Eliminating Redundant Tests in a Checking Sequence

Jessica Chen[1], Robert M. Hierons[2], Hasan Ural[3], and Husnu Yenigun[4]

[1] School of Computer Science, University of Windsor, Windsor, Ontario, N9B 3P4, Canada
[2] Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK
[3] School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada
[4] Faculty of Engineering and Natural Sciences, Sabanci University, Tuzla 34956, Istanbul, Turkey

Abstract. Under certain well–defined conditions, determining the correctness of a system under test (SUT) is based on a checking sequence generated from a finite state machine (FSM) specification of the SUT. When there is a distinguishing sequence for the FSM, an efficient checking sequence may be produced from the elements of a set $E_{\alpha'}$ of α'-sequences that verify subsets of states and the elements of a set E_C of subsequences that test the individual transitions. An optimization algorithm may be used in order to produce a shortest checking sequence by connecting the elements of $E_{\alpha'}$ and E_C using transitions drawn from an acyclic set. Previous work did not consider whether some transition tests may be omitted from E_C. This paper investigates the problem of eliminating subsequences from E_C for those transitions that correspond to the last transitions traversed when a distinguishing sequence is applied in an α'–sequence to obtain a further reduction in the length of a checking sequence.

1 Introduction

Finite state machines (FSMs) can be used to model many types of systems including communication protocols [1] and control circuits [2]. A number of specification languages such as SDL, Estelle, X-machines and Statecharts are based on extensions of FSMs. FSM based test techniques can often be applied to systems specified using such languages [3–8].

Given a formal model or specification of the required behaviour of the *system under test (SUT)* I it is normal to assume that I behaves like some unknown model that can be described using some particular formalism [9]. Given an FSM M, that models the required behaviour of SUT I, it is normal to assume that I behaves like some (unknown) FSM M_I with the same input and output alphabets as M. A common further assumption is that M_I has no more states than M.

Suppose M has n states. Let the set of deterministic FSMs with the same input and output alphabets as M and no more than n states be denoted $\Phi(M)$.

A finite set of input sequences is a *checking experiment* for M if, between them, they distinguish M from every element of $\Phi(M)$ which is not equivalent to M. Given FSM M, there is some checking experiment [10]. A *checking sequence* is an input sequence that forms a checking experiment.

The problem of generating a checking sequence for an FSM M is simplified if M has a distinguishing sequence: an input sequence $\bar{D}$ with the property that the output sequence produced by M in response to $\bar{D}$ is different for the different states of M. There has been much interest in the generation of short checking sequences from an FSM M when a distinguishing sequence is known [11–14]. Hierons and Ural [13] showed that an efficient checking sequence may be produced by combining the elements in some predefined set $E_{\alpha'}$ of α'–sequences that verify subsets of states and the elements of a set E_C of subsequences that test individual transitions using an acyclic set E'' of transitions from M. An optimization algorithm is then used in order to produce a shortest checking sequence by connecting the elements of $E_{\alpha'}$ and E_C using transitions drawn from E''. Their work did not consider whether some transition tests may be omitted from E_C.

In this paper, we show that the length of the checking sequences can be reduced even further, since not every element in E_C needs to be included in a checking sequence. Specifically, we eliminate subsequences from E_C for each transition that corresponds to the last transition traversed when a distinguishing sequence is applied in an α'–sequence to obtain further reductions in the length of a checking sequence. The reason we can eliminate the tests for such transitions is that, the existence of α'–sequences and the existence of the other transition tests in the checking sequence, already guarantee the correctness of the transitions for which the test segments are eliminated.

The shortest prefix of a distinguishing sequence $\bar{D}$ that distinguishes a state in M can actually be used as a special distinguishing sequence for that state [15]. Based on this observation, we use prefixes of distinguishing sequences as well, in order to further reduce the length of checking sequences.

The remaining of this paper is structured as follows. Section 2 introduces the basic concepts and notations used in this paper. Elements of an existing checking sequence construction method [13] which will be utilized in the proposed approach are also summarized here for completeness. Section 3 gives the details of the proposed approach to elimination of redundant transition tests in a checking sequence. A running example is used to illustrate the proposed approach and to compare the length of the resulting checking sequence to the one constructed by [13]. Conclusions are drawn in Section 4.

2 Preliminaries

2.1 Finite State Machines

A deterministic FSM M is defined by a tuple $(S, s_1, X, Y, \delta, \lambda)$ in which S is a finite set of *states*, $s_1 \in S$ is the *initial state*, X is the finite *input alphabet*, Y is

the finite *output alphabet*, δ is the *next state function* and λ is the *output function*. The functions δ and λ can be extended to input sequences in a straightforward manner. The number of states of M is denoted n and the states of M are enumerated, giving $S = \{s_1, \ldots, s_n\}$. An FSM is *completely specified* if the functions λ and δ are total. If an FSM M is not completely specified, it is possible to make M completely specified by either adding an error state or, for each $s_i \in S, a \in X$ such that $(s_i, a) \notin dom\ \delta$, adding the transition $(s_i, s_i, a/null)$ where *null* represents no output being produced.

A transition τ is defined by a tuple $(s_i, s_j, x/y)$ in which s_i is the *starting state*, x is the input, $s_j = \delta(s_i, x)$ is the *ending state*, and $y = \lambda(s_i, x)$ is the output.

Two states s_i and s_j of M are *equivalent* if, for every input sequence $\bar{x}$, $\lambda(s_i, \bar{x}) = \lambda(s_j, \bar{x})$. If $\lambda(s_i, \bar{x}) \neq \lambda(s_j, \bar{x})$ then $\bar{x}$ *distinguishes* between s_i and s_j. An FSM M is *minimal* if no FSM with fewer states than M is equivalent to M. A sufficient condition for M to be minimal is that every state can be reached from the initial state of M and no two states of M are equivalent. Throughout this paper $M = (S, s_1, X, Y, \delta, \lambda)$ will denote a deterministic, minimal, and completely specified FSM that describes the required behaviour of the SUT I.

An FSM, that will be denoted M_0 throughout this paper, is described in Figure 1. Here, $S = \{s_1, s_2, s_3, s_4, s_5\}$, $X = \{a, b\}$ and $Y = \{0, 1\}$.

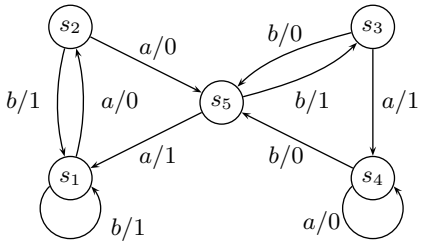

Fig. 1. The FSM M_0

Two FSMs M_1 and M_2 are *equivalent* if and only if for every state of M_1 there is an equivalent state of M_2 and vice versa. An input sequence distinguishes between two FSMs if its application leads to different output sequences for these FSMs. An input sequence $\bar{x}$ is a *checking sequence* for M if and only if $\bar{x}$ distinguishes between M and all elements of $\Phi(M)$ that are not equivalent to M.

Given FSM M, a *distinguishing sequence* is an input sequence $\bar{D}$ whose output distinguishes all the states of M. More formally, for all $s_i, s_j \in S$ if $s_i \neq s_j$ then $\lambda(s_i, \bar{D}) \neq \lambda(s_j, \bar{D})$. Thus, for example, M_0 has distinguishing sequence *aba*. We will use the notation $\bar{D}_i$ to denote the prefixes of distinguishing sequences that is sufficient to distinguish the states. Formally, given a distinguishing sequence $\bar{D}$ and a state s_i, $\bar{D}_i$ is the shortest prefix of $\bar{D}$ such that for any state s_j, if $s_i \neq s_j$ then $\lambda(s_i, \bar{D}_i) \neq \lambda(s_j, \bar{D}_i)$.

While not every minimal FSM has a distinguishing sequence and determining whether a minimal FSM has a distinguishing sequence is PSPACE-Complete [16], there has been interest in the problem of generating a checking sequence in the presence of a distinguishing sequence [12–14, 17]. This paper considers the problem of generating an efficient checking sequence from a deterministic, minimal, and completely specified FSM M with a known distinguishing sequence $\bar{D}$.

2.2 Directed Graphs

A *directed graph (digraph)* G is defined by a tuple (V, E) in which V is a set of vertices and E is a set of directed edges between the vertices. Each edge may have a label. An edge e from vertex v_i to vertex v_j with label l will be represented by (v_i, v_j, l). Edge e *leaves* v_i and *enters* v_j. For a vertex $v \in V$, $indegree_E(v)$ denotes the number of edges from E that enter v and $outdegree_E(v)$ denotes the number of edges from E that leave v.

Given an FSM, it is possible to produce a corresponding digraph in which each state is represented by a vertex and each transition is represented by an edge. Throughout this paper $G = (V, E)$ ($V = \{v_1, \ldots, v_n\}$) will be a digraph, that represents M, in which state s_i is represented by vertex v_i. A transition from state s_i to state s_j with input x and output y is represented by edge $e = (v_i, v_j, x/y)$ from E.

A sequence $\bar{P} = (n_1, n_2, x_1/y_1), \ldots, (n_{r-1}, n_r, x_{r-1}/y_{r-1})$ of pairwise adjacent edges from G forms a *walk* in which each *node* n_i represents a vertex from V and thus, ultimately, a state from S. Here $initial(\bar{P})$ denotes n_1, which is the *initial node* of $\bar{P}$, and $final(\bar{P})$ denotes n_r, which is the *final node* of $\bar{P}$. The sequence $\bar{T} = (x_1/y_1), \ldots, (x_{r-1}/y_{r-1})$ is the *label* of $\bar{P}$ and is denoted $label(\bar{P})$. In this case, $\bar{T}$ is said to *label* the walk $\bar{P}$. $\bar{T}$ is said to be a *transfer sequence* from n_1 to n_r. The walk $\bar{P}$ can be represented by the tuple $(n_1, n_r, \bar{T})$ or by the tuple $(n_1, n_r, \bar{x}/\bar{y})$ in which $\bar{x} = x_1, \ldots, x_{r-1}$ is the *input portion* of $\bar{T}$ and $\bar{y} = y_1, \ldots, y_{r-1}$ is the *output portion* of $\bar{T}$.

A *tour* is a walk whose initial and final nodes are the same. Given a tour $\bar{\Gamma} = e_1, \ldots, e_k$, $e_i = (n_i, n_{i+1}, l_i)$, $(1 \leq i \leq k)$ then $e_j, \ldots, e_k, e_1, \ldots, e_{j-1}$ is a walk formed by *starting* $\bar{\Gamma}$ with edge e_j, and hence by *ending* $\bar{\Gamma}$ with edge e_{j-1}. An *Euler Tour* is a tour that contains each edge exactly once. If the set of vertices represented by the nodes of walk $\bar{P}$ are distinct, $\bar{P}$ is said to be a *path*. A sequence of edges $e_1, \ldots, e_k$, $e_i = (n_i, n_{i+1}, l_i)$, $(1 \leq i \leq k)$ forms a *cycle* if $e_1, \ldots, e_{k-1}$ is a path and n_1 and n_{k+1} represent the same vertex. A set E' of edges from G is *acyclic* if no subset of E' forms a cycle.

A digraph is *strongly connected* if for any ordered pair of vertices (v_i, v_j) there is a walk from v_i to v_j. An FSM is *strongly connected* if the digraph that represents it is strongly connected. It will be assumed that any FSM considered in this paper is strongly connected.

2.3 Recognizing States and Verifying Edges

The algorithms of Ural et al. [14] and Hierons and Ural [13] use the notion of recognizing a node, corresponding to the state reached by a given input/output sequence, and verifying an edge of E. These notions, which are defined in terms of a given distinguishing sequence $\bar{D}$, are defined below. The key point is that, since the SUT I has no more states than M, if we observe the n possible responses of M to $\bar{D}$ when applied to I, then $\bar{D}$ must also be a distinguishing sequence for I. Once this has been demonstrated, we can use $\bar{D}$ to investigate the structure of I and thus to determine whether it is equivalent to M.

Consider a walk $\bar{P}$ and the nodes within it. Let $\bar{Q} = label(\bar{P})$.

Definition 1. *1. A node n_i of $\bar{P}$ is* d–recognized *in $\bar{Q}$ as state s of M if n_i is the initial node of a subpath of $\bar{P}$ whose label is input/output sequence $\bar{D}/\lambda(s, \bar{D})$.*

2. Suppose that $(n_q, n_i, \bar{T})$ and $(n_j, n_k, \bar{T})$ are subpaths of $\bar{P}$ and $\bar{D}/\lambda(s, \bar{D})$ is a prefix to $\bar{T}$ (and thus n_q and n_j are d–recognized in $\bar{Q}$ as state s of M). Suppose also that node n_k is d–recognized in $\bar{Q}$ as state s' of M. Then n_i is t–recognized *in $\bar{Q}$ as s'.*

3. Suppose that $(n_q, n_i, \bar{T})$ and $(n_j, n_k, \bar{T})$ are subpaths of $\bar{P}$ such that n_q and n_j are either d-recognized or t–recognized in $\bar{Q}$ as state s of M and n_k is either d–recognized or t–recognized in $\bar{Q}$ as state s' of M. Then n_i is t–recognized *in $\bar{Q}$ as s'.*

4. If node n_i of $\bar{P}$ is either d–recognized or t–recognized in $\bar{Q}$ as state s then n_i is recognized *in $\bar{Q}$ as state s.*

5. Edge $e = (v_a, v_b, x/y)$ is verified *in $\bar{Q}$ if there is a subpath $(n_i, n_{i+1}, x_i/y_i)$ of $\bar{P}$ such that n_i is recognized as s_a in $\bar{Q}$, n_{i+1} is recognized as s_b in $\bar{Q}$, $x_i = x$ and $y_i = y$.*

The first rule says that a node is d–recognized as a state s if it is followed by the input/output sequence $\bar{D}/\lambda(s, \bar{D})$. This is essentially saying that $\bar{D}$ defines a one–to–one correspondence between the states of the SUT and the states of M: this must be the case if the n different responses to $\bar{D}$ are observed in the SUT. The second and third rules say that if an input/output sequence is observed from two different nodes n and n' that are both recognized (d–recognized or t–recognized) as the same state then their final nodes should correspond to the same state of M.

The fifth rule is related to a transition test that is defined as follows: The *transition test* for a transition $\tau = (s_i, s_j, x/y)$ is $x/y\bar{D}/\lambda(s_j, \bar{D})\bar{T}_j$ for some transfer sequence $\bar{T}_j$. The following result, that provides a sufficient condition for an input sequence to be a checking sequence, may now be stated.

Theorem 1. *(Theorem 1, [14]) Let $\bar{P}$ be a walk from G representing M that starts at v_1 and $\bar{Q} = label(\bar{P})$. If every edge $(v_i, v_j, x/y)$ of G is verified in $\bar{Q}$, then the input portion of $\bar{Q}$ is a checking sequence of M.*

In this paper checking sequence generation is based on this result.

2.4 Defining α'–sequences

Suppose that an input/output sequence $\bar{x}/\bar{y}$, that labels a walk from the initial state of M, contains the n subsequences of the form $\bar{D}_i/\lambda(s_i, \bar{D}_i)$ $(1 \leq i \leq n)$. If $\bar{x}/\bar{y}$ labels a walk from the initial state of the SUT then, since the SUT has at most n states, $\bar{D}$ must be a distinguishing sequence for the SUT. Further, the response to $\bar{D}$ defines a bijection between the states of M and the states of the SUT. This observation lies at the heart of algorithms for generating a checking sequence on the basis of a distinguishing sequence and motivates the use of α'–sequences. We now adapt α'–sequences [13], so that they use prefixes of $\bar{D}$, and then explain their role in the construction of a checking sequence.

The first step is to choose a set $V_1, \ldots, V_q$, $q \geq 1$, of non-empty subsets of V whose union is V. We then order the elements of each V_k, $1 \leq k \leq q$, to give $V_k = \{v_1^k, \ldots, v_{m_k}^k\}$. Each v_i^k represents a state $s_{f(i,k)}$ of M (defining a function f). For each v_i^k, we produce a sequence $\bar{D}_{f(i,k)}/\lambda(s_{f(i,k)}, \bar{D}_{f(i,k)})\bar{T}_i^k$ that is the result of applying $\bar{D}_{f(i,k)}$ in state $s_{f(i,k)}$ followed by a (possibly empty) transfer sequence $\bar{T}_i^k = (\bar{x}_i^k/\bar{y}_i^k)$ whose final state corresponds to v_{i+1}^k, $1 \leq i \leq m_k$, ($v_{m_k+1}^k$ can be any v_w^j, $1 \leq j \leq q, 1 \leq w \leq m_j$). Each V_k thus defines a walk $\bar{P}_k$ whose starting state is $s_{f(1,k)}$ and whose label is the α'*–sequence* $\bar{\alpha}'_k = \bar{D}_{f(1,k)}/\lambda(s_{f(1,k)}, \bar{D}_{f(1,k)})\bar{T}_1^k \bar{D}_{f(2,k)}/\lambda(s_{f(2,k)}, \bar{D}_{f(2,k)})\bar{T}_2^k \ldots \bar{D}_{f(m_k,k)}/\lambda(s_{f(m_k,k)}, \bar{D}_{f(m_k,k)})\bar{T}_{m_k}^k \bar{D}_{f(w,j)}/\lambda(s_{f(w,j)}, \bar{D}_{f(w,j)})\bar{T}_w^j$ $(1 \leq j \leq q, 1 \leq w \leq m_j)$. The set $A = \{\bar{\alpha}'_1, \ldots, \bar{\alpha}'_q\}$ is called an α'–set. Given $\bar{\alpha}'_i \in A$, $\bar{\alpha}'_i$ is called an α'*–sequence from* A. Where A is clear, its members are simply called α'*–sequences*.

The α'–sequences play the following roles in checking sequence generation.

1. They verify that $\bar{D}$ is a distinguishing sequence for the SUT since they contain the n different $\bar{D}_i/\lambda(s_i, \bar{D}_i)$.
2. For each $s_i \in S$ they d–recognize the final state of the walk from s_i with label $\bar{D}_i/\lambda(s_i, \bar{D}_i)\bar{T}_i^k$. This is achieved by the subsequence $\bar{D}_i/\lambda(s_i, \bar{D}_i)\bar{T}_i^k$ followed by the sequence $\bar{D}_j/\lambda(s_j, \bar{D}_j)$ for some $1 \leq j \leq n$. Thus, if the subsequence $\bar{D}_i/\lambda(s_i, \bar{D}_i)\bar{T}_i^k$ is seen elsewhere in the label of a walk, then the final node of this is t–recognized as the state s_j reached from s_i by a walk with label $\bar{D}_i/\lambda(s_i, \bar{D}_i)\bar{T}_i^k$.
3. An α'–sequence with starting state s_i starts with $\bar{D}_i/\lambda(s_i, \bar{D}_i)$ and thus its initial node is recognized.
4. If the label of a walk $\bar{P}$ contains every α'–sequence from α'–set A then the final node of each $\bar{P}_k$ is t–recognized.

2.5 Checking Sequence Construction: An Existing Approach

The proposed reduction of the length of a checking sequence is an enhancement of the checking sequence generation approach given in [13] where first a digraph $G' = (V', E')$ is obtained by augmenting the given digraph $G = (V, E)$, representing an FSM as follows:

Let the labels $\bar{\alpha}'_1, \ldots, \bar{\alpha}'_q$ of walks $\bar{P}_1, \ldots, \bar{P}_q$ form an α'–set A. Then, from the elements of A, a set of transfer sequences, called T–set, is formed as a set of labels of subpaths $\bar{R}_1, \ldots, \bar{R}_p$ of walks $\bar{P}_1, \ldots, \bar{P}_q$, such that each element $\bar{T}_i$ of T–set is $label(\bar{R}_i)$ where $\{\bar{R}_i : i = 1, 2, \ldots, p\} = \{(v_j^k, \delta(v_j^k, \bar{D}_{f(j,k)}\bar{x}_j^k), \bar{D}_{f(j,k)}/\lambda(v_j^k, \bar{D}_{f(j,k)})\bar{T}_j^k : 1 \leq k \leq q \text{ and } 1 \leq j \leq m_k\}$. Thus, the starting state of $\bar{R}_i$ is recognized in some $\bar{\alpha}'_k$ because $\bar{D}_{f(j,k)}$ is applied to the starting state of $\bar{R}_i$ and the ending state of $\bar{R}_i$ is recognized in some $\bar{\alpha}'_k$ because the ending state of $\bar{R}_i$ is $\delta(v_j^k, \bar{D}_{f(j,k)}\bar{x}_j^k)$ to which $\bar{D}_{f(j+1,k)}$ is applied. The set of walks $\bar{P}_1, \ldots, \bar{P}_q$ and the set of subpaths $\bar{R}_1, \ldots, \bar{R}_p$ are included in G' as edges in $E_{\alpha'} \subset E'$ and in $E_T \subset E'$, respectively, in order to facilitate the recognition of vertices in the label $\bar{Q}$ of the solution $\bar{P}$. Moreover, a transition test for each edge of G is induced in G' as edges in $E_C \subset E'$ in order to verify every transition of M in $label(\bar{P}) = \bar{Q}$. Furthermore, a set of edges from E are included in G' as edges in $E'' \subset E'$ to increase the connectivity of the vertices in G'.

Formally, $G' = (V', E')$ is obtained from $G = (V, E)$ as follows:
$V' = V \cup U'$ where $U' = \{v'_i : \text{ for every } v_i \in V\}$ and $E' = E_{\alpha'} \cup E_C \cup E_T \cup E''$,
$E_{\alpha'} = \{((\text{starting state of } \bar{P}_k), (\text{ending state of } \bar{P}_k)', \bar{\alpha}'_k) : 1 \leq k \leq q\}$,
$E_C = \{(v'_i, v_j, x/y) : (v_i, v_j, x/y) \in E\}$,
$E_T = \{((\text{starting state of } \bar{R}_i), (\text{ending state of } \bar{R}_i)', \bar{T}_i) : 1 \leq i \leq p\}$,
E'' is a subset of $\{(v'_i, v'_j, x/y) : (v_i, v_j, x/y) \in E\}$ such that $G'' = (U', E'')$ has no tour and G' is strongly connected.

Once G' is formed, a minimal symmetric augmentation G^* of the graph induced by the edges in $E_{\alpha'} \cup E_C$, that may be produced by adding edges from E', is found. If G^*, with its isolated vertices removed, is connected, G^* has an Euler tour. Otherwise, a heuristic is applied to make G^* connected and an Euler tour is formed. This Euler tour $\bar{\Gamma}$ of G^* contains all edges in $E_{\alpha'} \cup E_C$. Let τ be a transition with ending state s_1 which is represented by an edge $e = (v'_i, v_1, x/y) \in E_C$ in $\bar{\Gamma}$. Let $\bar{P}$ be a walk of G' that is formed by ending $\bar{\Gamma}$ with edge e. Then, the input portion of $\bar{Q} = label(\bar{P})\bar{D}_1/\lambda(s_1, \bar{D}_1)$ is a checking sequence of M that starts at v_1, in accordance with Theorem 1.

3 Producing Checking Sequences

This section explains how, given an α'–set A, we can produce a checking sequence without considering some of the edges in E_C which represent test segments for a subset L of E. Thus, L is a set of edges which stand for transitions whose transition tests are redundant and can be eliminated. In this paper, we use prefixes of distinguishing sequences wherever it applies, and we use empty transfer sequences in the formation of α'-sequences. In the following, we first define L, and then explain the algorithm to generate the checking sequence.

3.1 Transition Test Exemption

Similar to showing an edge being verified as given in Definition 1, in order to show a sequence of edges being verified we first introduce the notion of a sequence of edges being traced.

Definition 2. *Let $\bar{\rho} = e_1 e_2 \dots e_h$ be a sequence of edges in G, where $e_m = (v_{i_m}, v_{i_{m+1}}, x_m/y_m)$ for $1 \leq m \leq h$. $\bar{\rho}$ is* traced in $\bar{Q}$= *label$(\bar{P})$ of a walk $\bar{P}$ in G if there exists a subpath $(n_1, n_{h+1}, x'_1 x'_2 \dots x'_h / y'_1 y'_2 \dots y'_h)$ in $\bar{P}$ such that n_1 is recognized as v_{i_1}, n_{h+1} is recognized as $v_{i_{h+1}}$, and $x_m/y_m = x'_m/y'_m$ for $1 \leq m \leq h$.*

Lemma 1. *Let $\bar{\rho} = e_1 e_2 \dots e_h$ ($h \geq 1$) be a sequence of edges in G, where $e_m = (v_{i_m}, v_{i_{m+1}}, x_m/y_m)$ for $1 \leq m \leq h$, and $\bar{Q} = label(\bar{P})$ be the label of a walk $\bar{P}$ in G. Assume that $\bar{\rho}$ is traced in $\bar{Q}$. Further assume that $e_1, e_2, \dots, e_{h-1}$ (all the edges in $\bar{\rho}$ except the last one) are all verified in $\bar{Q}$. Then, e_h is also verified in $\bar{Q}$.*

Proof. The proof is by induction on h. When $h = 1$ the lemma holds trivially, since $\bar{\rho}$ has only one edge, hence Definition 1 applies.

For the induction step, since e_1 is verified in $\bar{Q}$, there must exist a subpath $\bar{P}_1 = (n_j, n_k, x'_1/y'_1)$ in $\bar{P}$ where n_j is recognized as v_{i_1}, n_k is recognized as v_{i_2}, and $x'_1/y'_1 = x_1/y_1$. Since $\bar{\rho}$ is traced $\bar{Q}$, there must exist a subpath $\bar{P}_2 = (n_q, n_s, x''_1 x''_2 \dots x''_h / y''_1 y''_2 \dots y''_h)$ in $\bar{P}$ where n_q is recognized as v_{i_1}, n_s is recognized as $v_{i_{h+1}}$, and $x''_m/y''_m = x_m/y_m$ for $1 \leq m \leq h$. Let us divide the path $\bar{P}_2$ into two as $\bar{P}_{21} = (n_q, n_i, x_1/y_1)$ and $\bar{P}_{22} = (n_i, n_s, x_2 x_3 \dots x_h / y_2 y_3 \dots y_h)$. According to Definition 1, the paths $\bar{P}_1$ and $\bar{P}_{21}$ recognize n_i as v_{i_2}. Then, the existence of $\bar{P}_{22}$ in $\bar{P}$ implies that $\bar{\rho}' = e_2 e_3 \dots e_h$ is traced in $\bar{Q}$. This concludes the proof since the length of $\bar{\rho}'$ is $h-1$, $\bar{\rho}'$ is traced in $\bar{Q}$ and e_m (for $2 \leq m < h$) is verified in $\bar{Q}$. □

Lemma 1 suggests that if there is a sequence of edges which is traced in the label $\bar{Q}$ of a path, then $\bar{Q}$ already includes what it takes to verify the last edge in the sequence, provided that all the other edges in the sequence are verified in $\bar{Q}$. In fact, inclusion of α'–sequences in the checking sequences guarantee that there are some sequences of edges which are traced.

For each $s_i \in S$, there exists an α'-sequence in the α'-set that can d-recognize the final state of the walk from s_i with label $\bar{D}_i/\lambda(s_i, \bar{D}_i)$, as the subsequence $\bar{D}_i/\lambda(s_i, \bar{D}_i)$ is followed by $\bar{D}_j$ for some $1 \leq j \leq n$. This is due to the fact that we use empty transfer sequences between the applications of $\bar{D}_i$ and $\bar{D}_j$ in α'–sequences. Formally, we have the following result.

Lemma 2. *Let A be an α'–set, and $\bar{Q} = label(\bar{P})$ be the label of a walk $\bar{P}$ in G. If $\bar{Q}$ includes all the α'–sequences in A, then for all $s_i \in S$, $\bar{\rho}_i = e_{j_1} e_{j_2} \dots e_{j_{|\bar{D}_i|}}$, where $label(\bar{\rho}_i) = D_i/\lambda(s_i, D_i)$, $\bar{\rho}_i$ is traced in $\bar{Q}$.*

Proof. Note that $\bar{\rho}_i$ corresponds to the application of $\bar{D}_i$ at state s_i. Consider the occurrence of $\bar{D}_i/\lambda(s_i, \bar{D}_i)$ in $\bar{Q}$ which is immediately followed by an occurrence of $\bar{D}_j/\lambda(s_j, \bar{D}_j)$, for some $1 \leq j \leq n$, which is guaranteed since all α'–sequences are included in $\bar{Q}$. $initial(\bar{\rho}_i)$ will be recognized as state s_i. Since there exists an application of some $\bar{D}_j$ after $\bar{D}_i$, $final(\bar{\rho}_i)$ will be recognized as state $s_j = \delta(s_i, \bar{D}_i)$. □

Lemma 3. *Let A be an α'–set, and $\bar{Q} = label(\bar{P})$ be the label of a walk $\bar{P}$ in G such that $\bar{Q}$ includes all the α'–sequences in A. Let $\bar{\rho}_i = e_{j_1} e_{j_2} \dots e_{j_{|D_i|}}$, be a sequence of edges where $label(\bar{\rho}_i) = D_i/\lambda(s_i, D_i)$ for some state $s_i \in S$. If all the edges $e_{j_1} e_{j_2} \dots e_{j_{|D_i|-1}}$ are verified in $\bar{Q}$, then $e_{j_{|D_i|}}$ is also verified in $\bar{Q}$.*

Proof. The result follows from Lemma 1 and Lemma 2. □

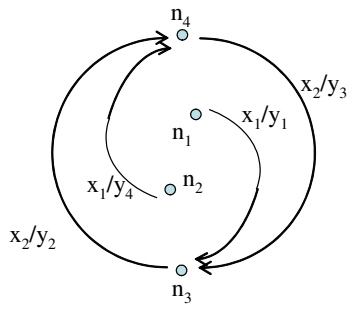

Fig. 2. An example to illustrate the edges that can be excluded from the transition tests

Suppose Figure 2 shows a subgraph of G where $\bar{D}_1 = \bar{D}_2 = x_1x_2x_2$. Suppose also that $\bar{Q}$ contains α'-sequences with

- x_1/y_1 x_2/y_2 x_2/y_3 $\bar{D}_3/\lambda(s_3, \bar{D}_3)$ and
- x_1/y_4 x_2/y_3 x_2/y_2 $\bar{D}_4/\lambda(s_4, \bar{D}_4)$

as subsequences. Then using the above lemma, we know that

- if edges $(n_1, n_3, x_1/y_1)$, $(n_3, n_4, x_2/y_2)$ are verified in $\bar{Q}$, then $(n_4, n_3, x_2/y_3)$ is verified in $\bar{Q}$;
- if edges $(n_2, n_4, x_1/y_4)$, $(n_4, n_3, x_2/y_3)$ are verified in $\bar{Q}$, then $(n_3, n_4, x_2/y_2)$ is verified in $\bar{Q}$.

Of course, we cannot draw conclusion in this case that if edges $(n_1, n_3, x_1/y_1)$ and $(n_2, n_4, x_1/y_4)$ are verified in $\bar{Q}$, then $(n_3, n_4, x_2/y_2)$ and $(n_4, n_3, x_2/y_3)$ are verified in $\bar{Q}$.

The following procedure shows a possible way to calculate a set of edges that can be excluded from the transition tests.

Let $P_S = \{\bar{\rho}_i \mid \forall s_i \in S, \bar{\rho}_i$ is a sequence of edges such that $label(\bar{\rho}_i) = \bar{D}_i/\lambda(s_i, \bar{D}_i)\}$. For any $\bar{\rho} \in P_S$, the last edge of $\bar{\rho}$, denoted as $last(\bar{\rho})$, is verified in $\bar{Q}$ provided that all other edges in $\bar{\rho}$ are verified in $\bar{Q}$ as proposed by Lemma 3.

Let $L_0 = \{e \mid e = last(\bar{\rho}),\ \bar{\rho} \in P_S\}$. Obviously, we want to have as many edges in L_0 as possible to be excluded from being considered for transition test.

First note that, for an α'–sequence $\bar{\alpha}'_k$ with starting state s_i, if we do not test any of the incoming transitions of s_i, then $\bar{\alpha}'_k$ will not be included in the generated checking sequence, since each α'–sequence is used to test the end state of a transition. Similarly, since we want to generate a checking sequence that starts from s_1, the initial state of M, at least one incoming transition of s_1 must be tested, so that the tour generated passes over v_1. Therefore let L_1 be a maximal subset of L_0 such that, $indegree_{L_1}(v_i) < indegree_E(v_i)$ for each v_i corresponding to a state s_i which is s_1 or a starting state of an α'–sequence.

Further note that according to Lemma 1, the test for a transition can be exempted only if some other transitions are tested. Therefore, we need to make sure that there is no cyclic dependency between the transitions that are exempted from transition tests. The following algorithm can be used for this purpose:

Construct a digraph $G_S = (V_S, E_S)$ where V_S contains one node for each $e \in L_1$. $(v_1, v_2) \in E_S$ if and only if $v_1 \neq v_2$, and for some $\bar{\rho} \in P_S$, v_2 corresponds to $e_2 = last(\bar{\rho})$, and v_1 corresponds to some e which appears in $\bar{\rho}$. Now, if G_S is cyclic, remove the minimal number of nodes from G_S so that it becomes acyclic. For each removed node, also remove its corresponding edge in L_1. This is an instance of Feedback Vertex Set problem [18], which is NP–complete. However certain heuristic approaches exist for this problem [19, 20].

The remaining edges in L_1 then represent transitions for which we do not need a transition test. We use L below to denote this set of edges. Since there can be at most n edges in L_0, there can be at most n transition tests that can be removed from the checking sequence.

3.2 Checking Sequence Construction

Now using L, we can improve on the algorithm in [13] for the checking sequence generation, by reducing the set of edges that must be included in the checking sequence. This is summarized below. Recall that we use prefixes of a distinguishing sequence and empty transfer sequences. The digraph $G' = (V', E')$ is obtained from $G = (V, E)$ as follows

- $V' = V \cup U'$ where $U' = \{v'_i$: for every $v_i \in V\}$, and $E' = E_{\alpha'} \cup E_C \cup E_T \cup E''$,
- $E_{\alpha'} = \{((\text{starting state of } \bar{P}_k), (\text{ending state of } \bar{P}_k)', \bar{\alpha}'_k) : 1 \leq k \leq q\}$,
- $E_C = \{(v'_i, v_j, x/y) : (v_i, v_j, x/y) \in E\}$,
- $E_T = \{(v_i, (\delta(v_i, \bar{D}_i))', \bar{D}_i/\lambda(v_i, \bar{D}_i))$: for every $v_i \in V$ s.t. there exists an edge in E_C ending at $v_i\}$,
- E'' is a subset of $\{(v'_i, v'_j, x/y) : (v_i, v_j, x/y) \in E\}$ such that $G'' = (U', E'')$ has no tour and, excluding isolated nodes in G', G' is strongly connected.

E'' can be constructed similarly as discussed in [14]. However, we obtain an additional issue in the proposed algorithm; since we are using empty transfer sequences in the α'-sequences, such a set E'' might not exist. Where this is the case it is necessary to use non-empty transfer sequences, along the lines of [13,14].

It is straightforward both to extend the approach given in this paper to the case where non-empty transfer sequences are used and to show how a set of transfer sequences can be chosen in order to ensure the existence of E''.

Theorem 2. *Let E'_C be defined as $E'_C = \{(v'_i, v_j, x/y) : (v_i, v_j, x/y) \in E - L\}$. Let $\bar{\Gamma}$ be a tour of G' that contains all edges in $E_{\alpha'} \cup E'_C$ which is found in the same manner as in [13]. Let τ be a transition with ending state s_1 which is represented by an edge $e = (v'_i, v_1, x/y) \in E'_C$ in $\bar{\Gamma}$. Let $\bar{P}$ be a walk of G' that is formed by ending $\bar{\Gamma}$ with edge e, and $\bar{Q} = label(\bar{P})\bar{D}_1/\lambda(s_1, \bar{D}_1)$. Then the input portion of $\bar{Q}$ is a checking sequence of M.*

Proof. All edges in $E - L$ are verified in $\bar{Q} = label(\bar{P})\bar{D}_1/\lambda(s_1, \bar{D}_1)$. According to Lemma 3 and the way L is constructed, if all edges in $E - L$ are verified in $\bar{Q}$, then all edges in L are verified in $\bar{Q}$. Thus, all edges of G are verified in $\bar{Q}$, and by Theorem 1, the input portion of $\bar{Q}$ is a checking sequence of M. □

3.3 Application

Let us consider FSM M_0 given in Figure 1. A distinguishing sequence for M_0 is $\bar{D} = aba$. The shortest prefixes of $\bar{D}$ that are sufficient to distinguish each state are: $\bar{D}_1 = aba$, $\bar{D}_2 = aba$, $\bar{D}_3 = ab$, $\bar{D}_4 = ab$, and $\bar{D}_5 = ab$. Using these $\bar{D}_i$'s, the α'–set for M_0 is $\{\bar{\alpha}'_1\}$, where $\bar{\alpha}'_1$, the label of $\bar{P}_1 = (v_3, v_1, \bar{\alpha}'_1)$, is

$$\bar{D}_3\bar{D}_5\bar{D}_1\bar{D}_2\bar{D}_4\bar{D}_5/\lambda(v_3, \bar{D}_3\bar{D}_5\bar{D}_1\bar{D}_2\bar{D}_4\bar{D}_5)$$

The set L_0 consists of the edges corresponding to the last transition of $\bar{D}_i$ when applied at s_i, $1 \leq i \leq 5$, hence $L_0 = \{(v_1, v_2, a/0), (v_3, v_4, a/1), (v_4, v_5, b/0), (v_1, v_1, b/1)\}$. The starting state of the only α'–sequence is s_3 and we have at least one incoming edge of v_3 (e.g. $(v_5, v_3, b/1)$) which is not included in L_0. Similarly, we have at least one incoming edge of v_1 (e.g. $(v_2, v_1, b/1)$) which is not in L_0. Therefore $L_1 = L_0$. The graph G_S for L_1 can shown to be acyclic, hence we also have $L = L_1$.

$E_T = \{\bar{T}_1, \bar{T}_2, \bar{T}_3, \bar{T}_4, \bar{T}_5\}$ where $\bar{T}_i = \bar{D}_i/\lambda(s_i, \bar{D}_i)$. The graph $G' = (V', E')$ is given in Figure 3.

A tour $\bar{\Gamma}$ over G' that contains all the edges in $E_{\alpha'} \cup E'_C$ is

$$(v_1, v'_2, \bar{T}_1), (v'_2, v'_5, a/0), (v'_5, v'_3, b/1), (v'_3, v'_4, a/1), (v'_4, v_4, a/0), (v_4, v'_5, \bar{T}_4),$$
$$(v'_5, v'_3, b/1), (v'_3, v_5, b/0), (v_5, v'_1, \bar{T}_5), (v'_1, v'_2, a/0), (v'_2, v_1, b/1), (v_1, v'_2, \bar{T}_1),$$
$$(v'_2, v_5, a/0), (v_5, v'_1, \bar{T}_5), (v'_1, v'_2, a/0), (v'_2, v'_5, a/0), (v'_5, v_3, b/1), (v_3, v'_1, \bar{\alpha}'_1),$$
$$(v'_1, v'_2, a/0), (v'_2, v'_5, a/0), (v'_5, v_1, a/1)$$

Note that $\bar{\Gamma}$ already starts at v_1. Hence when we consider the the walk $\bar{P}$ corresponding to $\bar{\Gamma}$ given above, the input portion of $\bar{Q} = label(\bar{P})\bar{D}_1/\lambda(s_1, \bar{D}_1)$ forms a checking sequence of length 44, which is a shorter checking sequence than the one given in [13], which was reported as 64.

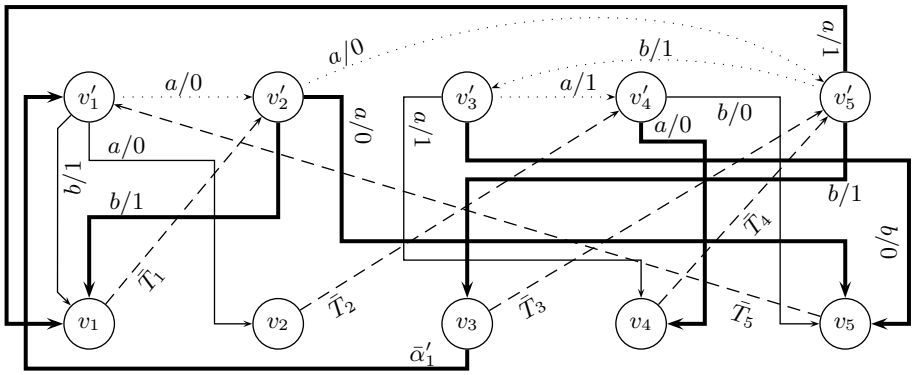

Fig. 3. $G' = (V', E')$ for M_0. The nodes in V and U' are at the bottom, and at the top respectively. The dashed lines are the edges in E_T, and the dotted lines are the edges in E''. The edges in $E_{\alpha'} \cup E_C$ are given in solid lines. The bold solid lines are the edges in $E_{\alpha'} \cup E'_C$, and the remaining solid lines are the edges in L.

4 Conclusion

We have shown that, when α'–sequences are used in constructing a checking sequence, some transitions tests can be identified as redundant. Such tests are then eliminated by the optimization algorithm used to construct a shorter checking sequence, and hence a further reduction is obtained in the length of a resulting checking sequence.

The approach proposed in this paper starts with a given set of α'–sequences where empty transfer sequences are used after the application of each distinguishing sequence or its prefix at a state. We believe that selecting α'–sequences judiciously will result in further reductions in the length of a checking sequence. A recent study by Hierons and Ural [21] show how α'–sequences can be chosen so that their use minimizes the sum of the lengths of the subsequences to be combined in checking sequence generation. The related checking sequence generation algorithm then produces the set of connecting transitions *during* the optimization phase. Our proposed approach can also be incorporated to the method given in [21].

Acknowledgment

This work was supported in part by "Natural Sciences and Engineering Research Council of Canada under grant RGPIN 976", "Leverhulme Trust grant number F/00275/D, Testing State Based Systems", and "Engineering and Physical Sciences Research Council grant number GR/R43150, Formal Methods and Testing (FORTEST)".

References

1. Tanenbaum, A.S.: Computer Networks. 3rd edition edn. Prentice Hall International Editions, Prentice Hall (1996)
2. Pomeranz, I., Reddy, S.M.: Test generation for multiple state–table faults in finite–state machines. IEEE Transactions on Computers **46** (1997) 783–794
3. Hierons, R.M., Harman, M.: Testing conformance to a quasi–non–determinstic stream X–machine. Formal Aspects of Computing **12** (2000) 423–442
4. Holcombe, M., Ipate, F.: Correct Systems: Building a Business Process Solution. Springer–Verlag (1998)
5. Luo, G., Das, A., v. Bochmann, G.: Generating tests for control portion of SDL specifications. In: Proceedings of Protocol Test Systems VI, Elsevier (North-Holland) (1994) 51–66
6. Tan, Q.M., Petrenko, A., v. Bochmann, G.: Modeling basic lotos by fsms for conformance testing. In: IFIP Protocol Specification, Testing, and Verification XV. (1995) 137–152
7. Ural, H., Saleh, K., Williams, A.: Test generation based on control and data dependencies within system specifications in SDL. Computer Communications **23** (2000) 609–627
8. v. Bochmann, G., Petrenko, A., Bellal, O., Maguiraga, S.: Automating the process of test derivation from SDL specifications. In: SDL Forum'97, Paris, France (1997)
9. International Telecommunications Union Geneva, Switzerland: Recommendation Z.500 Framework on formal methods in conformance testing. (1997)
10. Moore, E.P.: Gedanken-experiments. In Shannon, C., McCarthy, J., eds.: Automata Studies. Princeton University Press (1956)
11. Gonenc, G.: A method for the design of fault detection experiments. IEEE Transactions on Computers **19** (1970) 551–558
12. Hennie, F.C.: Fault–detecting experiments for sequential circuits. In: Proceedings of Fifth Annual Symposium on Switching Circuit Theory and Logical Design, Princeton, New Jersey (1964) 95–110
13. Hierons, R.M., Ural, H.: Reduced length checking sequences. IEEE Transactions on Computers **51** (2002) 1111–1117
14. Ural, H., Wu, X., Zhang, F.: On minimizing the lengths of checking sequences. IEEE Transactions on Computers **46** (1997) 93–99
15. Lee, D., Yannakakis, M.: Principles and methods of testing finite–state machines – a survey. Proceedings of the IEEE **84** (1996) 1089–1123
16. Lee, D., Yannakakis, M.: Testing finite state machines: state identification and verification. IEEE Trans. Computers **43** (1994) 306–320
17. Kohavi, I., Kohavi, Z.: Variable-length distinguishing sequences and their application to the design of fault–detection experiments. IEEE Transactions on Computers (1968) 792–795
18. Garey, M.R., Johnson, D.S.: Computers and Intractability. W. H. Freeman and Company, New York (1979)
19. Bar-Yehuda, R., Geiger, D., Naor, J., Roth, R.: Approximation algorithms for the vertex feedback set problem with applications to constraint satisfaction and bayesian inference. In: Proceedings of Fifth ACM-SIAM Symposium on Discrete Algorithms. (1994) 344–354
20. Fujito, T.: A note on approximation of the vertex cover and feedback vertex set problems. Information Processing Letters **59** (1996) 59–63
21. Hierons, R.M., Ural, H.: Optimizing the length of checking sequences. (submitted)