cdot v_i$;
4. Set $F := F \cup \{v_i\}$, $M := M \backslash \{v_i\}$. If $M = \emptyset$, stop, the minimum *color span* 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 $l(v_j) > s(p(v_i) \cdot (v_i, v_j))$
   /* $s()$ is defined in Section 3.1. */
   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 have not considered the *color span* of each augmented path. However, in problem 2, we only care about the *color span* of the final generated tour and the length of the tour is no longer an issue. An intuitive heuristic approach to problem 2 is to use path with smaller *color span* when augmenting. This is very similar to what is called the *minimax matching* problem [16].

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

The algorithm proposed in [10] can be modified to get the *minimax matching* of a bi-partite graph in time complexity $O(m^2)$, as shown in Algorithm 4.

Algorithm 4: Find the *Minimax Matching*.
Input: $H = (X, Y, E_H, W)$ /* 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 [10] to get the *maximin matching* M
   of H based on the new weight;
3. M is the *minimax matching* of the original graph H.

end

If we use the value of *color span* for each edge 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 *color span* is the minimal among all the matchings. By combining algorithm 1, 3 and 4, we now give a heuristic algorithm for problem 2 with complexity $O(m^2 + n^2 m)$, where m is the number of vertices and n is the number of edges of G.

Algorithm 5: A Heuristic Solution to Problem 2
Input: $G = (V, E, C)$ /* a strongly connected colored digraph */
Output: A *Postman Tour* with an approximately minimal *color span*.

begin
1. For each $v_i \in V$, set $\sigma_i := d^-(v_i) - d^+(v_i)$.
2. If $\sigma_i = 0, i = 1 \ldots n$, 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 minimum *color span* path from v_i to v_j using Algorithm 3;
4. Construct a complete bi-partite graph $H = (X, Y, E_H, W)$,with
   $X = \{x_{i,p} | v_i \in S, p = 1, 2, \cdots, \sigma_i\}$, $Y = \{y_{j,q} | v_j \in T, q = 1, 2, \cdots, |\sigma_j|\}$,
   $E_H = \{x_{i,p} y_{j,q} | x_{i,p} \in X, y_{j,q} \in Y\}$.
   Associates each edge $x_{i,p} y_{j,q}$ in H, $p = 1, 2, \cdots, \sigma_i$, $q = 1, 2, \cdots, |\sigma_j|$,
   a weight $w(v_i, v_j)$, where $w(v_i, v_j)$ is the *color span* of the
   minimum *color span* path from v_i to v_j in G.
5. Find the *minimax match* $M = \{e_1, e_2, \cdots, e_k\}$ in H using Algorithm 4.
6. For each edge $x_{i,p} y_{j,q} \in M$, suppose the minimum *color span* path between
   v_i and v_j in G is $C_{i,j}$, add every edge in $C_{i,j}$ 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

4 Test Generation For Internet Telephony End System

SIP-based Internet telephony systems have become popular with the introduction of 3GPP. Service creation, as well as feature interactions, has been well studied for the Internet telephony systems [1][12][13][14]. However we haven't seen any test generation work that considers the interaction detection problem

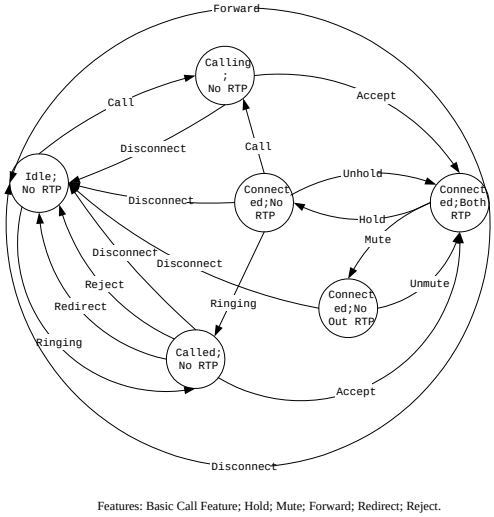


Fig. 2. Internet Telephony End System FSM

for SIP-based Internet telephony systems. In the following, we use an FSM to model an Internet telephony end system and apply the algorithms we discussed above to generate test sequences for the Internet telephony end system.

4.1 Model of the Internet Telephony End System

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

Fig. 2 gives the FSM of an Internet telephony end system. Since the end system is the only entity where signaling and media flows are guaranteed to converge, the state of the end system should include both control signalling state and media state. In the FSM, each state of the end system is denoted as $(Control\ State; Audio\ State)$, the combination of the signaling state and the audio media state. Basic call functions include the following actions: *call*, *ring*, *accept* and *disconnect*. The actions of the call hold feature include: *hold* and *unhold*; the actions of mute feature include: *mute* and *unmute*.

Fig. 3 gives the augmented graph of Fig. 2 after applying algorithm 2. There is a label $c : a$ on each transition, c is the color assigned to the transition, and a is the action that results in this transition. For this example FSM, since every shortest path used by algorithm 2 in finding the $perfect\ match$ is also the minimum *color span* path, the same augmented graph can be generated when applying algorithm 5.

Using algorithm 1, we generate the following *Euler tour* T for Fig. 3:
$Call \rightarrow Accept \rightarrow Hold \rightarrow Disconnect \rightarrow Call \rightarrow Accept \rightarrow Hold \rightarrow Call \rightarrow$

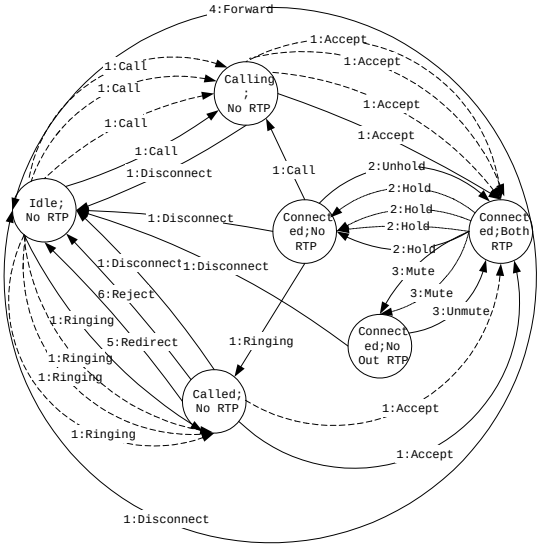


Fig. 3. Augmented Graph for Internet Telephony End System FSM

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

The corresponding color sequence for this test sequence is:
$CS(T) = 1121112111121611223115113341 11$, and its *color span* is $s(T) = 4$.

5 Test Generation For LMP

5.1 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. Link Management Protocol (LMP)[15] is one of the control plane components of GMPLS, and it provides the fundamental functions to support GMPLS routing and signaling protocols.

The features of LMP include: control channel management, link property correlation, link connectivity verification, and fault management. Control Channel Management allows two nodes in optical network to establish and maintain control channels between adjacent nodes. Link Property Correlation allows two nodes in optical network to automatically exchange their TE link properties, verify the TE link configuration. Link Connectivity Verification provides functions such that two nodes in optical network can discover their data plane neighbor,

exchange their interface ID, and verify their physical connectivity. Fault Management makes nodes in optical network suppress downstream alarms, localize faults for protection and restoration.

LMP features are specified using the Control Channel FSM, the Data Link FSM and the TE Link FSM in the LMP draft [15]. In most cases, Control Channel Management controls the state transition of a control channel, Link Property Correlation controls the state of a TE link, while behaviors of Link Connectivity Verification and Fault Management can change the state transition of a 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 a data link's state when it finds the data link property is not correlated in both sides, Control Channel Management can change a TE link's state when there is no active control channels for the TE link.

Some feature interaction problems of LMP have been identified in [2]. In the following, we study the feature interaction testing problem of LMP.We use the active data link FSM of LMP as an example, applying algorithm 2 to generate 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 checked.

5.2 Data Link Model of LMP

Fig. 4 shows the active 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.

Inputs:

1 : evCCUp: Control channel has gone up.
2 : evCCDown: LMP neighbor connectivity is lost.
3 : ?msgBeginVerifyAck: Receive BeginVerifyAck message.
4 : ?msgBeginVerifyOK: Receive correct BeginVerify message.
5: ?msgTstSuccess: Receive TestStatusSuccess message.
6 : ?msgTestOK: Receive compatible Test message.
7 : ?msgTstStatusFailure: Receive TestStatusFailure message.
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: Receive 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: Receive EndVerify message.
18: ?msgLinkSumNack: Receive LinkSummaryNack message.

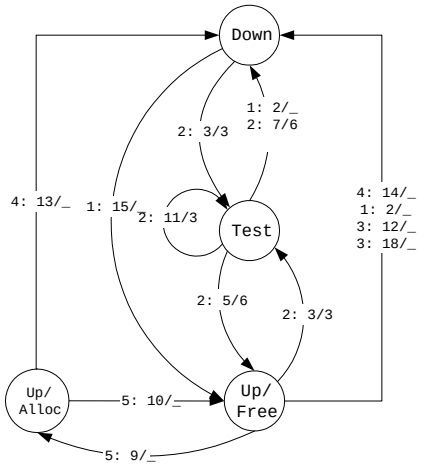


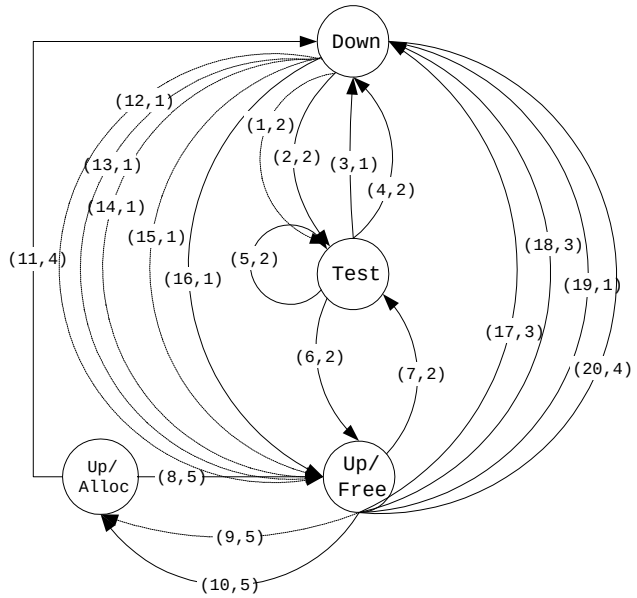
Fig. 4. Active LMP Data Link FSM

Output:
1 : !msgBeginVerify: Send out BeginVerify message.
2 : !msgBeginVerifyAck: Send out BeginVerifyAck message.
3 : !msgTest: Send out Test message.
4 : !msgTestSuccess: Send out TestSuccess message.
5 : !msgTestFailure: Send out TestFailure message.
6 : !msgTstStatusAck: Send out TstStatusAck message.
7: !msgEndVerify: Send out EndVerify message.
8: !msgEndVerifyAck: Send out EndVerifyAck message.
9: !msgBeginVerifyNack: Send out EndVerifyNack message.
10: !msgLinkSumNack: Send out LinkSumNack message.

5.3 Optimal Test Sequence for LMP

Fig. 5 shows the balanced augmentation of the active LMP data link FSM. Since the shortest paths adopted to augment Fig. 4 is also the minimum *color span* paths, the same augmented graph will be generated no matter which of algorithm 2 and algorithm 5 is used. In Fig. 5 the dashed links are the links augmented to Fig. 4. 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.

Using algorithm 1, we get an *Euler tour* T_1 for Fig. 5:
$T_1 = 1, 3, 2, 4, 12, 9, 11, 13, 10, 8, 17, 14, 18, 15, 20, 16, 7, 5, 6, 19$. Its corresponding color sequence is $CS(T_1) = 21221541553131412221$, and its *color span* is $s(T_1) = 3$. The following gives the test sequence corresponding to tour T_1.
?msgBeginVerifyAck/!msgTest → evCCDown/_ → ?msgBeginVerifyAck/!msgTest → ?msgTstStatusFailure/ !msgTstStatusAck → inBandConfigOK/_ → evLnkAlloc/_



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

Fig. 5. Symmetric Augmentation of the Active LMP Data Link FSM

→ evLocalizeFail/_ → inBandConfigOK/_ → evLnkAlloc/_ → evLnkDealloc/_ → ?msgLinkSumErr/ !msgLinkSumNack → inBandConfigOK/_ → ?msgLinkSumNack/_ → inBandConfigOK/_ → evDlDown/_ → inBandConfigOK/_ → ?msgBeginVerifyAck/!msgTest → evTestRet/!msgTest → ?msgTstSuccess/!msgTstStatusAck → evCCDown/_

6 Conclusion

In this paper, a technique is proposed to generate optimal conformance test sequences for the purpose of feature interaction detection for a complex feature-rich communication system. A feature-rich communication system may offer many features and these features can be implemented in multiple processes and as a result their operations can interleave with each other. Whether or not the implemented system can work correctly when such an interleaving occurs needs to be verified. 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 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 same approach can be used for many other purposes such as inter-operability testing

and fault detection. Some state verification techniques such as UIO sequences can also 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. 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.
4. David Lee, Mihalis Yannakakis, "Principles and Methods of Testing Finite State Machines - A Survey", Proceedings of the IEEE, Vol.84,No.8,August 1996.
5. T.S.Chow,"Testing software design modeled by finite-state machines ",IEEE Trans. Software Eng. Vol.SE-4,No.3,pp.178-187,1978.
6. K.K.Sabnani and A.T.Dahbura,"A protocol test generation procedure", Computer Networks and ISDN Syst.Vol.15,No.4,pp285-297,1988.
7. 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.
8. T. H. Cormen, C. E. Leiserson and R. L. Rivest, Introduction to Algorithms. The MIT Press, 1997.
9. J.A.Bondy and U.S.R.Murty, Graph Theory With Applications. New York: Elsevier North Holland,1976.
10. Gross O. The bottleneck assignment problem: an algorithm. In: Procedings, and Symposium on Mathematical Programming(Wolfe Ped), Rand Publication, 1960,87-88.
11. A.Gibbons, Algorithmic Graph Theory. Cambridge, MA:Cambridge University Press,1985.
12. John de Keijzer, Douglas Tait, and Rob Goedman, "JAIN: a new approach to services in communication networks". IEEE Communications Magazine, 38(1), January 2000.
13. Jonathan Lennox and Henning Schulzrinne, "Feature interaction in Internet telephony", In Feature Interaction in Telecommunications and Software Systems VI, Glasgow, United Kingdom, May 2000.
14. J. Rosenberg, J. Lennox, and Henning Schulzrinne. "Programming Internet telephony services". IEEE Network, 13(3):42C49, May/June 1999.
15. Jonathan P. Lang, "Link Management Protocol (LMP)", Internet draft, draft-ietf-ccamp-lmp-10.txt, October 2003, work in progress.
16. K. Imai, S. Sumino and H. Imai," Minimax Geometric Fitting of Two Corresponding Sets of Points and Dynamic Furthest Voronoi Diagrams", IEICE Transactions on Information and Systems, Vol.E81-D, No.11 (November 1998), pp.1162-1171.