MINIMAL LENGTH CHECKING SEQUENCE GENERATION FOR TESTING COMMUNICATION PROTOCOLS

by

Zhuo Han

A M. Sc. Thesis

submitted to the School of Graduate Studies and Research

of the University of Ottawa

in partial fulfillment of the requirements for the degree of

Master of Computer Science*

Department of Computer Science

University of Ottawa

Ottawa, Ontario

* The Master of Computer Science program is a joint program with Carleton University, administrated by the Ottawa-Carleton Institute for Computer Science.

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ABSTRACT

This study discusses the generation of the minimum length checking sequences for FSM-based protocol conformance testing. The discussion focuses on finding the minimal length of resulting checking sequences for an FSM under different conditions. Without interleaving state identification and transition verification sequences, four methods (D-method, W-method, Wp-method and UIOv-method) of generating minimum length test sequences for FSM_s with reliable reset feature are reviewed and proved to construct checking sequences. These four methods are then improved to generate minimum length checking sequences for FSM_s without reliable reset feature and proved to construct checking sequences. Moreover, the effects of interleaving the state identification and transition verification sequences on the length of the checking sequences are studied. An algorithm for interleaving the state identification and transition verification sequences generated by the above four methods with reliable reset feature is proposed. It is observed that the reduction in the length of checking sequences due to interleaving is significant. Finally, the two general models for constructing minimal length checking sequences using distinguishing sequences with interleaved state identification and transition verification sequences are proposed. The proof for the first model to find a minimum length checking sequence in polynomial time is provided. The sequences generated by using both models are proved to be checking sequences. Examples are provided as applications of all the proposed methods and models.

ACKNOWLEDGEMENTS

I would like to express my gratefulness to my supervisor Professor Hasan Ural. His advice in guiding me to complete my studies and to the right track of many things is most significant and invaluable. His endeavour and suggestions in reviewing this thesis are most essential for me to finish the writing of the thesis.

I would also like to acknowledge the support given by my husband, Donghui Xie, without his support I would not be able to finish this study.

I also wish to thank Telecommunication Research Institute of Ontario (TRIO) & Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support.

TABLE OF CONTENTS

LIST OF FIGURES

PAGE

LIST OF TABLES

PAGE

1. INTRODUCTION

1.1 Background

A communication system uses a precise set of rules, called a *protocol*, to define the orderly exchange of interactions among the elements of the system. A statement of a collection of such rules is called a *specification* for the protocol. In order to eliminate ambiguities and incompleteness in the protocol specification, protocols are typically modeled as extended finite-state machines [BOC82] where the control portion is a finite-state machine (FSM) and the data portion consists of program segments. The discussions on the data portion of protocols can be found in [URA91b][URA87][SAR87]. In this thesis, we will focus on the control portion of a communication protocol (henceforth referred to as protocol or *FSM-based protocol* for simplicity). The state of an FSM is defined as a stable condition in which the FSM rests until a stimulus, called an *input*, is applied. The FSM generates a response, called an *output*, (which may be null) when an input is applied, and moves into the next state (which may be the same as the previous state) where it stays until the next input is applied. An FSM is defined as two functions, namely next state function, S X I -> S and output function, S X I -> O where S, I, O are finite non-empty sets of states, inputs, and outputs, respectively.

To ensure the successful exchange of information between the communicating entities, every protocol implementation must be tested for conformance to the specification of the protocol. This procedure is called *protocol conformance testing*[RAY87]. It is usually carried out by applying, by means of an upper and a lower tester, a selected sequence of inputs to an implementation under test and verifying that the corresponding sequence of outputs is as expected according to the specification of the protocol [AHO88][SID89].

Practical limitations make the protocol conformance testing impossible to be exhaustive, and economic considerations may restrict testing still further [RAY87]. Therefore, only a partial behavior of an implementation will be checked for conformance to the protocol specification. It is assumed that by checking a limited set of behaviors of a protocol implementation, its conformance to the specification can be assured to an acceptable level [SID90]. In general, the transition level behavior is chosen as the partial behavior to be checked for the implementation of an FSM [KOH78].

Foundations of testing the implementation of a system modeled as an FSM can be found in sequential circuit testing literature [GON70][HEN64][KOH78], where determining, under certain assumptions, whether any given "black box" implementation of an FSM is functioning correctly is referred to as a *fault detection experiment.* This experiment consists of applying an input sequence, observing the actual output sequence produced in response to the application of the input sequence, and comparing the actual output sequence to the expected output sequence. The applied input sequence and the expected output sequence form a *checking sequence.*

The construction of a checking sequence must deal with the "black box" nature of a given implementation M' of an FSM M which allows only limited controllability and observability of M'. The limited controllability refers to not being able to directly transfer M' to a designated state and the limited observability refers to not being able to directly recognize the current state of M'. In order to overcome the restrictions imposed by the limited controllability and observability, some special input sequences must be utilized in the construction of a checking sequence such that the output sequences produced by M' in response to these input sequences provide sufficient information to deduce that every state transition of M is implemented correctly by M'. That is, in order to verify the state transition from state s_j to s_k under input x ,

a) before the application of x, M' must be transferred to the state recognized as state s_j,

b) the output produced by M' in response to the application of x must be as specified in M,

c) the state reached by M' after the application of x must be recognized as s_k .

Hence, a crucial part of testing the correct implementation of each transition is (1) recognizing the starting and terminating states of the transition (items a) and c) above), (2) verifying the output generated by the transition (item b) above). The recognition of a state of an FSM M can be achieved by using some special sequences called *state identification/verification (SIV) sequences*. Each state in FSM M is uniquely characterized by its SIV sequence(s).

Some commonly used SIV sequences are derived from *distinguishing sequences (D)* [GON70][KOH78], *characterizing sequences* [GIL62][CHO78][KOH78], and *unique input/output (UIO) sequences* [SAB85][SAB88]. Based on distinguishing sequences, characterizing sequences, and UIO sequences, a variety of the formal methods for the construction of checking sequences have been proposed in the literature [GON70] [HEN64] [SID89] [CHA89]. However, these methods do not make an attempt to construct minimal length checking sequences and do not provide proof that the resulting sequences are indeed checking sequences.

In the literature, adaptations of these methods to an FSM-based protocol specification yield a sequence of inputs and outputs, known as a *test sequence*, which, when applied to an implementation of the FSM, is considered to verify all specified transitions. Such applications consist of three phases. In the first phase, SIV sequences are constructed. In the second phase, a *test segment* for each transition in the FSM is formed. A *test segment* for a transition is a sequence of inputs and outputs which consists of an input (with its expected output) that forces the FSM undergo the transition, and an SIV sequence that checks whether the FSM has reached the expected state. Finally, in the third phase, first, the SIV sequences are joined to form a *state identification sequence* and the test segments are joined to form a *transition verification sequence*, then state identification and transition verification sequences are put together to construct a test sequence for the FSM.

According to whether SIV sequences are used or not, or the kind of SIV sequences utilized, test sequence generation methods can be classified as the *transition tour method* (or *T-method*) [NAI81][UYA86] where no SIV sequences are used, the *distinguishing sequence method* (or *DS method* or *D-method*) [SID89], the *characterizing sequence method* (or *W-set method* or *W-method*) [SID89], and the *unique input/output sequences method* (or *UIO-method* or *U-method*) [SAB85][SAB88][AHO88] where the SIV sequences which are derived from the distinguishing sequence, the characterizing sequences and the unique input/output sequences, respectively, are used.

Except the T-method, all other test sequence generation methods reveal faults both in the output and in the next state function of the FSM. The test sequences generated by the T-method can only detect faults in the output function [SID89]. The ability of detecting faults for a test sequence generated by a test sequence generation method is called the *fault coverage* of the method [SID89]. In general, a test sequence generated by these methods should be expected to be as short as possible, while covering as many faults as possible. For example, a modified W-method called *Wp-method* [FUJ91] is considered to be better than the W-method since the Wp-method generates shorter test sequences than those generated by the W-method while maintaining the same fault coverage as the W-method. Similarly, the UIO-method is considered to be better than the D- and W-methods [SID89]. However, contrary to this claim, it is found that the actual fault coverage of the UIO-method is weaker than that of the D-method and the W-method [CHA89], since the uniqueness of UIO sequences may not hold in a faulty implementation of an FSM. A method called UIOv-method has been proposed to overcome the weakness of the UIO-method [CHA89].

Reductions in the length of the test sequences have been achieved by integrating graph theoretic approaches into the third phase of the construction of a test sequence, that is, combining test segments optimally, while maintaining the fault coverage of the related

method. The method proposed by Uyar & Dahbura [UYA86] (henceforth called the *T/CPT method*) combines the T-method with the solution for the Chinese Postman Problem (CPP) [KUA62], for which efficient algorithms exist [EDM73]. The method proposed by Aho *et al.* [AHO88] (henceforth called the *SUIO/RCPT-method*) employs the solution for the Rural Chinese Postman Problem (RCPP) [KUA62] to optimize the length of a test sequence generated by the UIO-method. Finding the solution for the Rural Chinese Postman Problem is known to be NP-complete in the most general case [LEN76]. However, it is shown in [AHO88] that a polynomial-time algorithm exists for determining a minimum-length test sequence if the graph representation of the protocol possesses certain structure (i.e. the protocol satisfies certain sufficiency conditions). The solution for the Rural Chinese Postman Problem has also been applied to the D-method and the W-method, which forms the *D/RCPT-method* and the *W/RCPT-method*, respectively [URA91a].

There is, however, a theoretical limitation on the maximal reduction on the length of a test sequence. Existing efficient algorithms [AHO88], [SAB88], [SHE89], [UYA86] find minimum-length test sequences for FSM's without attempting to overlap test subsequences. Although further reductions in the length of a test sequence can be achieved by allowing such overlaps, it has been shown that to do so is NP-complete in the general case. This implies that it is unlikely that an efficient algorithm exists for finding an optimal length test sequence. It does not, however, preclude the possibility of reducing the length of a test sequence by attempting to overlap test subsequences in a heuristic manner [BOY91].

1.2 Motivation and Objectives

Although above mentioned test sequence generation methods attempt to produce as short a test sequence as possible which may cover as many faults as possible, they still

have some problems in verifying the correctness of an implementation of an FSM. One problem is that some methods, including W_method [CHO78], Wp-method [FUJ91], UIOv-method [VUO89], assume that the "reset" feature (i.e., a reset input taking the FSM from any state to the initial state while producing a null output) is correctly implemented by an implementation under test. This assumption is referred to as *the reliable reset feature* in the literature. However, as already pointed out by some researchers [FUJ91], the reliable reset feature may in some cases be difficult to realize and therefore the approaches cannot be applied. Another problem is that some methods, including SUIO-method [AHO88], MUIO-method [SHE89], don't guarantee that the starting state of a transition in an FSM is recognized before a test segment is applied to test the transition. Therefore, sometimes, these methods can not verify that every state transition of an FSM is implemented correctly.

In this thesis, we focus on the construction of checking sequences rather than test sequences for FSM-based protocol conformance testing, and address the minimization of the length of checking sequences. Our objectives are to:

a) study the effects of having a reliable reset feature versus not having a reliable reset feature of an FSM on the length of the resulting checking sequences,

b) study the effects of interleaving state identification sequences and transition verification sequences on the length of the resulting checking sequences.

c) propose two general models for constructing minimal length checking sequences using distinguishing sequences.

1.3 Contributions of the Thesis

The main contributions of this thesis are as follows:

a) Without interleaving state identification and transition verification sequences, different methods for generating minimal length checking sequences with and without reliable reset feature are studied.

(1) Four methods on generating minimum length test sequences based on D-method, W-method, Wp-method and UIOv-method, all of which assume the reliable reset feature, are reviewed and proved to construct checking sequences.

(2) Four methods on generating minimum length checking sequences based on D-method, W-method, Wp-method and UIOv-method are developed without making reliable reset feature assumption and proved to construct checking sequences.

b) The effects of interleaving the state identification and transition verification sequences on the length of the checking sequences are studied.

(1) An algorithm for interleaving the state identification and transition verification sequences generated by the methods reviewed in a)(1) is proposed.

(2) Two new models for constructing minimal length checking sequences using distinguishing sequences with interleaved state identification and transition verification sequences are proposed.

1.4 Organization of the Thesis

The organization of this thesis is as follows: In Chapter 2, related terminology is introduced, and characteristics of checking sequences are discussed. Chapter 3 first gives a review of generation of checking sequences based on four methods: D-method, W-method, Wp-method and UIOv-method. These methods are applicable to an FSM with a reliable reset feature. Then, in this section, improvements of these four methods to generate minimal length checking sequences from an FSM without making reliable reset feature assumption are proposed. The interleaving of state identification and transition verification

sequences is not considered in this chapter. In Chapter 4, the effects of interleaving state identification and transition verification sequences are analyzed, an algorithm is provided for interleaving these two sequence types, and two general models are proposed for generating minimal length checking sequences based on the use of distinguishing sequences. Finally conclusions and directions for future research are given in Chapter 5.

2. PRELIMINARIES

2.1 The FSM Model and its Graph Representation

An FSM M can be characterized by a quintuple M=(S, I, O, δ, λ), where

S : is a finite set of states $\{s_1, s_2, \ldots, s_n\}$;

I : is a finite set of input symbols;

O : is a finite set of output symbols;

δ : is a mapping S X I -> S, called *next state function*;

λ : is a mapping S X I -> O, called *output function.*

A state transition from state s_j to state s_k in M is denoted by $t_{jk} = (s_j, s_k; i/o)$, where $i \in I$ and $o \in O$. The *initial state* $s_1 \in S$ of M is a designated state for M. Often, there is a designated input $r \in I$, called *reset input*, which brings M to s_1 from every state with a single transition producing a *null* output, i.e., " - " $\in$ O. An FSM M which has such a reset input is said to possess *reset feature*. An FSM M is said to possess *self-loop feature* if for every state in M, there is a transition starting and ending at that state.

For a sequence of inputs $i_1 i_2 \ldots i_r$, $i_j \in I$, $1 \le j \le r$ and a state $s \in S$, $\delta(s, i_1 i_2 \ldots i_r) = \delta(\delta(s, i_1), i_2 \ldots i_r) = s_r$ and $\lambda(s, i_1 i_2 \ldots i_r) = \lambda(s, i_1)\lambda(\delta(s, i_1), i_2 \ldots i_r) = o_1 o_2 \ldots o_r$ where $o_j = \lambda(s_{j-1}, i_j)$, $s_j = \delta(s_{j-1}, i_j)$, $s_0 = s$, $o_j \in O$, $s_j \in S$. In this case, we say that $p = i_1 i_2 \ldots i_r$ is an input sequence, $\lambda(s, p) = o_1 o_2 \ldots o_r = q$ is an output sequence, and $p/q = (i_1 i_2 \ldots i_r)/(o_1 o_2 \ldots o_r) = (i_1/o_1)(i_2/o_2) \ldots (i_r/o_r)$ is an input/output sequence (or IO-sequence, in short) starting at state s.

A sequence $(i_1 i_2 \ldots i_k)$ is a *subsequence* of $(n_1 n_2 \ldots n_m)$ if there exists a Δ, $0 \le \Delta \le m-k$, such that for all j, $1 \le j \le k$, $i_j = n_{j+\Delta}$. A subsequence $(i_1 i_2 \ldots i_k)$ of $(n_1 n_2 \ldots n_m)$ is a *prefix* of $(n_1 n_2 \ldots n_m)$ if $i_j = n_j$, $1 \le j \le k$. A subsequence $(i_1 i_2 \ldots i_k)$ of $(n_1 n_2 \ldots n_m)$ is a *suffix* of $(n_1 n_2 \ldots n_m)$ if $i_j = n_{j+(m-k)}$, $1 \le j \le k$.

An FSM M is said to be *minimal* if every pair of its states does not produce identical output sequences when excited by any sequence of input symbols [KOH78]. An FSM M is said to be *deterministic*, if for each input $i \in I$, there is at most one transition defined at each state of M. An FSM M is said to be *completely specified*, if for each input $i \in I$, there is a transition defined at each state of M.

An FSM M can be represented by a directed graph G=(V, E) where a set of vertices $V=\{v_1,v_2,\cdots,v_n\}$ represents the set of states $S = \{s_1,s_2,...,s_n\}$ of M, and a set of directed edges $E=\{(v_j, v_k; i/o): v_j,v_k \in V\}$ represents all specified transitions of M. Each edge $e_{jk}=(v_j, v_k; i/o) \in E$, represents a state transition from s_j to s_k with input i and output o, where v_j, v_k, and i/o are called the *head*, the *tail*, and the *label* of the edge e_{jk}, and denoted by head(e_{jk}), tail(e_{jk}) and label(e_{jk}), respectively.

In the following, G and M, vertices and states, edges and transitions are used interchangeably.

2.2 Optimization Problems on Graphs

2.2.1 Basic Terminology for Graphs

In this section, definitions related to a directed graph G=(V, E) will be given. It is assumed that the label of an edge in G can be empty (denoted by ε) or a sequence of input/output pairs, $i_1/o_1, i_2/o_2,\ldots, i_k/o_k$, $k > 0$, denoted by L. The *length of the label L* of an edge in G is defined as the number of input/output pairs in L.

Let L, L_1 and L_2 be sequences of input/output pairs:

|L| represents the *length* of L (i.e., the number of input/output pairs in L).

$L_1@L_2$ represents the *concatenation* of L_1 and L_2, which forms a new sequence such that L_1 is followed by L_2.

L_1-L_2 represents the *deduction* of L_2 (a suffix of L_1) from L_1, which forms a new sequence from L_1 such that suffix L_2 in L_1 is removed.

Given a directed graph G=(V, E), the *indegree* and *outdegree* of a vertex v of G is defined as: $d^E_{in}(v) = |\{(w,v; L) : (w,v; L) \in E\}|$ and $d^E_{out}(v) = |\{(v,w; L) : (v,w; L) \in E\}|$. The *index* $\xi^E(v)$ of a vertex $v \in V$ is defined as: $\xi^E(v) = d^E_{in}(v) - d^E_{out}(v)$. G is said to be *symmetric*, if for every $v \in V$, $d^E_{in}(v) = d^E_{out}(v)$, i.e., $\xi^E(v) = 0$.

A *path* $P = (v_1,v_k; L_1@L_2@...@L_{k-1}) = (v_1, v_2; L_1)(v_2, v_3; L_2) \ldots (v_{k-1}, v_k; L_{k-1})$, $k > 1$, is a finite sequence of adjacent and not necessarily distinct edges, where v_1, v_k, and $L_1@L_2@\ldots@L_{k-1}$ are called the *head*, the *tail*, and the *label* of the path P, and denoted by head(P), tail(P) and label(P), respectively. A path $P' = (v_{i1}, v_{i2}; L_{i1})(v_{i2}, v_{i3}; L_{i2}) \ldots (v_{i(k-1)}, v_{ik}; L_{i(k-1)})$ is a *subpath* of $P = (v_{n1}, v_{n2}; L_{n1})(v_{n2}, v_{n3}; L_{n2}) \ldots (v_{n(m-1)}, v_{nm}; L_{n(m-1)})$ if there exists a Δ, $0 \le \Delta \le m-k$, such that for all j, $1 \le j \le k$, $i_j = n_{j+\Delta}$.

G is said to be *strongly connected*, if for every pair of vertices v_j and v_k, there exists a path from v_j to v_k. G is said to be *weakly connected* if its underlying undirected graph is connected.

The *cost* or *length of each edge* of G is equal to the length of its label. The *cost of a path* (or *length of a path*) P in G is the sum of the costs (or lengths) of edges included in P. A path with the minimum length among all paths from v_s to v_t is called a *shortest path* from v_s to v_t.

A *tour* in G is a path in G which starts and ends at the same vertex of G. An *Euler tour* of G is a tour which contains every edge in E exactly once. A *postman tour* (PT) of G is a tour which contains every edge in E at least once. A *rural postman tour* (RPT) of

G=(V, E) over a set $E_C \subseteq E$ is a tour traversing every edge in E_C at least once. A *Chinese postman tour* (CPT) of G is a minimum-cost tour which contains every edge in E at least once. A *rural Chinese postman tour* (RCPT) of G=(V, E) over a set $E_C \subseteq E$ is a minimum-cost tour traversing every edge in E_C at least once.

A graph G=(V, E) is a *subgraph* of G'=(V', E') if $V \subseteq V'$ and $E \subseteq E'$. A subgraph G which contains all the vertices of a graph G' (i.e., V=V') is called a *spanning subgraph* of the graph G'. An *edge-induced subgraph* $G[E_C]$ of G' is a subgraph of G' whose vertex set is the set of heads and tails of edges in E_C, and whose edge set is E_C where $E_C \subseteq E'$ [AHO88]. An edge-induced subgraph $G[E_C]$ of G' is called an *edge-induced spanning subgraph* of G' if its vertex set is V'.

2.2.2 The Rural Chinese Postman Problem

Given a directed graph G=(V, E) and a set $E_C \subseteq E$, the *Rural Chinese Postman Problem* (RCPP) is to find an RCPT of G over the set E_C [AHO88].

It is known that computing an RCPT is NP-complete in the most general case [LEN76]. Nevertheless, if G is strongly connected and the edge-induced spanning subgraph $G[E_C]$ is weakly connected, then any *minimum-cost rural symmetric augmentation* G* of G over E_C is strongly connected [AHO88]. Here, G*=(V*, E*) is a *minimum-cost rural symmetric augmentation* (RSA) of G over E_C if G* is symmetric, V*=V, and E* contains every edge in E_C at least once and every edge in E zero or more times such that the total cost of edges in E* is minimized. Since a strongly connected and symmetric digraph has an Euler tour [EDM73], an RSA G* has an Euler tour which is an RCPT of G over E_C. Therefore, in this special case, the RCPP can be solved in two steps:

(1) constructing an RSA G* of G over E_C;

(2) finding an Euler tour of G*.

Efficient algorithms exist for both of the above steps [AHO88].

If $G[E_C]$ is not weakly connected, edges from E can be added to E_C to obtain E_C^* such that $G[E_C^*]$ is weakly connected. An Euler tour of the RSA G^* of G over E_C^* is then an RPT of G over E_C, but not necessarily an RCPT.

2.3 Fault Detection and Checking Sequences

2.3.1 Fault Detection

Protocol conformance testing is to check if a protocol implementation under test has the same input/output behavior as that of the specification of the protocol. In general, we denote the specification of the protocol as FSM_s and the corresponding implementation as FSM_i. By applying a selected sequence of inputs to the FSM_i and verifying that the corresponding sequence of outputs is as expected according to the FSM_s, we can determine whether the FSM_i conforms to the FSM_s. If two sequences of outputs don't match, the FSM_i is said to be *faulty*.

A testing procedure will only check a limited set of behaviors of an FSM_i for conformance to an FSM_s, since an exhaustive testing of a protocol (i.e., checking all possible input/output behaviors) is not feasible. However, it is not always clear which subset of behaviors is meaningful to ensure the conformance of an FSM_i to the FSM_s it implements.

Current research on protocol conformance testing has included only those behaviors that correspond to testing of the states and individual transitions of an FSM-based protocol. Thus, a transition-level approach originating from sequential circuit checking experiments [KOH78] has been adopted by the formal methods for generating a selected sequence of inputs and corresponding outputs. Accordingly, the procedure of checking whether a

transition $t_{jk}=(s_j, s_k; i/o)$ of a given FSMs is correctly implemented by an FSM_i consists of three steps:

1) The FSM_i is brought to state s_j;
2) Input i is applied to the FSM_i and the output produced by the FSM_i is checked whether it is o;
3) The state the FSM_i reaches after the application of input i is checked whether it is s_k.

Steps 2) and 3) are used to detect the errors in the output function (i.e., output errors) and in the next state function (i.e., transition or tail state errors), respectively. However, this procedure is complicated by the limitations on the controllability and the observability of the FSM_i due to the black box nature of protocol conformance testing. Because of the limited controllability, the FSM_i cannot be directly put into a desired state and thus a *transfer sequence T* (i.e., the label of a shortest path to bring the FSM_i from s_i to a desired state, s_j in step 1)) is usually required. Limited observability prevents the direct observation of the state of the FSM_i, and thus an SIV sequence (i.e., a sequence of inputs and outputs to verify that the FSM_i reached a designated state, s_k in step 3)) is required.

2.3.2 Checking Sequences

Consider a completely specified, minimal, and deterministic FSM $M=(S,I,O,\delta,\lambda)$ which is represented by a strongly connected directed graph G. Let $s_1 \in S$ be the initial state of M. Suppose $\Phi(M)$ be the set of FSMs each of which has at most n states and the same input set as I of M. A *checking sequence* of M is an IO-sequence starting at a specific state of M that determines whether any FSM of $\Phi(M)$ is faulty or not. Since the checking sequence is designed to be applied to the FSM_i in a particular state (the initial state), the FSM_i should be in that particular state when the checking sequence is to be applied. The initial sequence which brings the FSM_i to the initial state may be a synchronizing sequence or a homing sequence (followed possibly by a transfer sequence). A *synchronizing*

sequence for M, which may not exist for every minimal FSM, is an input sequence Z which brings M to a unique state s regardsless of the state of M before its application, i.e., for any $s_i \in S$, $\delta(s_i, Z) = s$. A *homing sequence* for M, which exists for every minimal FSM, is an input sequence H for which the output sequence produced by M in response to H uniquely determines the state M reaches after the application of H i.e., for every pair of $s_i, s_j \in S$, $i \neq j$, $\lambda(s_i, H) \neq \lambda(s_j, H)$ implies $\delta(s_i, H) \neq \delta(s_j, H)$.

The checking sequence, as organized according to the procedure given in [GON70], consists of two parts. The first part will be called the *α-sequence (or state identification sequence)* which recognizes all states in FSM_i and the second part will be called, the *β-sequence (or transition verification sequence)* which verifies all transitions in FSM_i.

To produce the α-sequence, SIV sequences have to be used. As mentioned in section 1.1, the most commonly used SIV sequences are derived from distinguishing sequences (DS), characterizing sequences and unique input/output (UIO) sequences.

A sequence of inputs D is said to be a *distinguishing sequence* of FSM M if the output sequence produced by M in response to D is different for each state of M i.e., for every pair of $s_i, s_j \in S$, $i \neq j$, $\lambda(s_i, D) \neq \lambda(s_j, D)$. A distinguishing sequence may not exist for every FSM [KOH78]. An SIV sequence for a state s_k based on a D for the given FSM (henceforth denoted by D_k for state s_k) is a sequence of inputs and corresponding outputs where the sequence of inputs is D.

A set of *characterizing sequences (W-set or characterization set)* for an FSM M is a set of input sequences $w_1, w_2, \ldots, w_m$ such that each state is uniquely identified by the set of output sequences produced by M in response to the set of input sequences [GIL62][CHO78]. There always exists a set of characterizing sequences for a given FSM. A W-set (henceforth denoted by $\{w_1, w_2, \ldots, w_m\}$) for an FSM M is said to be a *minimal*

W-set if none of its components is a prefix of another component [GIL62]. A set of SIV sequences for a state s_k can be derived from a W-set for the given FSM (henceforth denoted by $\{W_{k1},W_{k2},...,W_{km}\}$) where W_{ki}, i=1,2,...,m is a sequence of inputs and corresponding outputs and sequence of inputs is the w_i in the W-set. If we choose Y_{ki} ($1 \leq i \leq m$) to be some sequence for state s_k that has the W_{ki} as its prefix and that takes the correctly-operating FSM from state s_k back to state s_k, then we form:

$$\gamma_1 = Y_{k1}$$

$$\gamma_2 = (\gamma_1)^{n+1}Y_{k2}$$

$$\gamma_3 = (\gamma_2)^{n+1}(\gamma_1)^{n+1}Y_{k3}$$

...

$$\gamma_m = (\gamma_{m-1})^{n+1}(\gamma_{m-2})^{n+1}...(\gamma_1)^{n+1}W_{km}$$

where γ_m is said to be *locating sequence* for state s_k which is denoted by LO_k. The input portion of LO_k is denoted by I_k, for state $s_k \in S$, $1 \leq k \leq n$. Let $LO=\{LO_1,LO_2,...,LO_n\}$ be a set of locating sequences for M. The locating sequence for state s enables us to locate a state in an implementation of M that behaves like s with respect to the characterizing sequences derived from M [HEN64].

A *unique input/output (UIO) sequence* for a state of FSM M is an input/output behavior that is not exhibited by any other state of M. UIO sequences may not exist for some states of a given FSM [SAB88]. In that case, a *unique signature* [SAB88] is constructed for each state which does not have a UIO sequence. An SIV sequence for a state based on a UIO sequence or a unique signature for that state (henceforth denoted by U_k for state s_k) is simply a UIO sequence or a unique signature for that state.

Therefore, the input sequence required to identify the state s_j consists of

T_a = the input portion of an optional transfer sequence to bring the FSM_i from its current state to state s_j;

T_b = the input portion of an SIV sequence for state s_j.

The sequence of inputs (together with their expected outputs) which cover T_a and T_b for each state in FSM_s forms the α-sequence.

To generate the β-sequence, one has to consider the input sequence required to verify each transition $t_{jk}=(s_j, s_k; i/o)$ which consists of

T_c = the input portion of an optional transfer sequence to bring the FSM_i from its current state to state s_j;

T_d = input symbol i for stimulating transition t_{jk};

T_e = the input portion of an SIV sequence for state s_k.

The sequence of inputs (together with their expected outputs) which cover T_d and T_e for a transition is called a *test segment* for the transition. In general, the test segment for the transition $t_{jk}=(s_j, s_k; i/o)$ is defined as $i/o@SIV(s_k)$ where @ is the string concatenation operator.

The sequence of inputs (together with their expected outputs) which cover T_c, T_d and T_e for each transition in FSM_s forms the β-sequence.

2.3.3 Optimization of the Length of Checking Sequences

It should be noted that the formation of a checking sequence may be subject to certain constraints, such as which type of SIV sequences are used, the existence of a requirement on the starting and terminating states of the checking sequence, whether the FSM has reliable reset feature, and whether interleaving between α-sequences and β-sequences is allowed. Based on these constraints, different optimization problems may be formed.

In the following, we assume that a deterministic, minimal, completely specified FSM M is represented by a strongly connected digraph G=(V,E) in which the cost (or length) of each edge is one. Also, it is assumed that the checking sequence derived from an FSM M is a tour or a path of G, which starts at the initial state s_1 of M. This implies the assumption that a given FSM_i always starts from its initial state in the conformance testing. Finally, minimum-length or minimum-cost checking sequence will be specifically used to refer to a checking sequence derived by an algorithm using certain optimization approaches, and a minimum-length checking sequence may or may not be an optimal length checking sequence.

2.3.4 Properties of a Checking Sequence

We are interested in the construction of an IO-sequence Q (= $L_1 \ldots L_{k-1}$, $k > 1$, $L_j=x_j/y_j$, $1 \le j \le k-1$) of M starting at v_1, which is the label of a path $P = (v_1,v_2; L_1)$ $(v_2,v_3; L_2)$ $\ldots(v_{k-1},v_k; L_k)$ of G. We shall prove that such an IO-sequence is in fact a checking sequence of M that starts at v_1. First, we define recognition of a vertex v_i of P in Q as some state of M, then define verification of an edge e = (a, b; x/y) of G in Q. In the following definitions and proofs, we consider two cases where the SIV is either a distinguishing sequence or a W-set of M.

(1) Suppose the SIV is a distinguishing sequence of M

I. *Recognition* of a vertex

Let depth(v_i)=∞ initially.

1. A vertex v_i of P is *d-recognized* in Q as state a of M if v_i is followed immediately by an IO-sequence D/λ(a, D), namely, $L_iL_{i+1} \ldots L_{i+m-1}$ =D/λ(a, D) and depth(v_i)=0. For a special case, if M has reliable reset feature, i.e., (v_x, v_i; r/-) is a subpath of P, then v_i is *d-recognized* in Q as initial state of M.

2. Suppose that $(v_q, v_i; T)$ and $(v_j, v_k; T)$ are subpaths of P and $D/\lambda(a, D)$ is a prefix of T, namely, v_q, v_j are d-recognized in Q as some state a of M, and suppose that v_k is d-recognized in Q as state a' of M. Then v_i is *t-recognized* in Q as state a' of M, and $depth(v_i)=1$.

3. Suppose that $(v_q, v_i; T)$ and $(v_j, v_k; T)$ are subpaths of P such that v_q, v_j are either d-recognized or t-recognized in Q as some state a of M, and v_k is either d-recognized or t-recognized in Q as state a' of M. Then v_i is *t-recognized* in Q as state a' of M, and $depth(v_i)=1+\max\{depth(v_q), depth(v_j), depth(v_k)\}$.

4. If vertex v_i of P is either d-recognized or t-recognized in Q as some state a of M, then it is said to be *recognized* as state a. A vertex of P is *recognized* if it is recognized as some state a .

II. *Verification* of an edge

An edge $e = (a, b; x/y)$ of G is *verified* in Q if there is an edge $(v_i, v_{i+1}; x_i/y_i)$ in P such that v_i and v_{i+1} are recognized in Q as states a, b of M respectively, and $x_i/y_i = x/y$.

Below, we prove that an IO-sequence in which every edge of G is verified is in fact a checking sequence of M that starts at v_1.

Proposition 1 Let Q be an IO-sequence of M starting at v_1 (i.e., the label of a path $P=(v_1, v_k; Q)$ of G) and let D be a distinguishing sequence for M. If every edge of G is verified in Q, then for each v_i, $1 \le i \le n$ of G, $D/\lambda(v_i, D)$ is a subsequence of Q.

Proof: For each vertex a of G, let (a, b; x/y) be an edge of G. Since (a, b; x/y) is verified in Q, there exists an edge $(v_i, v_{i+1}; x_i/y_i)$ of P such that v_i is recognized as a.

Let v_r be the vertex of P such that v_r is recognized as a and $depth(v_r)$ is minimized. Suppose that $depth(v_r)>0$. Then there exist subpaths $(v_q, v_r; T)$ and $(v_j, v_t; T)$ of P such

that v_q and v_j are recognized as state a', v_t is recognized as state a, and depth(v_r)=1+max{depth(v_q), depth(v_j), depth(v_t)}. So depth(v_t)<depth(v_r), which contradicts the fact that depth(v_r) is minimized. Hence, depth(v_r)=0, that is v_r is d-recognized as a. So D/λ(a, D) is a subsequence of Q. Since a is arbitraraly chosen, D/λ(v_i, D) is a subsequence of Q, for each v_i, $1 \le i \le n$ of G. ■

Let M' be an FSM of Φ(M). FSM M is isomorphic to FSM M', denoted by M ≡ M', if there is a one-to-one function f on the state sets of M and M' such that for any state transition (i, j; x/y) of FSM M, (f(i), f(j)); x/y) is a transition of FSM M'.

Proposition 2 Let Q be an IO-sequence of M starting at v_1 (i.e., the label of a path P = (v_1, v_k; Q) of G) and let D be a distinguishing sequence for M. Suppose that every edge of G is verified in Q and Q is also an IO-sequence of M'=(S', X, Y', δ', λ') of Φ(M) starting at state v_1'. If state v_i is recognized in Q as a and $v_i' = \delta'(v_1', x_1...x_{i-1})$, then λ'($v_i'$, D)=λ($v_i$, D)=λ(a, D).

Proof: We prove the claim by induction on the depth of v_i. If depth(v_i)=0, then v_i is d-recognized in Q as a, then λ'(v_i', D)=λ(v_i, D)=λ(a, D). Suppose that v_i is not d-recognized in Q as a. Then there exist subpaths (v_q, v_i; T) and (v_j, v_t; T) of P with both v_q and v_j recognized as b, and v_t recognized as a. By inductive hypothesis, λ'(v_j', D) = λ(v_j, D)=λ(b, D)=λ(v_q, D)=λ'(v_q', D) and λ'(v_t', D)=λ(v_t, D)=λ(a, D). Since M' has n states and D is a distinguishing sequence of M' (from Proposition 1), v_q' and v_j' are the same state of M'. So v_i' and v_t' are the same state. Hence λ'(v_i', D) = λ'(v_t', D) = λ(v_t, D) = λ(v_i, D) = λ(a, D) ■

Theorem 1 Let Q be an IO-sequence of M starting at v_1 (i.e., the label of a path P =(v_1, v_k; Q) of G) and let D be a distinguishing sequence for M. If every edge (i, j; x/y) of G is verified in Q, then Q is a checking sequence of M.

Proof: Let M' be an arbitrary FSM of $\Phi(M)$ such that Q is also an IO-sequence in M' starting at a vertex v'. Then, M' has n distinct states $b_1,...,b_n$ such that $\lambda'(b_i, D) = \lambda(i, D)$ by Proposition 1 and Proposition 2.

Now let f be the mapping from M to M' such that $f(i)=b_i$. It is clear that this mapping is well-defined, one-to-one, and onto. Let (b, b'; x/y) be the edge of M' in a path $P' =(v_1, v_k; Q)$ corresponding to (i, j; x/y) in P. Then b is recognized as i and b' is recognized as j. By Proposition 2, b=f(i) and b'=f(j). So (f(i), f(j); x/y) is a transition of M'. Hence $M \equiv M'$ and Q is a checking sequence of M. This completes the proof. ■

To check whether Q is an IO-sequence of FSM M', we need to bring FSM M' to the state corresponding to the initial state v_1 if $M \equiv M'$. This can be done by applying a homing sequence of M (followed possibly by a transfer sequence).

(2) Suppose the SIV is a W-set of M

The properties of the locating sequence LO_s for state s are such that the set of input sequences $w_1,w_2,...,w_m$ is applied to the state $\delta(s,I_s\backslash w_m))$, where I_s is the input portion of LO_s and $I_s\backslash w_m$ is the sequence of inputs obtained after suffix w_m is deleted from I_s. $\delta(s,I_s\backslash w_m))$ is defined as the *LO_s-related state.*

I. *Recognition* of a vertex

1. Suppose that $(v_j, \delta(v_j, I_a\backslash w_m); (I_a\backslash w_m)/\lambda(a, I_a\backslash w_m))@(\delta(v_j, I_a\backslash w_m), \delta(\delta(v_j, I_a\backslash w_m), w_m); w_m/\lambda(a, w_m)) = (v_j, \delta(v_j, I_a); I_a/\lambda(a,I_a))$ is a subpath of P where $\delta(v_j, I_a\backslash w_m) = v_i$. Then, v_i is *l-recognized* in Q as state a of M, i.e., v_i is an LO_a-related state. Note that only v_i is *l*-recognized as a, but v_j or $\delta(v_j, I_a)$ are not.

2. Suppose that each $(v_x, \delta(v_x, w_t); w_t/\lambda(a, w_t))$, $1 \le t \le m$, is a subpath of P, and $v_x = \delta(u_x, w_m)$, where u_x is *l*-recognized in P as the same state of M. Then, v_x are *w*-

recognized in P as state a of M. For a special case, if M has reliable reset feature, i.e. $(v_x, v_i; r/-)$ is a subpath of P, then v_i is *w-recognized* in Q as initial state of M.

3. Suppose that $(v_q, v_i; T)$ and $(v_{j_t}, v_{k_t}; T)$, $1 \le t \le m$, are subpaths of P such that v_q and v_{j_t} are *l*-recognized in Q as some state a of M , and suppose that v_{k_t} are w-recognized in Q as state a' of M. Then v_i is *t-recognized* in Q as state a' of M. Note that input portion of T must contain w_m as a prefix.

4. Suppose that $(v_q, v_i; T)$ and $(v_{j_t}, v_{k_t}; T)$, $1 \le t \le m$, are subpaths of P such that v_q and v_{j_t} are t-recognized in Q as some state a of M , and suppose that v_{k_t} are w-recognized in Q as state a' of M. Then v_i is *t-recognized* in Q as state a' of M.

II. *Verification* of an edge

An edge e = (a, b; x/y) of G is *verified* in Q if

(a) there are subpaths $(v_{i_t}, v_{j_t}; x_i/y_i)$, $1 \le t \le m$, of P such that v_{i_t} are t-recognized in Q as state a of M and v_{j_t} are w-recognized in Q as state b of M.

(b) $x_i/y_i = x/y$.

(c) there are subpaths $(v_k, v_m; I_k/\lambda(s_k,I_k))$ of P, for each $s_k \in S$, i.e., there must be a vertex $v_l = \delta(v_k, I_k \backslash w_m)$ in P which is *l*-recognized in Q as state s_k of M.

Proposition 3 Let an IO-sequence Q of M be the label of a path $P = (v_1, v_k; Q)$ of G. If every edge of G is verified in Q, then for each $s_i \in S$,

(a) $w_t/\lambda(s_i,w_t)$, $1 \le t \le m$, are subsequences of Q,

(b) $I_i/\lambda(s_i, I_i)$ are subsequences of Q.

Proof: Let (a, b; x/y) be any edge of G. Since (a, b; x/y) is verified in Q, by the definition of verification of an edge, there exists m edges $(v_{i_t}, v_{j_t}; x_i/y_i)$, $1 \le t \le m$, of P such that each v_{j_t} is w-recognized as state b of M. By the definition of w-recognition, $w_t/\lambda(b,w_t)$, $1 \le t \le m$, are subsequences of Q. Since G is strongly connected every vertex is a tail state of some edge in G, thus $w_t/\lambda(s_i,w_t)$, $1 \le t \le m$, for all $s_i \in S$, are subsequences of Q. It also follows from the definition of verification of an edge that, for each state $s_i \in S$, there must be a vertex in P which is l-recognized as s_i of M, that is, $I_i/\lambda(s_i, I_i)$, $1 \le i \le n$, are subsequences of Q. ■

Proposition 4 Let an IO-sequence Q of M be the label of a path $P = (v_1, v_k; Q)$ of G, every edge of G be verified in Q and Q also be an IO-sequence of $M'=(S', X, Y', \delta', \lambda')$ of $\Phi(M)$ starting at state v_1'. If vertex v_i is w- or t- recognized in Q as state a of M where $v_i = \delta(v_1, x_1...x_{i-1})$ and $v_i' = \delta'(v_1', x_1...x_{i-1})$, then $\lambda'(v_i', w_t)=\lambda(v_i, w_t)=\lambda(a, w_t)$, $1 \le t \le m$.

Proof: Since M' has n states, Q is an IO-sequence of M' and by Proposition 3, Q contains $w_t/\lambda(s_i,w_t)$ $(1 \le t \le m)$ and $I_i/\lambda(s_i, I_i)$, $1 \le i \le n$, it follows that $W = \{w_1, w_2,..., w_m\}$ is a characterization set for M' also, and $LO= \{LO_1, LO_2,..., LO_n\}$ is a set of locating sequences for M'. If v_i is w- or t-recognized in Q as state a of M, $\lambda(v_i, w_t) = \lambda(a, w_t)$, $1 \le t \le m$, by the definition of w- and t-recognition. Since $v_i = \delta(v_1, x_1,...,x_{i-1})$, $v_i'= \delta'(v_1', x_1,...,x_{i-1})$, and Q is an IO-sequence for both M and M', $\lambda'(v_i', w_t) = \lambda(v_i, w_t)$, $1 \le t \le m$. It follows that $\lambda'(v_i',w_t) = \lambda(v_i,w_t) = \lambda(a,w_t)$, $1 \le t \le m$. ■

Theorem 2 Let an IO-sequence Q of M be the label of a path $P = (v_1, v_k; Q)$ of G. If every edge (i, j; x/y) of G is verified in Q, then Q is a checking sequence of M.

Proof: Let M' be an arbitrary FSM of $\Phi(M)$ such that Q is also an IO-sequence of M' starting at a vertex v_1'. Then, M' has n distinct states $b_1,...,b_n$ such that $\lambda'(b_i, w_t) = \lambda(i, w_t)$, $1 \le t \le m$, by Proposition 3 and Proposition 4, with f being the mapping from M to

M' such that f(i)=b_i. It is clear that this mapping is a bijection. Let (v_i', v_j'; x/y) be an edge in the path P' = (v_1', v_k'; Q) of G' corresponding to (v_i, v_j; x/y) in P, where edge (i,j;x/y) of G is verified, thus, v_i is t-recognized as i and v_j is w-recognized as j. By Proposition 4, v_i' is t-recognized as i and v_j' is w-recognized as j in P'. Since (v_i', v_j'; x/y) is an edge of G', v_i' is also t-recognized as b_i = f(i) and v_j' is also w-recognized as b_j = f(j). So, (f(i), f(j); x/y) is verified to be a transition of M'. Since every edge of G is verified in Q, it is true for any transition (i,j;x/y) of M that (f(i), f(j); x/y) is a transition of M'. Hence M ≡ M'. Then, Q is a checking sequence of M. ■

3. GENERATION OF CHECKING SEQUENCES WITHOUT INTERLEAVING α AND β SUBSEQUENCES

Several methods have been proposed for generating checking sequences from an FSM_S. This chapter focuses on reviewing four generic methods with respect to the case where it is assumed that the reset function is correctly implemented in an FSM_I. Then, in the second part of this section, we extend these methods to the case where it is not assumed that the reset function is correctly implemented in an FSM_I. For these two cases, we consider the construction of a minimal-length checking sequence (abbreviated as MLCS) using D-method [HEN64],[GON70], W-method [CHO78], Wp-method [FUJ91] and UIOv-method [CHA89].

3.1 FSM_I with a Reliable Reset Feature

3.1.1 Constructing an MLCS Using a Distinguishing Sequence

Assume that the FSM_I has a reliable reset feature and the FSM_S is minimal, strongly connected, completely specified and has at least one distinguishing sequence. The D-method can be used to construct a minimal length checking sequence as follows:

Method_1:

1) For the given FSM_S, compute a distinguishing sequence, and denote it as D.

2) For the FSM_S, compute a minimum spanning tree (MST) rooted at the initial state (s_1). Use the labels of all partial paths starting at the root of MST as transfer sequences, i.e. $TR(s_1,s_k)$ is the label of the partial path from s_1 to s_k in MST, $\forall s_k \in S$.

3) Construct an α-sequence (state identification)

a) $\forall s_k \in S$, form D_k which is a sequence of labels obtained by applying D to s_k.

b) $\forall s_k \in S$, form a path $P_k = (v_x,v_1;\ r/-)@(v_1,v_k;\ TR(s_1,s_k))@(v_k,v_z;\ D_k)$ and denote label(P_k) by $\alpha_k = r/-@TR(s_1,s_k)@D_k$.

c) Eliminate P_i whose label is a prefix of the label of some P_j, $1 \le i, j \le n$, $i \ne j$. Concatenate all remaining P_i to form P_α.

d) Label(P_α) is the α-sequence.

4) Construct a β-sequence (transition verification)

a) $\forall t_{jk} = (s_j,\ s_k;\ i/o) \in FSM_s$, form a path $P_{jk} = (v_x,v_1;\ r/-)@(v_1,v_j;\ TR(s_1,s_j))@(v_j,v_k;\ i/o)@(v_k,v_z;\ D_k)$ and denote label(P_{jk}) by $\beta_{jk} = r/-@TR(s_1,s_j)@label(t_{jk})@D_k, 1 \le j, k \le n$.

b) Eliminate P_{jk} whose label is a prefix of the label of some P_{lm}. $1 \le l, m \le n$ Concatenate all remaining P_{jk} to form P_β.

c) Label(P_β) is the β-sequence.

5) The minimum-length checking sequence for FSM_s is:

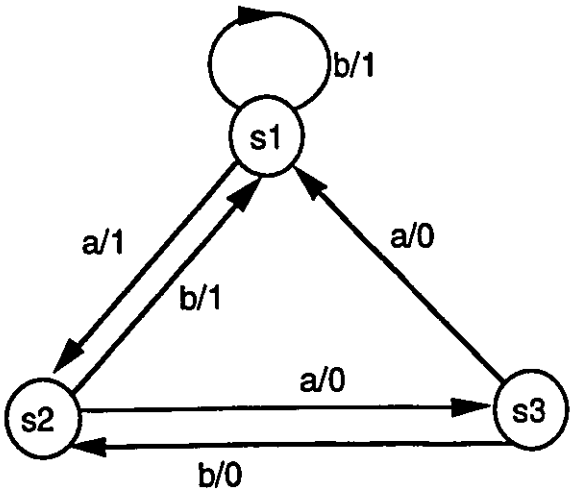
Label($P=P_\alpha@P_\beta$) = α-sequence @ β-sequence ■

3.1.2 An Example Using Method_1

We apply Method_1 to the FSM_s M in Figure 1:

1) A distinguishing sequence of the FSM_s is D = ab.

2) The MST of FSM_s is shown in Figure 2.



input / state	a	b	D=ab
s1	s2/1	s1/1	s1/11
s2	s3/0	s1/1	s2/00
s3	s1/0	s2/0	s1/01

(The reset transitions are not shown)

Figure 1 An Example FSM_S

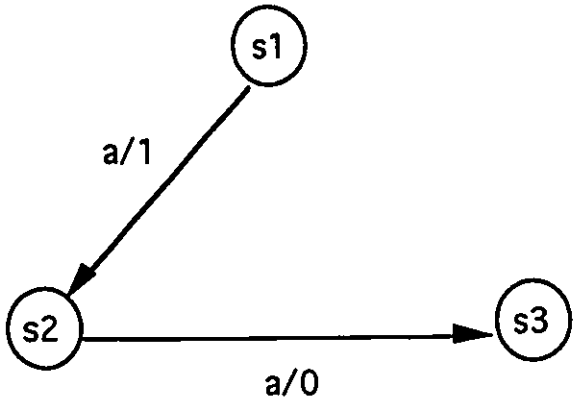


Figure 2 Minimum Spanning Tree of the FSM_S in Figure 1

3) α-sequence:

α_1 r/- a/1 b/1

α_2 r /- a/1 a/0 b/0

α_3 r /- a/1 a/0 a/0 b/1

The input portion of α-sequence: rabraabraaab

4) β-sequence:

β_{11} r/- b/1a/1b/1 β_{21} r/- a/1b/1a/1b/1 β_{31} r/- a/1a/0a/0a/1b/1

β_{12} r/- a/1a/0b/0 β_{23} r /- a/1a/0a/0b/1 β_{32} r /- a/1a/0b/0a/0b/0

The input portion of β-sequence: rbabrababraaabraaaabraabab

where the underlined are the transitions to be verified.

(Since β_{12} is a prefix of β_{32}, it is eliminated).

5) The checking sequence (input portion):

rabraabraaabrbabrababraaabraaaabraabab

Its length is 38 and its ending state is s_2.

3.1.3 Constructing an MLCS Using an Expanded W-set

Assume that the FSM_i has a reliable reset feature and the FSM_s is minimal, strongly connected, and completely specified. Expanded W-set method can be used to construct a minimal length checking sequence as follows:

Method_2:

1) For the given FSM_s, compute the expanded W-set and denote it as $W=\{w_1, w_2,\ldots,w_m\}$

2) For the FSM_s, compute a minimum spanning tree (MST) rooted at the initial state (s_1). Use the labels of all partial paths starting at the root of MST as transfer sequences, i.e. $TR(s_1,s_k)$ is the label of the partial path from s_1 to s_k in MST, $\forall s_k \in S$.

3) Construct an α-sequence (state identification)

a) $\forall s_k \in S$, form $W_{k1}, W_{k2}, \ldots, W_{km}$, which are sequences of labels obtained by applying $w_1, w_2, \ldots, w_m$ to s_k, respectively.

b) $\forall s_k \in S$, form a path $P_{ki} = (v_x, v_1; r/-)@(v_1, v_k; TR(s_1, s_k))@(v_k, v_{zi}; W_{ki})$ and denote label(P_{ki}) by $\alpha_{ki} = r/-@TR(s_1, s_k)@W_{ki}$ $(i=1,\ldots,m)$.

c) Eliminate P_{ij} whose label is a prefix of the label of some $P_{i'j'}$, $1 \le i' \le n$, $1 \le j' \le m$, $i \ne i'$. Concatenate all remaining P_{ij} to form P_α.

d) Label(P_α) is the α-sequence.

4) Construct a β-sequence (transition verification)

a) $\forall t_{jk} = (s_j, s_k; i/o) \in FSM_s$, form a path $P_{jki} = (v_x, v_1; r/-)@(v_1, v_j; TR(s_1, s_j))@(v_j, v_k; i/o)@(v_k, v_{zi}; W_{ki})$ and denote label(P_{jki}) by $\beta_{jki} = r/-@TR(s_1, s_j)@label(t_{jk})@W_{ki}$, $(i=1,..,m)$, $1 \le j, k \le n$.

b) Eliminate P_{jki} whose label is a prefix of the label of some $P_{j'k'i'}$. Concatenate all remaining P_{jki} to form P_β, $1 \le j', k' \le n$, $1 \le i' \le m$.

c) Label(P_β) is the β-sequence.

5) The minimum-length checking sequence for FSM_s is:

Label($P=P_\alpha@P_\beta$) = α-sequence@β-sequence ■

3.1.4 An Example Using Method_2

We apply Method_2 to the FSM_s M in Figure 1:

1) A W-set of the FSM_s is w_1=a, w_2=b, W={w_1,w_2}.

Table 1 A W-set applied to every state of the FSM_s

state \ input	a	b
s1	W_{11}=a/1	W_{12}=b/1
s2	W_{21}=a/0	W_{22}=b/1
s3	W_{31}=a/0	W_{32}=b/0

2) The MST of FSM_s is shown in Figure 2.

3) α-sequence:

α_{11} r/- a/1 α_{12} r/- b/1

α_{21} r/- a/1 a/0 α_{22} r/- a/1 b/1

α_{31} r/- a/1 a/0 a/0 α_{32} r/- a/1 a/0 b/0

The input portion of α-sequence: rbrabraaaraab

(Since α_{11} and α_{21} are prefixes of α_{31}, they are eliminated).

4) β-sequence:

β_{111} r/- b/1 a/1 β_{211} r/- a/1b/1 a/1 β_{311} r/- a/1a/0 a/0 a/1

β_{112} r/- b/1 b/1 β_{212} r/- a/1b/1 b/1 β_{312} r/- a/1a/0 a/0 b/1

β_{121} r/- a/1 a/0 β_{231} r/- a/1a/0 a/0 β_{321} r/- a/1a/0 b/0 a/0

β_{122} r/- a/1 b/1 β_{232} r/- a/1a/0 b/0 β_{322} r/- a/1a/0 b/0 b/1

The input portion of β-sequence: rbarbbrabarabbraaaaraaabraabaraabb

where the underlined are the transitions to be verified.

(Since β_{121} and β_{231} are prefixes of β_{311}, β_{122} is a prefix of β_{211}, β_{232} is a prefix of β_{321}, they are eliminated).

5) The checking sequence (input portion):

rbrabraaaraabrbarbbrabarabbraaaaraaabraabaraabb

Its length is 47 and its ending state is s_1.

3.1.5 Constructing an MLCS Using a Partial W-set

Assume that the FSM_i has a reliable reset feature and the FSM_s is minimal, strongly connected, and completely specified. Partial W-set method can be used to construct a minimal length checking sequence as follows:

Method_3:

1) For the given FSM_s, compute the expanded W-set and denote it as $W=\{w_1, w_2,..., w_m\}$

2) For the FSM_s, compute a minimum spanning tree (MST) rooted at the initial state (s_1). Use the labels of all partial paths starting at the root of MST as transfer sequences, i.e. $TR(s_1,s_k)$ is the label of the partial path from s_1 to s_k in MST, $\forall s_k \in S$.

3) Construct an α-sequence (state identification)

a) $\forall s_k \in S$, form $W_{k1},W_{k2},...,W_{km}$, which are sequences of labels obtained by applying $w_1,w_2,...,w_m$ to s_k, respectively.

b) $\forall s_k \in S$, form a path $P_{ki} = (v_x,v_1; r/-)@(v_1,v_k; TR(s_1,s_k))@(v_k,v_{zi}; W_{ki})$ and denote label(P_{ki}) by $\alpha_{ki} = r/-@TR(s_1,s_k)@W_{ki}$ (i=1,...,m).

c) Eliminate P_{ij} whose label is a prefix of the label of some $P_{i'j'}$, $1 \le i' \le n$, $1 \le j' \le m$, $i \ne i'$. Concatenate all remaining P_{ij} to form P_α.

d) Label(P_α) is the α-sequence.

4) $\forall s_k \in S$, form an identification set W_k, which is a subset of W, and can distinguish the state s_k from all other states.

5) Construct a β-sequence (transition verification)

a) $\forall t_{jk} = (s_j, s_k; i/o) \in FSM_s$, form a path $P_{jki} = (v_x, v_1; r/-)@(v_1, v_j; TR(s_1, s_j))@(v_j, v_k; i/o)@(v_k, v_{zi}; W_{ki})$ and denote label(P_{jki}) by β_{jki} = r/-@$TR(s_1, s_j)$@label(t_{jk})@W_{ki}, (i=1,..,m, $1 \le j, k \le n$), where W_{ki} is the sequence of labels obtained by applying the $w_i \in W_k$ to state s_k.

b) Eliminate P_{jki} whose label is a prefix of the label of some $P_{j'k'i'}$. Concatenate all remaining P_{jki} to form P_β, $1 \le j', k' \le n$, $1 \le i' \le m$.

c) Label(P_β) is the β-sequence.

6) The minimum-length checking sequence for FSM_s is:

Label($P = P_\alpha @ P_\beta$) = α-sequence@β-sequence ■

3.1.6 An Example Using Method_3

We apply Method_3 to the FSM_s M in Figure 1:

1) A W-set of the FSM_s is w_1=a, w_2=b, W={w_1,w_2}

2) The MST of FSM_s is shown in Figure 2.

3) α-sequence:

α_{11} r/- a/1 α_{12} r/- b/1

α_{21} r/- a/1 a/0 α_{22} r/- a/1 b/1

α_{31} r/- a/1 a/0 a/0 α_{32} r/- a/1 a/0 b/0

The input portion of α-sequence: rbrabraaaraab

(Since α_{11} and α_{21} are prefixes of α_{31}, they are eliminated).

4) Identification set for each state: W_1={a}, W_2={a,b}, W_3={b}

5) β-sequence:

β_{111} r/- b/1 a/1 β_{211} r/- a/1b/1 a/1 β_{321} r/- a/1a/0 b/0 a/0

β_{121} r/- a/1 a/0 β_{232} r/- a/1a/0 b/0 β_{322} r/- a/1a/0 b/0 b/1

β_{122} r/- a/1 b/1 β_{311} r/- a/1a/0 a/0 a/1

The input portion of β-sequence: rbarabaraaaaraabaraabb

where the underlined are the transitions to be verified.

(Since β_{121} is a prefix of β_{232}, β_{122} is a prefix of β_{211}, β_{232} is a prefix of β_{321}, they are eliminated).

6) The checking sequence(input portion):

rbrabraaaraabrbarabaraaaaraabaraabb

Its length is 35 and its ending state is s_1.

3.1.7 Constructing an MLCS Using UIO Sequences

Assume that the FSM_i has a reliable reset feature and the FSM_s is minimal, strongly connected, and completely specified. Also, assume that the FSM_s has unique input output (UIO) sequence for each state. Then, the UIO sequence of each state of FSM_s must be verified to be unique in FSM_i. UIOv method can be used to construct a minimal length checking sequence as follows:

Method_4:

1) For the given FSM_s, compute a minimal length UIO sequence U_k for each state s_k and form $U=\{U_1, U_2,..., U_n\}$, the input portion of each U_i can be denoted by u_i, i=1, ..., n.

2) For the FSM_s, compute a minimum spanning tree (MST) rooted at the initial state (s_1). Use the labels of all partial paths starting at the root of MST as transfer sequences, i.e. $TR(s_1,s_k)$ is the label of the partial path from s_1 to s_k in MST, $\forall s_k \in S$.

3) Construct an α-sequence (state identification)

   a) $\forall s_k \in S$, form a path $P_k = (v_x,v_1;\ r/-)@(v_1,v_k;\ TR(s_1,s_k))@(v_k,v_z;\ U_k)$ and denote label(P_k) by $\alpha_k = r/-@TR(s_1,s_k)@U_k$.

   b) $\forall s_k \in S$, form a path $\sim P_{kj} = (v_x,v_1;\ r/-)@(v_1,v_k;\ TR(s_1,s_k))@(v_k,v_z;\ u_j/\lambda(s_k,u_j)$ where u_j ($1 \le j \le n, j \ne k$) is the input sequence which is not a prefix of the input portion of U_k. The label of $\sim P_{kj}$ can be denoted by $\sim\alpha_{kj}$, k= 1, 2, ..., n, $1 \le j \le n$.

   c) Eliminate P_i or $\sim P_{ir}$ whose label is a prefix of the label of some P_j or $\sim P_{jt}$, $1 \le j \le n$, $1 \le t, r \le n$, $i \ne j$. Concatenate all remaining P_i or $\sim P_{ir}$ to form P_α.

d) Label(P_α) is the α-sequence.

4) Construct a β-sequence (transition verification)

a) $\forall t_{jk} = (s_j, s_k; i/o) \in FSM_s$, form a path $P_{jk} = (v_x,v_1; r/-)@(v_1,v_j; TR(s_1,s_j))@(v_j,v_k; i/o)@(v_k,v_z; U_k)$ and denote label(P_{jk}) by β_{jk} = r/-@TR(s_1,s_j)@label(t_{jk})@U_k, $1 \le j, k \le n$.

b) Eliminate P_{jk} whose label is a prefix of the label of some P_{lm}. $1 \le l, m \le n$

Concatenate all remaining P_{jk} to form P_β.

c) Label(P_β) is the β-sequence.

5) The minimum-length checking sequence for FSM_s is:

Label($P=P_\alpha@P_\beta$) = α-sequence @ β-sequence ■

The above method is provided based on UIOv method [CHA89]. However, in the literature [FUJ90], it can be seen that the UIOv method is a special case of Wp-method, so, in section 3.2.7, we will treat the extension of UIOv-method as a special case of the extension of Wp-method.

3.1.8 An Example Using Method_4

We apply Method_4 to the FSM_s M in Figure 1:

1) A minimal length UIO sequence for each state is:

U_1 = a/1

U_2 = a/0 a/0

U_3 = b/0

2) The MST of FSM_s is shown in Figure 2.

3) α-sequence:

α_1 r/- a/1

α_2 r/- a/1 a/0 a/0

α_3 r/- a/1 a/0 b/0

$\sim\alpha_{11}$ r/- b/1

$\sim\alpha_{12}$ r/- a/1 a/0

$\sim\alpha_{21}$ r/- a/1 b/1

$\sim\alpha_{31}$ r/- a/1 a/0 a/0 a/1

The input portion of α-sequence: raabrbrabraaaa

(Since α_1 and α_2 and $\sim\alpha_{12}$ are prefixes of $\sim\alpha_{31}$, they are eliminated).

4) β-sequence:

β_{11} r/- b/1 a/1 β_{21} r/- a/1b/1 a/1 β_{31} r/- a/1a/0a/0 a/1

β_{12} r/- a/1 a/0 a/0 β_{23} r/- a/1a/0 b/0 β_{32} r/- a/1a/0b/0 a/0a/0

The input portion of β-sequence: rbarabaraaaaraabaa

where the underlined are the transitions to be verified.

(Since β_{12} is a prefix of β_{31}, β_{23} is a prefix of β_{32}, they are eliminated).

5) The checking sequence (input portion):

raabrbrabraaaarbarabaraaaaraabaa

Its length is 32 and its ending state is s_1.

3.1.9 Discussion

In this section, we are going to determine whether the above methods (Method 1, 2, 3 and 4) produce checking sequence.

Theorem 3 Let a strongly-connected digraph G = (V, E) represent a deterministic, minimal, and completely-specified FSM M which has a distinguishing sequence. If P is a path of G that is generated by Method_1, then the label(P), IO-sequence Q, starting at any vertex v_x is a checking sequence of M.

Proof: For every edge e=(a, b; x/y) of G, we are going to prove that e is verified in Q.

From the construction of the β-sequence in Method_1, it can be seen that for each edge (a, b; x/y) of G, there is a subpath $(v_s,v_1;\ r/\text{-})@(v_1,v_j;\ TR(s_1,s_j))@(v_j,v_k;\ x/y)@(v_k,v_t;\ D_k)$ in P. From the construction of the α-sequence in Method_1, it can be seen that for any vertex v_j of G, there is a subpath $(v_p,v_1;\ r/\text{-})@(v_1,v_j;\ TR(s_1,s_j))@(v_j,v_q;\ D_j)$ in P, where v_j is d-recognized in Q as state a of M. Since Method_1 assumes the existence of a reliable reset feature, for each $(v_p,v_1;\ r/\text{-})$, v_1 is recognized in Q as the initial state of M. Therefore, in the subpath $(v_s,v_1;\ r/\text{-})@(v_1,v_j;\ TR(s_1,s_j))@(v_j,v_k;\ x/y)@(v_k,v_t;\ D_k)$ for (a, b; x/y), v_j is t-recognized and v_k is d-recognized in Q as states a and b of M. So (a,b; x/y) is verified. From Theorem 1, Q is a checking sequence of M. ■

Theorem 4 Let a strongly-connected digraph G = (V, E) represent a deterministic, minimal, and completely-specified FSM M which has a characterization set. If P is a path of G that is generated by Method_2, then the label(P), IO-sequence Q, starting at any vertex v_x is a checking sequence of M.

Proof: For every edge e=(a ,b; x/y) of G, we are going to prove that e is verified in Q.

From the construction of the β-sequence in Method_2, it can be seen that for each edge (a, b; x/y) of G, there are m subpaths spe_i =(v_s,v_1; r/-)@(v_1,v_j; TR(s_1,s_j))@(v_j,v_k; x/y)@(v_k,v_{ti}; W_{ki}) (i=1, ..., m) in P. From the construction of the α-sequence in Method_2, it can be seen that for any vertex v_j of G, there are m subpaths spv_i=(v_p,v_1; r/-)@(v_1,v_j; TR(s_1,s_j))@(v_j,v_{qi}; W_{ji}) (i=1, ...,m) in P by which v_j is w-recognized in Q as state a of M. Since Method_2 assumes the existence of a reliable reset feature, for each (v_p, v_1; r/-), v_1 is recognized in Q as the initial state of M. Therefore, in each subpath spe_i (i=1,...,m) for (a, b; x/y), v_j is t-recognized and v_k is w-recognized in Q as states a and b of M. So (a,b; x/y) is verified. From Theorem 2, Q is a checking sequence of M. ■

Theorem 5 Let a strongly-connected digraph G = (V, E) represent a deterministic, minimal, and completely-specified FSM M which has a characterization set or a UIO sequence for each state. If P is a path of G that is generated by Method_3 or Method_4, then the label(P), IO-sequence Q, starting at any vertex v_x is a checking sequence of M.

Proof: The proof for Method_3 (Wp method) is similar to the proof of Theorem 4, and Method_4 (UIOv method) can be seen as a special case of Wp method [FUJ91]. ■

3.2 FSM_1 without a Reliable Reset Feature

Up to this point, we have mainly discussed the checking sequence generation methods which assume the the presence of a correctly implemented reset feature, which brings the implementation, as well as the specification, from any state back into the initial state. However, as already pointed out by some researchers ([FUJ91], for instance), the reliable reset feature may in some cases be difficult to realize and therefore the above methods cannot be applied.

In the following subsections, we will extend to the above four methods for the case where FSM_i doesn't have a reliable reset feature.

3.2.1 Constructing an MLCS Using a Distinguishing Sequence

Suppose that the FSM_i doesn't have a reliable reset feature, i.e., the reset feature is not considered reliable. Assume that the FSM_s is minimal, strongly connected, completely specified and has at least one distinguishing sequence. D-method can be extended to construct a minimal length checking sequence as follows:

Method_5:

1) For the given FSM_s, compute a distinguishing sequence and denote it as D.

2) For the FSM_s, compute a synchronizing sequence (SS). If there is no SS, compute a homing sequence (HS).

3) Construct an α-sequence (state identification):

a) Construct the P-set:

(1) Construct $G_d=(V,E_d)$ from $G=(V,E)$, where $E_d=\{(k,\delta(k,D);D/\lambda(k,D)) : k=1,...,n\}$.

(2) Let the current P-set be empty and r = 1.

(3) While there is an edge of E_d that is not contained in any component of the current P-set = $\{P_1, ..., P_{r-1}\}$, do the following:

i) If there is an edge e=(i, j; D/λ(i,D)) not contained in any component of the current P-set and indegree(head(e))=0, then let P_r =e. Otherwise, choose an arbitrary e=(i, j; D/λ(i,D)) not contained in any component of the current P-set, and let P_r =e.

ii) If the next edge (tail(e), δ(tail(e),D); D/λ(tail(e),D)) $\notin$ P_k, $\forall$ $P_k \in$ P-set, $1 \leq k \leq r$, let e=(tail(e), δ(tail(e),D); D/λ(tail(e),D)) and P_r=P_re, goto ii). Otherwise, let e=(tail(e), δ(tail(e),D); D/λ(tail(e),D)) and P_r=P_re, goto iii).

iii) Let P-set = P-set $\cup\{P_r\}$, r=r+1.

(4) The result is the P-set.

b) Form E_α = {(head (P_k), tail(P_k); label(P_k))|$\forall$ $P_k \in$ P-set} where label(P_k)=α_k

c) Construct G'(V', E'), where V'= V, E'= E $\cup$ E_α

The problem is thus reduced to the problem of finding an RPP starting at v_1 and terminating at v_e over E_α in G'.

d) find

+ A minimum-cost symmetric augmentation G* of G[E_α] of G' by adding some edges from G to G[E_α]

+ An Euler path of G* starting at state v_1 which is the one reached after applying the SS or HS (followed possibly by the input portion of a transfer sequence) and terminating at some state v_e.

The label of this Euler path of G* is an α-sequence for G (or M).

4) Construct a β-sequence (transition verification):

a) Form $E\beta$ = {(s_j, δ(s_k, D); i/o@D_k) | e_{jk}=(s_j , s_k; i/o) in FSM_s}

b) Construct G"(V", E"), where V"=V, E"= $E\beta \cup$ E

The problem is thus reduced to the problem of finding an RPP starting at v_e and terminating at any vertex over E_β in G".

c) find

+ A minimum-cost symmetric augmentation G* of G[E_β] of G" by adding some edges from G to G[E_β].

+ An Euler path of G* starting at state v_e which is the tail of the Euler path found in 3.d and terminating at any vertex.

Such an Euler path of G* can be found by using the following procedure:

```
Mark all edges from Eβ in G* as UNTRAVERSED
For each vertex vi ∈ G*, define a set Ei that contains all the
        UNTRAVERSED outgoing edges of vi
let P*= nil and i=e (since the β-sequence starts at ve)
while not all edges from Eβ in G* are TRAVERSED do
        If not all edges in Ei are TRAVERSED
                THEN choose one UNTRAVERSED edge cik from Ei.
                        mark it TRAVERSED
                        append it to P*
                ELSE choose an edge tik from E in G* starting from vi
                        append it to P*
        let i=k
end_while
```

The resulting P* is the Euler path of G* we are looking for.

The label of this Euler path of G* is a β-sequence for G (or M).

5) The minimum-length checking sequence for FSM_s is:

α-sequence @ β-sequence

3.2.2 An Example Using Method_5

We apply Method_5 to the FSM_S M in Figure 1:

1) A DS of the FSM_S is ab.

2) An SS of the FSM_S is bb. The synchronizing tree of the FSM_S is shown in Figure 3.

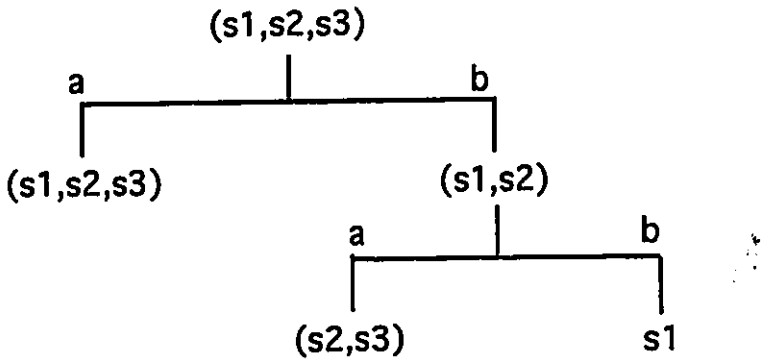


Figure 3 Synchronizing Tree of the FSM_S in Figure 1

G_d and P-set are shown in Figure 4 and Figure 5, respectively.

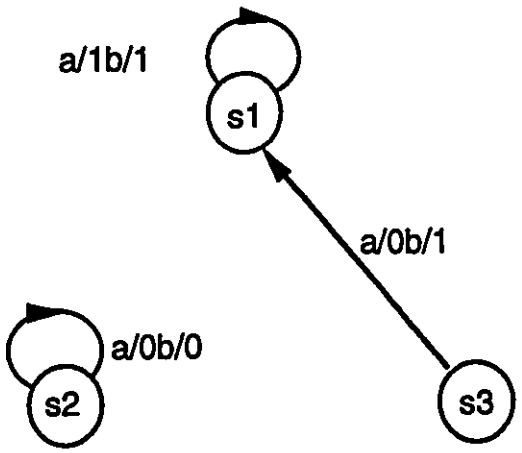


Figure 4 G_d for the FSM_S in Figure 1

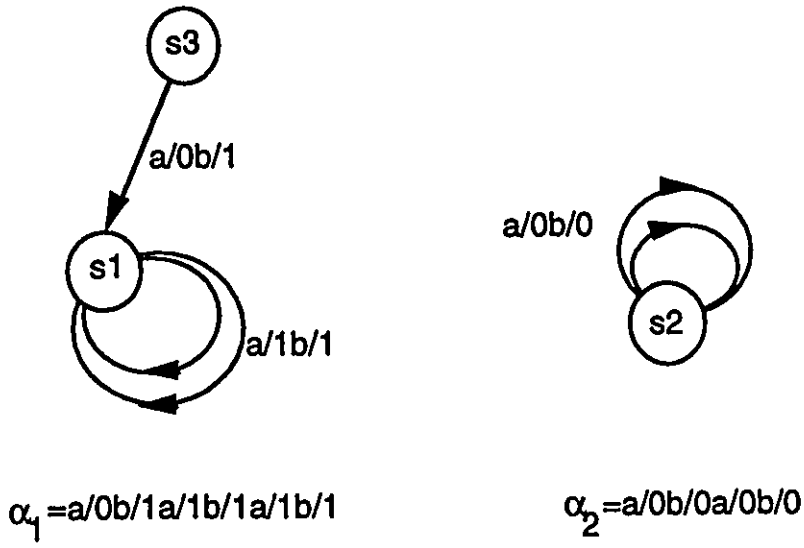


Figure 5 P-set for the FSM_s in Figure 1

3) α-sequence:

G* of G[E_α] is shown in Figure 6.

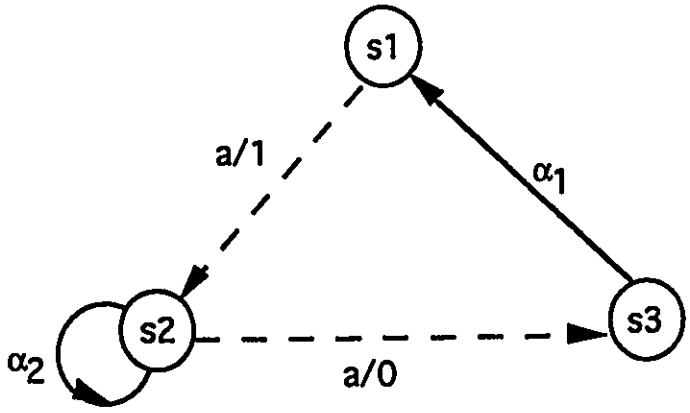


Figure 6 G* of G[E_α] for Method_5

where the edges from G added to G[E_α] are shown as dashed lines.

An Euler Path of G* of G[E_α] starting at state s_1 is

a/1 a/0b/0 a/0b/0 a/0 a/0b/1 a/1b/1 a/1b/1

$TR(s_1,s_2)$ D_2 D_2 $TR(s_2,s_3)$ D_3 D_1 D_1

The input portion of α-sequence: aababaababab

4) β-sequence:

G* of G[Eβ] is shown in Figure 7.

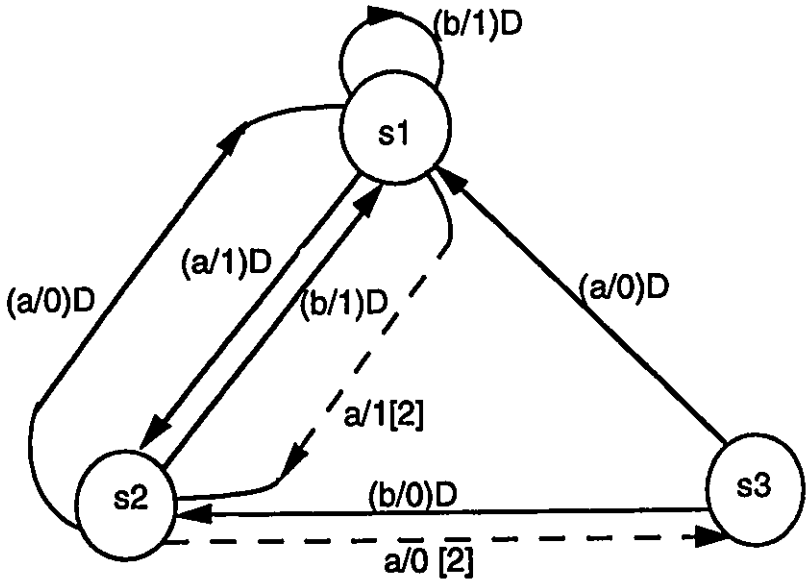


Figure 7 G* of G[Eβ] for Method_5

where the edges indicated by plain lines belong to Eβ and the edges represented by dashed lines belong to E, and the numbers in [] are the number of copies.

An Euler path of G* of G[Eβ] starting at s_1 is

a/1a/0b/0b/1 a/1b/1b/1 a/1b/1a/1a/0 a/0b/1a/1a/0b/0 a/0 b/0 a/0a/0a/1b/1

where the transitions to be verified are underlined and transfer sequences are represented by bold letters.

The input portion of β-sequence: aabbabbabaaabaababaaab

5) The checking sequence (input portion):

aababaababababaabbabbabaaabaababaaab

Its length is 34 and its ending state is s_1.

3.2.3 Constructing an MLCS Using an Expanded W-set

Suppose that the FSM_i doesn't have a reliable reset feature, i.e., the reset feature is not considered reliable. Assume that the FSM_s is minimal, strongly connected, and completely specified. Expanded W-set method can be extended to construct a minimal length checking sequence as follows:

Method_6:

1)For the given FSM_s, compute an expanded W-set and denote it as $W=\{w_1,w_2,...,w_m\}$, where the length of $w_i \leq$ length of w_j, if $i < j$.

2) For the FSM_s, compute a synchronizing sequence (SS). If there is no SS, compute a homing sequence (HS).

3) Construct a locating sequence for s_k, $\forall s_k \in S$

a) $\forall s_k \in S$, form $W_{k1},W_{k2},...,W_{km}$, which are sequences of labels obtained by applying $w_1,w_2,...,w_m$ to s_k,

b) $\forall s_k \in S$, choose Y_{ki} to be some sequence of input/output pairs that has W_{ki} as its prefix and that takes the correctly-operating FSM from state s_k back to state s_k, $1 \leq i \leq m$.

c) $\forall s_k \in S$, form:

$\gamma_1 = Y_{k1}$

$\gamma_2 = (\gamma_1)^{n+1} Y_{k2}$

$\gamma_3 = (\gamma_2)^{n+1}(\gamma_1)^{n+1} Y_{k3}$

...

$\gamma_m = (\gamma_{m-1})^{n+1}(\gamma_{m-2})^{n+1}...(\gamma_1)^{n+1} W_{km}$

where the n is the number of states in FSM_S.

The locating sequence for state s_k is γ_m which is denoted by LO_k. The input portion of LO_k is denoted by I_k. Locating sequence LO_k enables us to "locate" a state in FSM M that behaves like s_k with respect to m characterizing sequences.

4) Construct an α-sequence (state identification):

a) Form $E_{\alpha 00} = \{(s_k, \delta(s_k, I_k); LO_k) | \forall s_k \in S\}$

b) G(V,E) is augmented to form G' by adding all edges of $E_{\alpha 00}$ to G, i.e., construct G'(V',E'), V'=V, $E' = E \cup E_{\alpha 00}$.

The problem is thus reduced to the problem of finding an RPP starting at v_1 and terminating at v_e over $E_{\alpha 00}$ in G'.

c) find

+ A minimum-cost symmetric augmentation G* of $G[E_{\alpha 00}]$ of G' by adding some edges from G to $G[E_{\alpha 00}]$

+ An Euler path of G* starting at state s_1 which is the one reached after applying an SS or a HS (followed possibly by the input portion of a transfer sequence) and terminating at some state s_e.

The label of this Euler path of G* is denoted as α_{00} which is the first part of the α-sequence for G (or M).

d) Choose the shortest locating sequence and denote it by LO_S.

Let s_p=head(s, δ(s, I_S); LO_S), s_q=tail(s, δ(s,I_S); LO_S) and find a minimum spanning tree (MST) rooted at s_q.

e)$\forall s_k \in S$, form a path $P_{ki} = (v_p, v_q; LO_S)@(v_q,v_k;TR(s_q, s_k))@(v_k,\delta(s_k,w_i); W_{ki})@(\delta(s_k, w_i),v_p;TR(\delta(s_k,w_i), s_p))$ and denote label(P_{ki}) by $\alpha_{ki} = LO_S@TR(s_q, s_k)@W_{ki}@TR(\delta(s_k,w_i), s_p)$ (i=1,...,m), where $TR(s_q,s_k)$ is the label of a path from s_q to s_k in the MST.

f) Eliminate P_{ij} whose label is a prefix of the label of some $P_{i'j'}$, $1 \le i' \le n$, $1 \le j' \le m$.

Concatenate $TR(s_e,s_p)$ and all remaining P_{ij} to form P_α, where s_e is the terminating state of α_{00}.

g) Label(P_α) is the α-sequence.

5) Construct a β-sequence: (transition verification)

a) $\forall t_{jk} = (s_j, s_k; i/o) \in FSM_s$, form a path $P_{jki} = (v_p, v_q; LO_S) @ (v_q, v_j; TR(s_q, s_j)) @ (v_j, v_k; i/o) @ (v_k, \delta(s_k,w_i); W_{ki}) @ (\delta(s_k, w_i), v_p; TR(\delta(s_k,w_i), s_p))$ and denote label(P_{jki}) by $\beta_{jki}=LO_S@TR(s_q, s_j)@label(t_{jk})@W_{ki}@TR(\delta(s_k, w_i), s_p))$, (i=1,..,m), where $TR(s_q,s_j)$ is the label of the path from s_q to s_j in the MST.

b) Eliminate P_{jki} whose label is a prefix of the label of some $P_{j'k'i'}$, $1 \le i' \le m$, $1 \le j', k' \le n$. Concatenate all remaining P_{jki} to form P_β.

c) Label(P_β) is the β-sequence.

6) The minimum-length checking sequence for FSM_S is:

Label($P=P_\alpha @ P_\beta$) = α-sequence @ β-sequence ■

3.2.4 An Example Using Method_6

We apply Method_6 to the FSM_S M in Figure 1:

1) A W-set of the FSM_S is w_1=a, w_2=b, W={w_1,w_2}.

2) An SS of the FSM_S is bb.

3) Locating sequences:

LO_1: a/1

LO_2: a/0b/0a/0b/0a/0b/0b/1

LO_3: a/0a/1a/0a/0a/1a/0a/0a/1a/0b/0

Since state 1 is the only state that responds to input a by producing 1, it is not necessary to include b in LO_1. State 2 and state 3 respond to input a by producing an output of 0, so, a/0b/0 in LO_2 and a/0a/1a/0 in LO_3 need only be repeated 3 times.

4) α-sequence:

G* of $G[E_{\alpha 00}]$ is shown in Figure 8.

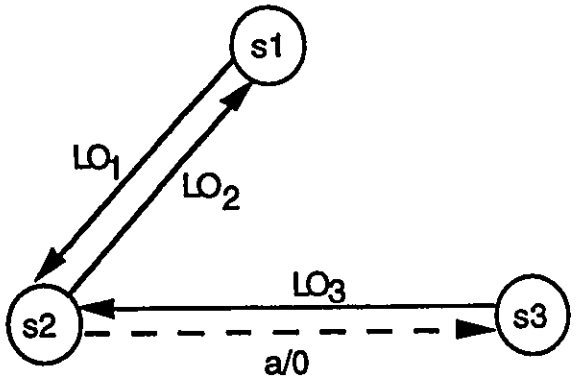


Figure 8 G* of G[$E_{\alpha 00}$] for Method_6

An Euler path of G* of G[$E_{\alpha 00}$] starting at s_1 and terminating at s_1 is:

a/1 a/0 a/0a/1a/0a/0a/1a/0a/0a/1a/0b/0 a/0b/0a/0b/0a/0b/0b/1

LO_1 $T(s_2,s_3)$ LO_3 LO_2

α_{00}: a/1a/0a/0a/1a/0a/0a/1a/0a/0a/1a/0b/0a/0b/0a/0b/0a/0b/0b/1

α_{11} LO_1 b/1a/1b/1 α_{12} LO_1 b/1b/1

α_{21} LO_1a/0a/0 α_{22} LO_1b/1

α_{31} LO_1 a/0 a/0 α_{32} LO_1a/0 b/0b/1

The input portion of α-sequence: aaaaaaaaaaababababbababababbaaaaabb

(Since α_{22} is a prefix of α_{12}, α_{21} is a prefix of α_{31}, they are eliminated).

5) β-sequence:

β_{111}: LO_1 **b/1b/1a/1b/1**

β_{112}: LO_1 **b/1b/1b/1**

β_{121}: LO_1 **b/1a/1a/0a/0**

β_{122}: LO_1 **b/1**a/1**b/1**

β_{211}: LO_1 b/1**a/1b/1**

β_{212}: LO_1 b/1**b/1**

β_{231}: LO_1 a/0**a/0**

β_{232}: LO_1 a/0**b/0 b/1**

β_{311}: LO_1 **a/0** a/0 **a/1b/1**

β_{312}: LO_1 **a/0**a/0**b/1**

β_{321}: LO_1 **a/0**b/0**a/0a/0**

β_{322}: LO_1 **a/0**b/0**b/1**

where the underlined letters are the transitions to be verified, the bold letters are transfer sequences.

Since β_{231} is a prefix of β_{311}, β_{232} is a prefix of β_{322}, β_{211} is a prefix of β_{122}, β_{212} is a prefix of β_{112}, the sequences β_{231}, β_{232}, β_{211}, β_{212} are eliminated.

Concatenate the remaining subsequences as follows:

LO_1b/1b/1a/1b/1LO_1b/1b/1b/1LO_1b/1a/1a/0a/0LO_1b/1a/1b/1

LO_1a/0a/0a/1b/1LO_1a/0a/0b/1LO_1a/0b/0a/0a/0LO_1a/0b/0b/1

The input portion of β-sequence: abbababbbabaaaababaaaabaaabaabaaaabb

6) The checking sequence (input portion):

aaaaaaaaaaababababbababababbaaaaabbabbababbbabaaaababaaaabaaabaab

aaaabb

Its length is 69 and its ending state is s_1.

3.2.5 Constructing an MLCS Using a Partial W-set

Suppose that the FSM_i doesn't have a reliable reset feature, i.e., the reset feature is not considered reliable. Assume that the FSM_s is minimal, strongly connected, and completely specified. Partial W-set method can be extended to construct a minimal length checking sequence as follows:

Method_7:

1)For the given FSM_s, compute an expanded W-set, denoted as $W=\{w_1,w_2,...,w_m\}$, where the length of $w_i \leq$ length of w_j, if $i < j$.

2) For the FSMs, compute a synchronizing sequence (SS). If there is no SS, compute a homing sequence (HS).

3) Construct a locating sequence for s_k, $\forall s_k \in S$

a) $\forall s_k \in S$, form $W_{k1},W_{k2},...,W_{km}$, which are sequences of labels obtained by applying $w_1,w_2,...,w_m$ to s_k.

b) $\forall s_k \in S$, choose Y_{ki} to be some sequence of input/output pairs that has W_{ki} as its prefix and that takes the correctly-operating FSM from state s_k back to state s_k, $1 \leq i \leq m$.

c) $\forall s_k \in S$, form:

$\gamma_1 = Y_{k1}$

$\gamma_2 = (\gamma_1)^{n+1}Y_{k2}$

$\gamma_3 = (\gamma_2)^{n+1}(\gamma_1)^{n+1}Y_{k3}$

...

$\gamma_m = (\gamma_{m-1})^{n+1}(\gamma_{m-2})^{n+1}...(\gamma_1)^{n+1}W_{km}$

where the n is the number of states in FSM_s.

The locating sequence for state s_k is γ_m which is denoted by LO_k. The input portion of LO_k is denoted by I_k.

4) Construct an α-sequence (state identification)

a) Form $E_{\alpha 00} = \{(s_k, \delta(s_k,I_k); LO_k) | \forall s_k \in S\}$

b) G(V,E) is augmented to form G' by adding all edges of $E_{\alpha 00}$ to G, i.e., construct G'(V',E'), V'=V, $E'=E \cup E_{\alpha 00}$.

The problem is thus reduced to the problem of finding an RPP starting at v_1 and terminating at v_e over $E_{\alpha 00}$ in G'.

c) find

+ A minimum-cost symmetric augmentation G* of $G[E_{\alpha 00}]$ of G' by adding some edges from G to $G[E_{\alpha 00}]$

+ An Euler path of G* starting at state s_1 which is the one reached after applying an SS or a HS (followed possibly by the input portion of a transfer sequence) and terminating at some state s_e.

The label of this Euler path of G* is denoted as α_{00} which is the first part of the α-sequence for G (or M).

d) Choose the shortest locating sequence and denote it by LO_s.

Let s_p=head(s, δ(s, I_s); LO_s), s_q=tail(s, δ(s,I_s); LO_s) and find a minimum spanning tree (MST) rooted at s_q.

e)$\forall s_k \in S$, form a path $P_{ki} = (v_p,v_q;LO_S)@(v_q,v_k;TR(s_q,s_k))@(v_k,\delta(s_k,w_i);W_{ki})@(\delta(s_k,w_i),v_p;TR(\delta(s_k,w_i),s_p))$ and denote label(P_{ki}) by $\alpha_{ki} = LO_S@TR(s_q,s_k)@W_{ki}@TR(\delta(s_k,w_i),s_p)$ (i=1,...,m), where $TR(s_q,s_k)$ is the label of a path from s_q to s_k in the MST.

f) Eliminate P_{ij} whose label is a prefix of the label of some $P_{i'j'}$, $1 \le i' \le n$, $1 \le j' \le m$.

Concatenate $TR(s_e,s_p)$ and all remaining P_{ij} to form P_α, where s_e is the terminating state of α_{00}.

g) Label(P_α) is the α-sequence.

5) $\forall s_k \in V$, form an identification set W_k, which is a subset of W, and can distinguish the state s_k from all other states.

6) Construct a β-sequence: (transition verification)

a) $\forall t_{jk} = (s_j, s_k; i/o) \in FSM_s$, form a path $P_{jki} = (v_p, v_q; LO_S) @ (v_q, v_j; TR(s_q, s_j))@(v_j, v_k; i/o)@(v_k, \delta(s_k, w_i); W_{ki})@(\delta(s_k, w_i), v_p; TR(\delta(s_k, w_i), s_p))$ and denote label(P_{jki}) by $\beta_{jki} = LO_s@TR(s_q, s_j) @ label(t_{jk}) @ W_{ki} @ TR(\delta(s_k, w_i), s_p))$, $(1 \le i \le |W_k|)$, where W_{ki} is the sequence of labels obtained by applying $w_i \in W_k$ to state s_k, $TR(s_q,s_j)$ is the label of the path from s_q to s_j in the MST.

b) Eliminate P_{jki} whose label is a prefix of the label of some $P_{j'k'i'}$, $1 \le i' \le m$, $1 \le j', k' \le n$. Concatenate all remaining P_{jki} to form P_β.

c) Label(P_β) is the β-sequence.

7) The minimum-length checking sequence for FSM_s is:

Label($P=P_\alpha @ P_\beta$) = α-sequence @ β-sequence ■

3.2.6 An Example Using Method_7

We apply Method_7 to the FSM_s M in Figure 1:

1) A W-set of the FSM_s is w_1=a , w_2=b, W={w_1,w_2}

2) An SS of the FSM_s is bb.

3) Locating sequences:

They are the same locating sequences used in the application of Method_6.

LO_1: a/1

LO_2: a/0b/0a/0b/0a/0b/0b/1

LO_3: a/0a/1a/0a/0a/1a/0a/0a/1a/0b/0

4) α-sequence:

α-sequence is the same as that shown in Method_6.

The input portion of α-sequence: aaaaaaaaaaababababbababababbaaaaabb

5) Identification set for each state: W_1={a}, W_2={a,b}, W_3={b}

6) β-sequence:

β_{111}: LO_1**b/1**b/1a/**1b/1**

β_{121}: LO_1**b/1**a/1a/**0a/0**

β_{122}: LO_1**b/1**a/1**b/1**

β_{211}: LO_1b/1**a/1b/1**

β_{232}: LO_1a/0**b/0 b/1**

β_{311}: LO_1 **a/0** a/0 **a/1b/1**

β_{321}: LO_1**a/0**b/0**a/0a/0**

β_{322}: LO_1**a/0**b/0**b/1**

where the underlined letters are the transitions to be verified, the bold letters are transfer sequences.

Since β_{232} is a prefix of β_{322}, β_{211} is a prefix of β_{122}, the sequences β_{232} and β_{211} are eliminated.

Concatenate the remaining subsequences as follows:

LO_1b/1b/1a/1b/1LO_1b/1a/1a/0a/0LO_1b/1a/1b/1LO_1a/0a/0a/1b/1LO_1a/0b/0a/0a/0 LO_1a/0b/0b/1

The input portion of β-sequence: abbababaaaababaaaabaabaaaabb

7) The checking sequence (input portion):

aaaaaaaaaaababababbababababbaaaaabbabbababaaaababaaaabaabaaaabb

Its length is 61 and its ending state is s_1.

3.2.7 Constructing an MLCS Using UIO Sequences

Suppose that the FSM_i doesn't have a reliable reset feature, i.e., the reset feature is not considered reliable. Assume that the FSM_s is minimal, strongly connected, completely specified and has UIO sequence for each state. UIOv method can be extended to construct a minimal length checking sequence.

In fact, if the FSM_s has a UIO sequence for each state, the UIOv method is a special case of the Wp method. The Wp method would use the union of all UIO sequences as the W-set, and identification set for each state is the same as the input portion of UIO for that state.

Method_8:

1) For the given FSM_s, compute a minimal length UIO sequence U_k for each state s_k and form $U=\{U_1,U_2,...,U_n\}$, the input portion of each U_k can be denoted by u_k, $k=1,...,n$.

2) For the FSM_s, compute a synchronizing sequence (SS). If there is no SS, compute a homing sequence (HS).

3) Let $W = \{u_k \mid U_k \in U\}$ and $W_k = u_k$, $k=1, 2, ..., n$.

4) To construct the α-sequence and the β-sequence, apply Method_7 step 3), 4), 5) and 6) with above W and W_k sets.

5) The minimum-length checking sequence for FSM_s is:

$$\text{Label}(P=P_\alpha @ P_\beta) = \alpha\text{-sequence} \ @ \ \beta\text{-sequence}$$ ■

3.2.8 An Example Using Method_8

We apply Method_8 to the FSM_s M in Figure 1:

1) A minimal length UIO sequence for each state is:

U_1 = a/1

U_2 = a/0 a/0

U_3 = b/0

2) An SS of the FSM_s is bb.

3) A W-set of the FSM_s is W={aa,b} or W={b,aa}. In this example, we let w_1=b and w_2=aa. W_1=input portion of U_1=a, W_2=input portion of U_2=aa, W_3=input portion of U_3=b.

4) Locating sequences:

LO_1: b/1b/1b/1a/1a/0

LO_2: b/1a/1b/1a/1b/1a/1a/0a/0

LO_3: b/0

Since state 3 is the only state that responds to input b by producing 0, it is not necessary to include aa in LO_3. State 1 and state 2 respond to input b by producing an output of 1, so, b/1a/1 in LO_2 and b/1 in LO_1 need only to be repeated 3 times.

5) α-sequence:

G* of $G[E_{\alpha 00}]$ is shown in Figure 9.

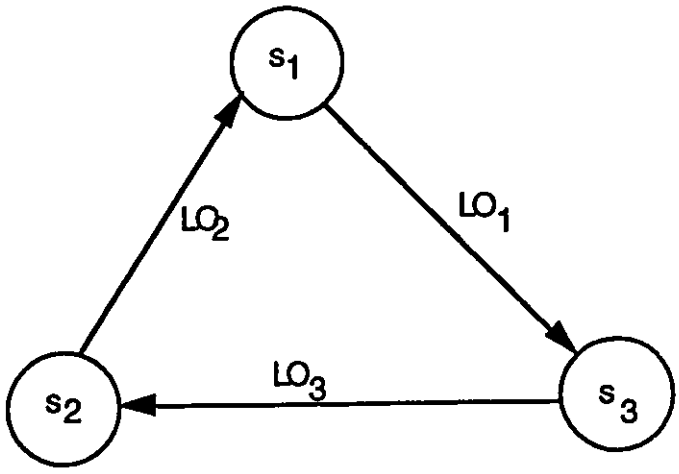


Figure 9 G* of G[$E_{\alpha 00}$] for Method_8

An Euler tour of G*of G[$E_{\alpha 00}$] starting at s_1 is

b/1b/1b/1a/1a/0 b/0 b/1a/1b/1a/1b/1a/1a/0a/0

LO_1 LO_3 LO_2

α_{00}: b/1b/1b/1a/1a/0b/0b/1a/1b/1a/1b/1a/1a/0a/0

Where the end state $s_e = s_1$ and $TR(s_e,s_p) = TR(s_1,s_3)$ =a/1a/0

α_{11} LO_3 b/1b/1a/1a/0 α_{12} LO_3b/1a/1a/0

α_{21} LO_3b/1a/1a/0 α_{22} LO_3a/0a/0a/1a/0

α_{31} LO_3a/0 b/0a/0 α_{32} LO_3a/0 a/0a/1a/0

The input portion of α-sequence: bbbaabbbabababaaaaabbbaabbbaabababababaaaa

(Since α_{22} is a prefix of α_{32}, α_{12} is a prefix of α_{21}, they are eliminated).

6) Identification set for each state: W_1={a}, W_2={aa}, W_3={b}

7) β-sequence:

β_{111}: LO_3**b/1**b/1**a/1a/0**

β_{122}: LO_3**b/1a/**1a/**0a/0a/1a/0**

β_{211}: LO_3b/1**a/1a/0**

β_{233}: LO_3a/0b/0 **a/0**

β_{311}: LO_3 **a/0** a/0 **a/1a/0**

β_{322}: LO_3**a/0**b/0**a/0a/0a/1a/0**

where the underlined letters are the transitions to be verified, the bold letters are transfer sequences.

Since β_{211} is a prefix of β_{122}, β_{233} is a prefix of β_{322}, sequences β_{211} and β_{233} are eliminated.

Concatenate the remaining subsequences as follows:

LO_3b/1b/1a/1a/0LO_3b/1a/1a/0a/0a/1a/0LO_3a/0a/0a/1a/0LO_3a/0b/0a/0a/0a/1a/0

The input portion of β-sequence: bbbaabbaaaaabaaaababaaaa

8) The checking sequence (input portion):

bbbaabbababaaaaabbbaabbaabababaaaabbbaabbaaaaabaaaababaaaa

Its length is 58 and its ending state is s_3.

3.2.9 Discussion

In this section, we are going to answer two questions related to the above checking sequence generation methods (Method 5, 6, 7 and 8). One is whether the above methods

produce checking sequences, another is whether each resulting checking sequence is of optimal length, or under which conditions the length of each resulting checking sequence is optimal.

The following theorems answer the first question:

Theorem 6 Let $G^\wedge = (V^\wedge, E^\wedge)$ constructed by adding (v_1,v_e; α-sequence) to G^* of $G[E_\beta]$ generated by Method_5 where v_e is the tail of the path whose label is α-sequence (denoted by L_α). Suppose that $P^\wedge$ is a path of $G^\wedge$ and contains every edge of $E_\alpha \cup E_\beta$ starting at vertex v_1, then the label($P^\wedge$), IO-sequence Q, starting at vertex v_1 and containing the L_α as its prefix is a checking sequence of M.

Proof: Suppose that $P^\wedge=(u_1, u_2; L_1)(u_2, u_3; L_2)...(u_{k-1}, u_k; L_{k-1})$, where u_1 is recognized as v_1. Then Q is the IO-sequence $L_1L_2 ... L_{k-1}$ starting at v_1, and $L_1 = L_\alpha$. Let $P=(a_1, a_2; L_1)(a_2, a_3; L_2)...(a_{k-1}, a_k; L_{k-1})$ be the corresponding path of G. We note that each (a_j, a_{j+1}; L_j) may be a path of G instead of a single edge of G.

We first show that a_i, i=1,2, ..., k-1, is recognized in Q. Suppose this is not true.

When i =1, u_i is v_1 and by hypothesis, a_i is recognized as s_1 of M, a contradiction.

When i =2, u_i is v_e, and from the construction of the α-sequence in Method_5, a_i is recognized in Q as some state a of M, a contradiction.

Suppose $u_1, u_2,..., u_{i-1}$ are the members of $V^\wedge$ such that $a_1, a_2,..., a_{i-1}$ are recognized in Q, (1< i <k), as some states of M.

We are going to prove that a_i is recognized in Q. Suppose this is not true.

If $(u_{i-1}, u_i; L_{i-1})$ is in E_β, then L_{i-1} has some D_j as suffix, from the construction of the α-sequence, a_i is recognized in Q as some state a of M, a contradiction.

If $(u_{i-1}, u_i; L_{i-1})$ is in E, then $(a_{i-1}, a_i; L_{i-1})$ is an edge of G where $L_{i-1}=x/y$ for some input x. From the construction of the β-sequence (step 4,c), it can be seen that P^ contains the edge $(u_j, u_{j+1}; L_j)$ which occurs before $(u_{i-1}, u_i; L_{i-1})$ and represents the test segment for the edge $(a_{i-1}, a_i; L_{i-1})$ where a_j is t-recognized in Q as state a of M, and $\delta(a_j, x)$ is d-recognized in Q as $\delta(a, x)$ of M. For $(a_j, \delta(a_j, x); x/y)$ and $(a_{i-1}, a_i; x/y)$, since a_j and a_{i-1} are t-recognized in Q as some state a of M and $\delta(a_j, x)$ is d-recognized in Q as some state $\delta(a, x)$ of M, based on the definition of recognition of a vertex, a_i is t-recognized in Q as $\delta(a, x)$ of M, a contradiction.

So a_i is recognized in Q, (i=1,2,...,k-1).

For every edge e=(a, b; x/y) of G, there is an edge $((a, (\delta(a, xD)); (xD)/\lambda(a, xD))$ of Eβ. Without loss of generality, we assume that this edge of Eβ is $(u_{i-1}, u_i; L_{i-1})$ in P^. Then, in Q, a_{i-1} is recognized as state a of M and $\delta(a_{i-1}, x)$ is recognized as state b of M. So (a, b; x/y) is verified. From Theorem 1, Q is a checking sequence of M. ■

Theorem 7 Let a strongly-connected digraph G = (V, E) represent a deterministic, minimal, and completely-specified FSM M with a W-set. If P is a path of G that is generated by Method_6, then the label(P), IO-sequence Q, starting at vertex v_1 is a checking sequence of M.

Proof: For every edge e=(a ,b; x/y) of G, we are going to prove that e is verified in Q.

From the construction of the β-sequence in Method_6, it can be seen that for each edge (a, b; x/y) of G, there are m subpaths $spe_i = (v_p, v_q; LO_s)@(v_q, v_j; TR(s_q, s_j))@(v_j, v_k; x/y)@(v_k, \delta(s_k, w_i); W_{ki})@(\delta(s_k, w_i), v_p; TR(\delta(s_k, w_i), s_p)) = (v_p, v_s; LO_s \backslash W_{sm})@(v_s, v_q; W_{sm})@(v_q, v_j; TR(s_q, s_j))@(v_j, v_k; x/y)@(v_k, \delta(s_k, w_i); W_{ki})@(\delta(s_k, w_i), v_p; TR(\delta(s_k, w_i), s_p))$ (i = 1, ..., m) in P, where v_p and v_q are the head and tail of the subpath representing the shortest locating sequence LO_s, v_s is LO_s-related state.

From the construction of the α-sequence in Method_6, it can be seen that for any vertex v_j of G, there are m subpaths $spv_i = (v_p, v_q; LO_s)@(v_q, v_j; TR(s_q, s_j))@(v_j, \delta(s_j, w_i); W_{ji})@(\delta(s_j, w_i), v_p; TR(\delta(s_j, w_i), s_p)) = (v_p, v_s; LO_s \backslash W_{sm})@(v_s, v_q; W_{sm})@(v_q, v_j; TR(s_q, s_j))@(v_j, \delta(s_j, w_i); W_{ji})@(\delta(s_j, w_i), v_p; TR(\delta(s_j, w_i), s_p))$ (i=1, ..., m) in P by which v_j is w-recognized in Q as state a of M. Then v_s in both spe_i and spv_i is *l*-recognized in Q as state s of M, therefore, in each subpath spe_i, i=1, ..., m, for (a, b; x/y), v_j is t-recognized and v_k is w-recognized in Q as states a and b of M. So (a,b; x/y) is verified. From Theorem 2, Q is a checking sequence of M. ■

Theorem 8 Let a strongly-connected digraph G = (V, E) represent a deterministic, minimal, and completely-specified FSM. If P is a path of G that is generated by

(1) Method 7, if the FSM has characterization set; or

(2) Method 8, if the FSM has unique input output sequence;

then the label(P), IO-sequence Q, starting at vertex v_1 is a checking sequence of M.

Proof: For every edge e=(a ,b; x/y) of G, we are going to prove that e is verified in Q.

From the construction of the β-sequence in Method_7 or Method_8, it can be seen that for each edge (a, b; x/y) of G, there are $|W_k|$ subpaths $spe_i = (v_p, v_q; LO_s)@(v_q, v_j; TR(s_q, s_j))@(v_j, v_k; x/y)@(v_k, \delta(s_k, w_i); W_{ki})@(\delta(s_k, w_i), v_p; TR(\delta(s_k, w_i), s_p)) = (v_p, v_s; LO_s \backslash W_{sm})@(v_s, v_q; W_{sm})@(v_q, v_j; TR(s_q, s_j))@(v_j, v_k; x/y)@(v_k, \delta(s_k, w_i); W_{ki})@(\delta(s_k, w_i), v_p; TR(\delta(s_k, w_i), s_p))$ ($1 \le i \le |W_k|$) in P, where v_p and v_q are the head and tail of the subpath representing the shortest locating sequence LO_s, v_s is LO_s-related state, and W_k is a subset of W which can distinguish the state s_k from all other states. W_{ki} is the sequence of labels obtained by applying $w_i \in W_k$ to state s_k. From the construction of the α-sequence in Method_7 or Method_8, it can be seen that for any vertex v_j of G, there are m subpaths $spv_i = (v_p, v_q; LO_s)@(v_q, v_j; TR(s_q, s_j))@(v_j, \delta(s_j, w_i); W_{ji})@(\delta(s_j, w_i), v_p; TR(\delta(s_j, w_i), s_p)) = (v_p, v_s; LO_s \backslash W_{sm})@(v_s, v_q; W_{sm})@(v_q, v_j; TR(s_q, s_j))@(v_j, \delta(s_j,$

w_i);W_{ji})@ ($\delta(s_j, w_i)$, v_p; TR($\delta(s_j, w_i)$, s_p)) (i=1, ..., m) in P by which v_j is w-recognized in Q as state a of M. Then v_s in both spe_i and spv_i is *l*-recognized in Q as state s of M, therefore, in each subpath spe_i, $1 \le i \le |W_k|$, for (a, b; x/y), v_j is t-recognized and v_k is w-recognized in Q as states a and b of M. So (a,b; x/y) is verified. From Theorem 2, Q is a checking sequence of M. ■

In both Method 7 and 8, locating sequences are used to recognize all the states in the FSM and to guarantee that the same state has been reached before different w_i is applied. The only difference is that in Method 7, the identification set is used to verify the tail state of each transition instead of the entire W-set whereas in Method 8, UIO sequence is used to verify the tail state of each transition. In order to guarantee the identification set or UIO sequence is "unique" in an implementation of M, during the α-sequence construction, all the states are identified by using the entire W-set or the union of all U_k(k=1,...,n).

We will analyze the solution of the second problem from two perspectives: α-sequence construction and β-sequence construction. In Method 6,7,8, since we don't use any optimization approaches to produce β-sequence, it is not relevant to discuss whether the resulting sequence is an optimal length sequence. For the α-sequence construction of Method 5, 6, 7 and 8, $G[E_\alpha]$ or $G[E_{\alpha 00}]$ of G' is not weakly connected, so, in most cases, the α-sequence is not an optimal length sequence. However, edges from E can be added to $G[E_\alpha]$ or $G[E_{\alpha 00}]$ to obtain G* of $G[E_\alpha]$ or $G[E_{\alpha 00}]$ such that G* is weakly connected, and an Euler path of the RSA G* of $G[E_\alpha]$ or $G[E_{\alpha 00}]$ is then a RPP of G over E_α or $E_{\alpha 00}$.

For the β-sequence construction of Method 5, we have the following theorem:

Theorem 9 Let a strongly-connected digraph G = (V,E) represent a deterministic, minimal, and completely-specified FSM M. If M has the reset or the self-loop feature then

the edge-induced subgraph G[Eβ] of the strongly connected digraph G" constructed in Method 5 step 4 is a spanning subgraph of G" and is weakly connected.

Proof: If M has the reset feature then there exists an edge $(v_i,v_1; L_i)$ in G from each vertex v_i, i=1,...,n to vertex v_1, representing the initial state s_1 in M. Suppose that $\delta(v_1, D)=v_k$, and $D/\lambda(v_1,D)=D_1$ then Eβ in G" contains the edges $(v_i, v_k; r/-@D_1)$, i=1,...,n, corresponding to the test segments for the reset edges $(v_i, v_1; L_i)$. This subset of edges of Eβ clearly induces a subgraph of G" which is weakly connected.

If M has the self-loop feature, for every $v_j \in V$, there is an edge $(v_j, \delta(v_j, D); L_{jj}D_j)$ in G[Eβ]. Thus the edge-induced subgraph G[Eβ] of G" is a spanning subgraph of G". Suppose that G[Eβ] is not weakly connected. Then there exists a subset $V"_s \subseteq V"$ such that $V"_s \cap (V" - V"_s) = \emptyset$ and there is no edge $(v_p, \delta(v_q, D); L_{pq}D_q) \in G[E\beta]$ such that (1) $v_p \in V"_s$ and $\delta(v_q, D) \in V" - V"_s$ or (2) $v_p \in V" - V"_s$ and $\delta(v_q, D) \in V"_s$. Since G is strongly connected, there exists an edge $(v_i,v_j; L_{ij}) \in E = E" - E\beta$ such that $v_i \in V"_s$ and $v_j \in V"-V"_s$. Let $\delta(v_j, D)=v_k$. If $v_k \in V"- V"_s$, this implies that there exists an edge $(v_i,\delta(v_j, D); L_{ij}D_j) \in G[E\beta]$, since $v_i \in V"_s$ and $v_k \in V" - V"_s$, it contradicts the assumption that there is no edge in G[Eβ] from $V"_s$ to $V"-V"_s$. If $v_k \in V"_s$, there would be an edge $(v_j,\delta(v_j, D); L_{jj}D_j)$ in G[Eβ] from $V"- V"_s$ to $V"_s$, it also produces a contradiction. Therefore, G[Eβ] of G" is weakly connected. ■

From theorem 9, it can been seen that if the FSM has the reset or the self-loop feature, the β-sequence constructed by using Method 5 step 4 is an optimal length sequence.

Method 5 is proposed for an FSM_i without reliable reset feature which implies either: (1) The FSM_s does not have the reset feature; or (2) The FSM_s does have reset transitions but they may not be implemented correctly in the FSM_i, so these reset transitions need to be verified as other transitions. In the first case, we can not find an

RCPT over E_β in G", but RPT over E_β in G" is achievable. In the second case, it meets the requirements in theorem 9, so there exists an RCPT over E_β in G".

Although we can use any of Method 5 to Method 8 to produce checking sequences without reliable reset feature, there are still a lot of redundancies in the resulting sequences. It can be seen that if we merge the construction of α-sequence and β-sequence, shorter checking sequences can be obtained and more efficient algorithms can be provided. In the next chapter, we will study the effects of interleaving α and β sequences.

4. THE EFFECTS OF INTERLEAVING OF α AND β SUBSEQUENCES

In this chapter, we will study the effects of interleaving of state identification subsequences (α-sequence) and transition verification subsequences (β-sequence) in two cases. In the first case, we will assume that the FSM_i has a reliable reset feature and in the other case, we will not make this assumption. For the first case, an algorithm which interleaves the α-sequence and the β-sequence generated by Method 1, 2, 3 and 4 in chapter 3 will be provided. For the second case two general models to construct shorter checking sequences by interleaving α-sequences and β-sequences will be proposed. For the first case, we will consider four methods: D-method, expanded W-method, Wp-method, UIOv-method and apply the proposed algorithm to the example in chapter 3. For the second case, we will consider the FSM_s with a distinguishing sequence and apply the proposed models to the example FSM in chapter 3.

4.1 The Effects of Interleaving for FSMs with Reliable Reset Feature

If an FSM_i has a reliable reset feature, we can use one of Method_1, ... , Method_4 given in Section 3.1 to construct a checking sequence. However, there may be additional redundancies (called *Cross - Redundancies*) in the resulting checking sequence other than those redundancies eliminated during the checking sequence construction in chapter 3. Suppose C is a checking sequence produced by employing one of Method_1, ... , Method_4 as a concatenation of an α-sequence and a β-sequence. Suppose A and B are two sets which contain all α-subsequences $\alpha_k, 1 \leq k$, and β-subsequences $\beta_j, 1 \leq j$, respectively. $\forall \alpha_k \in A$, $\forall \beta_j \in B$, if α_k is a prefix of β_j or β_j is a prefix of α_k, then we say there are *cross-redundancies* in C. For example, refer to α_1, α_2, α_3 appearing in β_{21}, β_{32}, β_{23}, respectively, in Case 1 of Section 4.1.2. In order to optimize the length of the

resulting checking sequence, we can interleave α-sequence and β-sequence by removing all cross-redundancies in the checking sequence.

4.1.1 An Algorithm for MLCS Construction

Theorem 10 Suppose A is a set which contains all α-subsequences produced by employing Method_1, ... , Method_4, $\forall\alpha \in A, \forall\alpha' \in A$, if α' is a prefix of α, then $\alpha = \alpha'$.

Proof: Suppose $\alpha \neq \alpha'$, so, the length of the α' is less than that of α or the length of α is less than that of α'. Since α' is a prefix of α, the length of α' is less than that of α. According to the step 3c in Method_1, ... , Method_4, α' should be removed from A, so $\alpha' \notin A$, a contradiction. So, $\alpha=\alpha'$. ■

Theorem 11 Suppose B is a set which contains all β-subsequences produced by employing Method_1, ... , Method_4, $\forall\beta \in B, \forall\beta' \in B$, if β' is a prefix of β, then $\beta = \beta'$.

Proof: Suppose $\beta \neq \beta'$, so, the length of the β' is less than that of β or the length of β is less than that of β'. Since β' is a prefix of β, the length of β' is less than that of β. According to the step 4b in Method_1, ... , Method_4, β' should be removed from B, so $\beta' \notin B$, a contradiction. So, $\beta=\beta'$. ■

Theorem 12 If $\alpha \in A$ is a prefix of $\beta \in B$, then there is not such an $\alpha' \in A$ that α' is a prefix of β and $\alpha \neq \alpha'$.

Proof: Suppose $\alpha' \in A$ is also a prefix of $\beta \in B$, the length of α is L, the length of α' is K and the length of β is P. So, $L \leq P$ and $K \leq P$.

Since α is a prefix of β, then the first L symbols of β are same as to that of α.

Since α' is a prefix of β, then the first K symbols of β are same as to that of α'.

If $L<K$, then α is a prefix of α', from theorem 10, it is a contradiction.

If $K<L$, then α' is a prefix of α, from theorem 10, it is a contradiction.

If $L=K$, then $\alpha=\alpha'$, from above assumption, it is a contradiction.

So, there is no such an $\alpha' \in A$ that α' is a prefix of β and $\alpha \neq \alpha'$. ■

Theorem 13 If $\beta \in B$ is a prefix of $\alpha \in A$, then there is not such a $\beta' \in B$ that β' is a prefix of α and $\beta \neq \beta'$.

Proof: Suppose $\beta' \in B$ is also a prefix of $\alpha \in A$, the length of β is L, the length of β' is K and the length of α is P. So, $L \leq P$ and $K \leq P$.

Since β is a prefix of α, then the first L symbols of α are same as to that of β.

Since β' is a prefix of α, then the first K symbols of α are same as to that of β'.

If $L<K$, then β is a prefix of β', from theorem 11, it is a contradiction.

If $K<L$, then β' is a prefix of β, from theorem 11, it is a contradiction.

If $L=K$, then $\beta=\beta'$, from above assumption, it is a contradiction.

So, there is no such an $\beta' \in B$ that β' is a prefix of α and $\beta \neq \beta'$. ■

Based on above four theorems, we give an algorithm in which set A and set B contain all subsequences of the α-sequence and β-sequence, respectively and set C is an empty set in the beginning.

Algorithm_1

Partition the set A and set B, according to the second input symbol (since first symbol is reset) of each sequence and denote them as $A_1, A_2, \ldots, A_q, B_1, B_2, \ldots, B_q$. q is the number of input symbols in FSM_s. The cardinality of each A_i (B_i) is denoted by t_i (l_i) (i=1,...,q), respectively.

i=0, C = nil

while i<q

r=1, j=1, i=i+1

while $j \leq l_i$

Case $\alpha_r \in A_i$ is a prefix of $\beta_j \in B_i$

put β_j in C, delete β_j from B_i, $l_i=l_i-1$

If $l_i = 0$

Then put $\alpha_{r+1}, \alpha_{r+2}, \ldots, \alpha_{ti}$ in C, j =1;

Else If $r < t_i$

Then r=r+1, j=1;

Else put rest of $\beta_j \in B_i$ in C, $j = l_i+1$;

Case $\beta_j \in B_i$ is a prefix of $\alpha_r \in A_i$

put α_r in C, delete β_j from B_i, $l_i=l_i-1$

If $l_i = 0$

Then put $\alpha_{r+1}, \alpha_{r+2}, ..., \alpha_{ti}$ in C, j =1;

Else If $r < t_i$

Then r=r+1, j=1;

Else put the rest of $\beta_j \in B_i$ in C, $j = l_i+1$;

Case $\alpha_r \in A_i$ is not a prefix of $\beta_j \in B_i$ and β_j is not a prefix of $\alpha_r \in A_i$

If $j < l_i$

Then j= j+1;

Else put α_r in C

If $r < t_i$

Then r=r+1, j=1;

Else put the rest of $\beta_j \in B_i$ in C, $j = l_i+1$;

end_while

end_while

Concatenate the elements of C, and the resulting sequence is the minimal length checking sequence. ■

Theorem 14 Suppose P_1 is the number of elements in set A, P_2 is the number of elements in set B, L is the length of the longest sequence in set A, q is the number of the input symbols in FSM_s. The average complexity of Algorithm_1 is:

$$P_1 + P_2 + P_1P_2(L-2)/q$$

and the upper bound of the complexity of Algorithm_1 is:

$$P_1 + P_2 + P_1(P_2-(q-1))(L-2)$$

Proof: Based on the algorithm, Set A and set B have to be partitioned into q subsets, so, the total time is: P_1+P_2. On the average, set A will be partitioned into q subsets, and each subset has P_1/q elements, for set B, there should be q subsets and each of them has P_2/q elements. Since, for each element in set A, we don't need to compare the first two symbols (first symbol is reset r/-, second symbol in two elements to be compared is identical), the length of elements to be compared in set A is at most (L-2), so, the average complexity of Algorithm_1 is:

$$P_1 + P_2 + (P_1/q)(P_2/q)(L-2)q = P_1 + P_2 + P_1P_2(L-2)/q$$

Since the FSM_s is completely specified, the set B must be partitioned into q subsets which has at least 1 element, so, in the worst case, set B has q-1 subsets which have 1 element and one subset which has $(P_2-(q-1))$ elements. For set A, the worst case is that set A only has one subset which has P_1 elements, so the upper bound of the complexity of Algorithm_1 is:

$$P_1 + P_2 + P_1(P_2-(q-1))(L-2)$$ ■

4.1.2 Applications of Algorithm_1

In this section, we will consider the FSM_s in Figure 1, and apply the α-sequences and β-sequences, which have been produced in 3.1.2, 3.1.4, 3.1.6, and 3.1.8, to Algorithm_1.

Case 1: The Interleaved MLCS Based on D-method

From section 3.1.2, we get the following α-sequence and β-sequence:

α-sequence:

α_1 r /- a/1 b/1

α_2 r/- a/1 a/0 b/0

α_3 r/- a/1 a/0 a/0 b/1

β-sequence:

β_{11} r/- b/1a/1b/1

β_{21} r/- a/1b/1a/1b/1

β_{23} r/- a/1a/0a/0b/1

β_{31} r/- a/1a/0a/0a/1b/1

β_{32} r/- a/1a/0b/0a/0b/0

where the underlined are the transitions to be verified.

By applying Algorithm_1, we can construct the following checking sequence (input portion):

rbabrababraaabraaaabraabab

(Since α_1,α_2 and α_3 are prefixes of β_{21},β_{32} and β_{23}, respectively, they are eliminated).

Its length is 26 and its ending state is s_2.

Suppose in FSM_s, n is the number of states, s is the depth of minimum spanning tree, d is the length of distinguishing sequence, from theorem 14, the average and worst-case complexity of Algorithm_1 for case 1 are:

Average: $n+nq+nnq(s+d-1)/q = n^2(s+d-1)+n(q+1)$

Worst-Case: $n+nq+n(nq-(q-1)(s+d-1) = n^2q(s+d-1)+n(1+q-(q-1)(s+d-1))$

Case 2: The Interleaved MLCS Based on Expanded W-method

From section 3.1.4, we get the following α-sequence and β-sequence:

α-sequence:

α_{12} r/- b/1

α_{22} r/- a/1 b/1

α_{31} r/- a/1 a/0 a/0

α_{32} r/- a/1 a/0 b/0

β-sequence:

β_{111} r/- b/1 a/1

β_{112} r/- b/1 b/1

β_{211} r/- a/1b/1 a/1

β_{212} r/- a/1b/1 b/1

β_{311} r/- a/1a/0 a/0 a/1

β_{312} r/- a/1a/0 a/0 b/1

β_{321} r/- a/1a/0 b/0 a/0

β_{322} r/- a/1a/0 b/0 b/1

where the underlined are the transitions to be verified.

By applying Algorithm_1, we can construct the following checking sequence (input portion):

rbarbbrabarabbraaaaraaabraabaraabb

(Since α_{12}, α_{22}, α_{31} and α_{32} are prefixes of β_{111}, β_{211}, β_{312} and β_{321}, respectively, they are eliminated).

Its length is 34 and its ending state is s_1.

Suppose in FSM_s, n is the number of states, s is the depth of minimum spanning tree, m is the number of elements in W-set, w is the length of longest characterizing sequence, from theorem 14, the average and worst-case complexity of Algorithm_1 for case 2 are:

Average: $nm+nmq+nmnmq(s+w-1)/q = n^2m^2(s+w-1)+nm(q+1)$

Worst-Case: $nm+nmq+nm(nmq-m(q-1))(s+w-1)$

$= n^2m^2q(s+w-1)-m^2n(q-1)(s+w-1)+nm(q+1)$

Case 3: The Interleaved MLCS Based on Wp-method

From section 3.1.6, we get the following α-sequence and β-sequence:

α-sequence:

α_{12} r/- b/1

α_{22} r/- a/1 b/1

α_{31} r/- a/1 a/0 a/0

α_{32} r/- a/1 a/0 b/0

β-sequence:

β_{111} r/- b/1 a/1

β_{211} r/- a/1b/1 a/1

β_{311} r/- a/1a/0 a/0 a/1

β_{321} r/- a/1a/0 b/0 a/0

β_{322} r/- a/1a/0 b/0 b/1

where the underlined are the transitions to be verified.

By applying Algorithm_1, we can construct the following checking sequence (input portion):

rbarabaraaaaraabaraabb

(Since α_{12}, α_{22}, α_{31} and α_{32} are prefixes of β_{111}, β_{211}, β_{311} and β_{321}, respectively, they are eliminated).

Its length is 22 and its ending state is s_1.

Suppose in FSM_s, n is the number of states, s is the depth of minimum spanning tree, m is the number of elements in W-set, m_i is the number of elements in characterization set W_i ($m_i \leq m$), w is the length of longest characterizing sequence, from theorem 14, the average and worst-case complexity of Algorithm_1 for case 3 are:

Average: $nm+q\sum_{i=1}^{n}m_i+nm(q\sum_{i=1}^{n}m_i)(s+w-1)/q = q\sum_{i=1}^{n}m_i+nm((\sum_{i=1}^{n}m_i)(s+w-1)+1)$

Worst-Case: $nm+q\sum_{i=1}^{n}m_i+nm((q\sum_{i=1}^{n}m_i)-(q-1))(s+w-1)$

$= q\sum_{i=1}^{n}m_i+nm(((q\sum_{i=1}^{n}m_i)-(q-1))(s+w-1)+1)$

Case 4: The Interleaved MLCS Based on UIOv-method

From section 3.1.8, we get the following α-sequence and β-sequence:

α-sequence:

$\sim\alpha_{11}$ r/- b/1

$\sim\alpha_{21}$ r/- a/1 b/1

α_3 r/- a/1 a/0 b/0

$\sim\alpha_{31}$ r/- a/1 a/0 a/0 a/1

β-sequence:

β_{11} r/- b/1 a/1

β_{21} r/- a/1b/1 a/1

β_{31} r/- a/1a/0a/0 a/1

β_{32} r/- a/1a/0b/0 a/0a/0

where the underlined are the transitions to be verified.

By applying Algorithm_1, we can construct the following checking sequence (input portion):

rbarabaraaaaraabaa

(Since $\sim\alpha_{11}$, $\sim\alpha_{21}$, α_3 and $\sim\alpha_{31}$ are prefixes of β_{11}, β_{21}, β_{32} and β_{31}, respectively, they are eliminated).

Its length is 18 and its ending state is s_1.

Suppose in FSM_S, n is the number of states, s is the depth of minimum spanning tree, u is the length of longest UIO sequence, from theorem 14, the average and worst-case complexity of Algorithm_1 for case 4 are:

Average: $n^2+nq+n^2nq\ (s+u-1)/q = n(n^2(s+u-1)+q+n)$

Worst-Case: $n^2+nq+n^2(nq-(q-1))(s+u-1)= n^2((s+u-1)(nq-q+1)+1)+nq$

From above applications, it can be seen that the length of the interleaved checking sequence is much shorter than that of uninterleaved checking sequence. The result of four cases is shown in Table 2.

Table 2 The Comparison of Two Kinds of Checking Sequences

Length / Cases	Un-interleaved Checking Sequence	Interleaved Checking Sequence
D-method	38	26
W-method	47	34
Wp-method	35	22
UIOv-method	32	18

4.2 The Effects of Interleaving for FSMs without Reliable Reset Feature

If an FSM_i doesn't have reliable reset feature, we can use one of Method_5, ... , Method_8 given in section 3.2 to construct a checking sequence. However, each of these methods either requires a considerable amount of time for the construction of checking sequence or yields a considerably long checking sequence.

In the following, two general models for constructing minimal length checking sequences by interleaving α and β sequences and utilizing distinguishing sequence are proposed.

4.2.1 Model M1 for Constructing MLCS by Using DS

Suppose that a strongly connected directed graph G = (V, E) representing a completely specified, minimal, and deterministic FSM M is given. Suppose that the FSM

M has a distinguishing sequence. Assume that a set of paths $\{P_1,...,P_r\}$ of G is given such that for each $P_k=(s_i, s_j; \alpha_k)$, $1 \le k \le r$, head(P_k) and tail(P_k) are recognized in α_k and the distinguishing sequence is a prefix and a suffix of α_k. Let α_k ($1 \le k \le r$) without the last distinguishing sequence be denoted as α'_k. The α-set (i.e. the set of labels $\alpha_1,...,\alpha_r$ of paths $P_1,...,P_r$ of G) satisfying the above condition is the one that can be obtained using Method_5 3a) and 3b).

We construct our model M1 as a new graph G'=(V', E') from G = (V, E) as follows.

Model M1

Let V'=V and $E'=E \cup E_\alpha \cup E_c \cup E''$, where

$E_\alpha=\{(\text{head}(P_k), \text{tail}(P_k); \alpha_k)$: for every P_k in G, $1 \le k \le r\}$

$E_c=\{(v_i, \delta(v_i, xD); x/y@D_j)$: for every $(v_i, v_j; x/y) \in E\}$

$E''=\{ (v_i, \delta(v_i,D); D_i)$: for every $v_i \in V$, (1) there is no self-loop edge $t_{ii}=(v_i, v_i; L_{ii})$ $(1 \le i \le n)$ in E such that $L_{ii} = x/y$ or (2) there is no edge $(v_i,v_i; L_{ii})$ such that $L_{ii} = \alpha'_i$, $(1 \le i \le r)\}$

By using above model, the construction of a checking sequence for the given FSM M has been reduced to the problem of finding an RCPT over $E_\alpha \cup E_c \cup E''$ in G' starting at v_1. The following steps can be used to solve this problem:

1) Construct an RSA $G^* = (V^*, E^*)$ of G' where $V^* = V'$ and E^* contains each edge in $E_\alpha \cup E_c \cup E''$ of G' and some edges in E.

Efficient algorithm exists for this step [AHO88].

2) Find an Euler tour starting at v_1 of G^* by using the following algorithm:

In G*, mark all edges from $E_\alpha \cup E_c \cup E''$ in G* as UNTRAVERSED

For each vertex $v_i \in G^*$, define a set E_i that contains all the UNTRAVERSED outgoing edges from v_i

let i=1 and P*=nil

while not all edges from $E_\alpha \cup E_c \cup E''$ in G* are TRAVERSED do

If not all edges in E_i are TRAVERSED

THEN choose one UNTRAVERSED edge e_{ik} from E_i

mark it TRAVERSED

append it to P*

ELSE choose an edge t_{ik} from E in G* starting from v_i.

append it to P*

let i=k

end_while

If v_i is not v_1, choose a $TR(v_i,v_1)$ from E in G* and append it to P*

The resulting P* is an Euler tour staring at vertex v_1 of G*. ■

Theorem 15 Suppose that P* is a path of G* that is generated by Model M1 and contains every edge of $E_\alpha \cup E_c \cup E''$ starting at vertex v_1. Then the label(P*), IO-sequence Q, starting at vertex v_1 is a checking sequence of M.

Proof: Suppose that $P^*=(u_1, u_2; L_1)(u_2, u_3; L_2)...(u_{k-1}, u_k; L_{k-1})$, where u_1 is recognized as v_1. Then Q is the IO-sequence $L_1L_2 ... L_{k-1}$ starting at v_1. Let $P=(a_1, a_2; L_1)(a_2, a_3; L_2)...(a_{k-1}, a_k; L_{k-1})$ be the corresponding path of G. We note that each $(a_j, a_{j+1}; L_j)$ may be a path of G instead of a single edge of G.

We first show that a_i, i=1,2, ..., k-1, is recognized in Q. Suppose this is not true. Since P* is generated by Model M1, P* is an ordered sequence of edges in which every edge (v_i, $\delta(v_j, D)$; x/y@D_j) from E_c has been traversed before the corresponding edge (v_i, v_j; x/y) from E is used as a transfer sequence.

When i=1, u_1 is v_1, so a_1 is recognized as state s_1 of M, a contradiction.

Suppose $u_1, u_2, \ldots, u_{i-1}$ are the members of V* such that $a_1, a_2, \ldots, a_{i-1}$ are recognized in Q, (1<i<k), as some states of M.

We are going to prove that a_i is recognized in Q. Suppose this is not true.

If (u_{i-1}, u_i; L_{i-1}) is in E_α, then from the choice of the α-set, a_i is recognized in Q as some state a of M, a contradiction.

If (u_{i-1}, u_i; L_{i-1}) is in E_c, then L_{i-1} has some D_j as suffix, from the choice of the α-set, every $\delta(v_j, D)$ is recognized, so a_i is recognized in Q as some state a of M, a contradiction.

If (u_{i-1}, u_i; L_{i-1}) is in E", then L_{i-1} is some D_{i-1}, and from the choice of the α-set, every $\delta(v_i, D)$ (i=1,..,n) is recognized, so a_i is recognized in Q as some state a of M, a contradiction.

If (u_{i-1}, u_i; L_{i-1}) is in E, then (a_{i-1}, a_i; L_{i-1}) is an edge of G where L_{i-1}=x/y for some input x. From the algorithm in Model M1 applied to form P*, P* contains an edge (u_j, u_{j+1}; L_j) which occurs before (u_{i-1}, u_i; L_{i-1}) and represents the test segment for the edge (a_{i-1}, a_i; L_{i-1}) where a_j is t-recognized in Q as state a of M, and $\delta(a_j, x)$ is d-recognized in Q as $\delta(a, x)$ of M. For (a_j, $\delta(a_j, x)$; x/y) and (a_{i-1}, a_i; x/y), since a_j and a_{i-1} are t-recognized in Q as some state a of M and $\delta(a_j, x)$ is d-recognized in Q as some state $\delta(a, x)$ of M, according to the definition of recognition of a vertex, a_i is t-recognized in Q as $\delta(a, x)$ of M, a contradiction.

So a_i is recognized in Q, (i=1,...,k-1).

For every edge e=(a, b; x/y) of G, there is an edge (a, δ(a, xD); (xD)/λ(a, xD)) of E_c. Without loss of generality, we assume that this edge of E_c is $(u_{i-1}, u_i; L_{i-1})$ in P*. Then, in Q, a_{i-1} is recognized as state a of M and $\delta(a_{i-1}, x)$ is recognized as state b of M. So (a, b; x/y) is verified. From Theorem 1, Q is a checking sequence of M. ■

We have established with Theorem 15 that the resulting sequence from the above model is a checking sequence. Below, we will prove that there exists an RCPT in G' over $E_\alpha \cup E_c \cup E''$, and the resulting checking sequence is an optimal sequence with respect to the Model M1.

Theorem 16 For any FSM M represented by G=(V,E) in Model M1, there exists an RCPT in G' over $E_\alpha \cup E_c \cup E''$, and the corresponding checking sequence is an optimal sequence with respect to the Model M1.

Proof: For every $v_j \in V'$, there is an edge $(v_j, v_k; x/y)$ in E, and an edge $(v_j, \delta(v_j, xD); x/y@D_k)$ in $G[E_\alpha \cup E_c \cup E'']$. Thus the edge-induced subgraph $G[E_\alpha \cup E_c \cup E'']$ of G' is a spanning subgraph of G'. Suppose that $G[E_\alpha \cup E_c \cup E'']$ is not weakly connected. Then there exists a subset $V'_s \subseteq V'$ such that $V'_s \cap (V' - V'_s) = \emptyset$ and there is no edge $(v_p, v_q; L_{pq}) \in G[E_\alpha \cup E_c \cup E'']$ such that (1)$v_p \in V'_s$ and $v_q \in V' - V'_s$ or (2) $v_p \in V' - V'_s$ and $v_q \in V'_s$. Since G is strongly connected, there exists an edge $(v_i, v_j; L_{ij}) \in E = E' - E_\alpha \cup E_c \cup E''$ such that $v_i \in V'_s$ and $v_j \in V' - V'_s$. Let $\delta(v_j, D) = v_k$. If $v_k \in V' - V'_s$, this implies that there exists an edge $(v_i, \delta(v_j, D); L_{ij}@D_j) \in G[E_c]$, since $v_i \in V'_s$ and $v_k \in V' - V'_s$, it contradicts the assumption that there is no edge in $G[E_\alpha \cup E_c \cup E'']$ from V'_s to $V' - V'_s$. If $v_k \in V'_s$, there would be three cases: (1) if there is a self-loop transition $t_{jj} = (v_j, v_j; x/y)$ at v_j, then there is an edge $(v_j, \delta(v_j, D); x/y@D_j)$ in $G[E_c]$ from $V' - V'_s$ to V'_s, (2) if there is no self-loop transition at v_j, but there is an edge $(v_j, v_j; \alpha'_j)$ at v_j, then there is an edge $(v_j, \delta(v_j, D); \alpha'_j@D_j)$ in $G[E_\alpha]$ from $V' - V'_s$ to V'_s, (3) if neither self-loop transition nor

α'_j exists for v_j, then there would be an edge $(v_j,\delta(v_j, D); D_j)$ in G[E"] from V'- V'_ε to V'_s. Above three cases all produce a contradiction. Therefore, $G[E_\alpha \cup E_c \cup E"]$ of G' is weakly connected. So, there exists an RCPT in G' over $E_\alpha \cup E_c \cup E"$. For a given set $E_\alpha \cup E_c \cup E"$ in an augmented graph G*, an RCPT is a minimum-length (minimum-cost) tour that contains each edge in $E_\alpha \cup E_c \cup E"$ at least once. Therefore, the RCPT forms an optimal length checking sequence with respect to the Model M1. ■

4.2.2 An Application of Model M1

Consider the FSM M and its graphical representation G given in Figure 1. A distinguishing sequence for the FSM M in Figure 1 is D=ab. The transition table of M is given in Table 3. The cost of each edge of G is 1.

Table 3 Transition Table of M and Responses to Distinguishing Sequence

States	(next state, output) for Input a	(next state, output) for Input b	(next state, output) for Input D = ab
v_1	v_2, 1	v_1, 1	v_1, 11
v_2	v_3, 0	v_1, 1	v_2, 00
v_3	v_1, 0	v_2, 0	v_1, 01

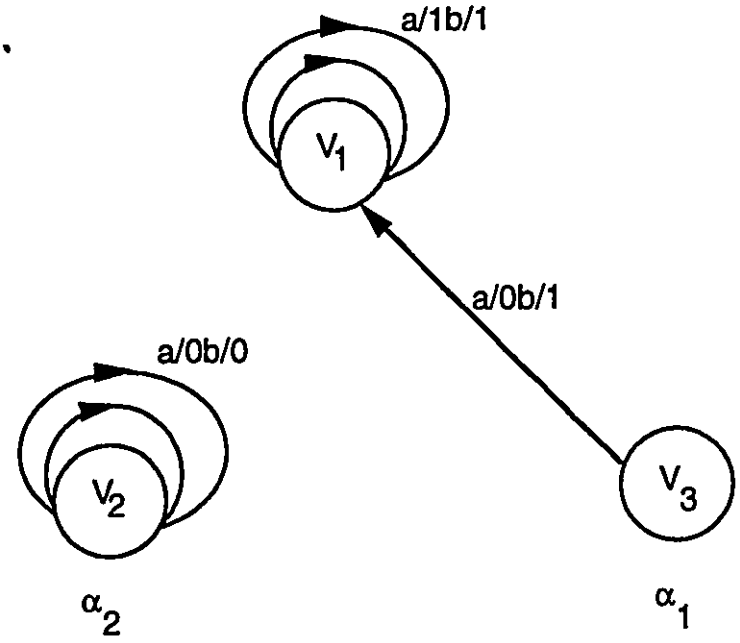


Figure 10. α-set for Model M1

The α-set obtained by using Method_5 3a) and 3b) is $\{\alpha_1, \alpha_2\}$, where α_1=(ababab)/(011111), α_2=(abab)/(0000), α'_1=(abab)/(0111), α'_2=(ab)/(00). This α-set is shown in Figure 10. Table 4, 5, 6 show all the edges in E_c, E_α and E", respectively.

Table 4 Edges in E_c

Starting State	Ending State	Label of the Edge
v_1	v_1	(bD)/(111)
v_1	v_2	(aD)/(100)
v_2	v_1	(aD)/(001)
v_2	v_1	(bD)/(111)
v_3	v_1	(aD)/(011)
v_3	v_2	(bD)/(000)

Table 5 Edges in E_α for Model M1

Starting State	Ending State for α'	Ending State for α	Label of the Edge
v_3	v_1	v_1	(ababab)/(011111)
v_2	v_2	v_2	(abab)/(0000)

Table 6 Edges in E" for Model M1

Starting State	Ending State	Label of the Edge
v_3	v_1	$D_3 = (ab)/(01)$

Since v_1 has a self-loop edge $(v_1,v_1; b/1)$ and v_2 has a self-loop edge $(v_2,v_2; \alpha'_2)$, E" only contains edge $(v_3,v_1;D_3)$. The only edges of E added to $E_\alpha \cup E_c \cup E"$ are $(v_2,v_3;$ a/0) and $(v_1,v_2;$ a/1) to form G* of $G[E_\alpha \cup E_c \cup E"]$. Each of these edges are replicated 4 times.

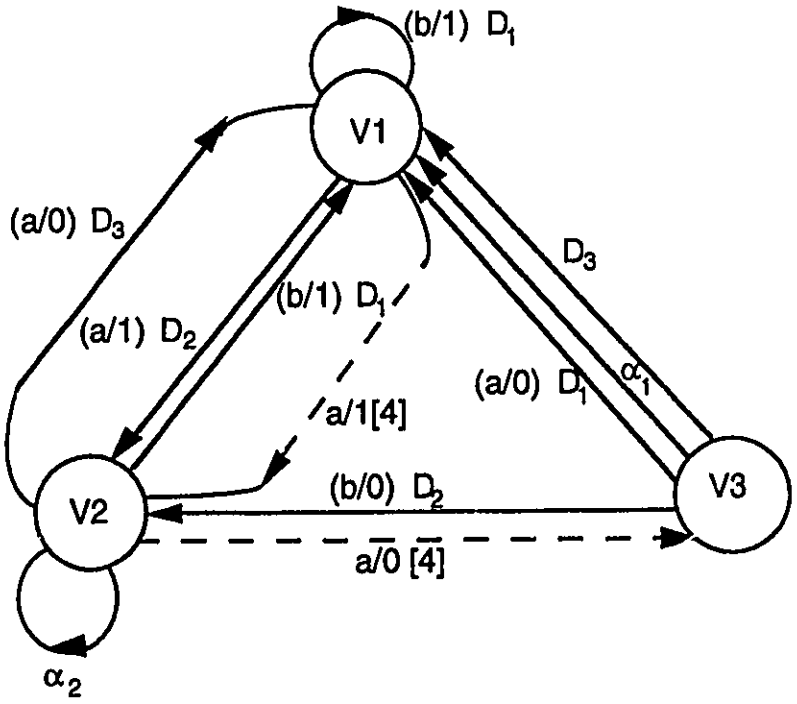


Figure 11. G* for Model M1

A path of G* (shown in Figure 11) which is an RCPT over $E_\alpha \cup E_c \cup E"$ starting at v_1 is as follows:

$(v_1, v_1; (bD)/(111))(v_1, v_2; (aD)/(100))(v_2, v_2; \alpha_2)(v_2, v_1;(bD)/(111))$

$(v_1, v_2;$ a/1$)(v_2, v_1; (aD)/(001))(v_1, v_2;$ a/1$)(v_2, v_3;$ a/0$)(v_3, v_1; \alpha_1)$

$(v_1, v_2; a/1)(v_2, v_3; a/0)(v_3, v_2; (bD)/(000))(v_2, v_3; a/0)(v_3, v_1; (aD)/(011))$

$(v_1, v_2; a/1)(v_2, v_3; a/0)(v_3, v_1; D/(01))$

A checking sequence of M which is the label of the above path of G* is

$((bD)/(111))((aD)/(100))\alpha_2((bD)/(111))(a/1)((aD)/(001))(a/1)(a/0)$

$\alpha_1(a/1)(a/0)((bD)/(000))(a/0)((aD)/(011))(a/1)(a/0)(D/(01))$.

where (i/o) indicates a transfer sequence.

The length of this sequence is 38.

In the next section, we will propose Model M2 which can produce a much shorter checking sequence than the one generated by Model M1.

4.2.3 Model M2 for Constructing MLCS by Using DS

Suppose that a strongly connected directed graph G = (V, E) representing a completely specified, minimal, and deterministic FSM M is given. Suppose that the FSM M has a distinguishing sequence. Assume that a set of paths $\{P_1,...,P_r\}$ of G is given such that for each $P_k=(s_i, s_j; \alpha_k)$, $1 \le k \le r$, head(P_k) and tail(P_k) are recognized in α_k and the distinguishing sequence is a prefix and a suffix of each α_k. The α-set (i.e. the set of labels $\alpha_1,...,\alpha_r$ of paths $P_1,...,P_r$ of G) satisfying the above condition is the one that can be obtained using Method_5 3a) and 3b). Let $\{T_1,...,T_q\}$, called T-set, be a set of labels of paths $R_1,...,R_q$ of G such that for each T_j, $1 \le j \le q$, tail(R_j) is recognized in some α_k and there is a distinguishing sequence as a prefix of each T_j.

We construct our model as a new graph G'=(V', E') from G = (V, E) as follows:

Model M2

For each vertex $v_i \in V$, we make a copy v'_i. Let $U'=\{v'_i: v_i \in V\}$ and $V'=V \cup U'$.

Let $E'=E \cup E_c \cup E_\alpha \cup E_T$, where

$E_c=\{(v_i, v'_j; x/y)$: for every $(v_i, v_j ; x/y) \in E\}$

$E_\alpha=\{((\text{head}(P_k))', \text{tail}(P_k); \alpha_k)$: for every $P_k, 1 \le k \le r\}$

$E_T=\{((\text{head}(R_j))', \text{tail}(R_j); T_j)$: for every $R_j, 1 \le j \le q\}$;

By using above model, we can produce the checking sequence for the given FSM M. The problem has thus been reduced to the problem of finding an RPT over $E_\alpha \cup E_c$ in G' starting at v_1. The following steps can be used to solve this problem:

1) Construct an RSA $G^*=(V^*, E^*)$ of G' where $V^*=V'$ and E^* contains each edge in $E_\alpha \cup E_c$ of G', some edges in E_T of G' and some edges in E.

Since it is possible that $G[E_\alpha \cup E_c]$ is not weakly connected, there may not exist an RCPT over $E_\alpha \cup E_c$ of G'. But we still can use some heuristic algorithm to construct an RSA $G^*=(V^*, E^*)$ of G'.

2) Find an Euler tour of G^* starting at v_1 by using the following algorithm:

mark all edges from $E_\alpha \cup E_c$ in G^* as UNTRAVERSED

for each vertex $v_i \in G^*$, define a set E_i that contains all the UNTRAVERSED outgoing edges of v_i

for each vertex $v'_i \in G^*$, define a set E'_i that contains all the UNTRAVERSED outgoing edges of v'_i

let i=1 and P*=nil

while not all edges from $E_\alpha \cup E_c$ are TRAVERSED do

If not all edges in E_i are TRAVERSED

THEN choose one UNTRAVERSED edge (v_i, v'_k; x/y) from E_i

mark it TRAVERSED

append it to P*

let i=k

If not all edges in E'_i are TRAVERSED

THEN choose one UNTRAVERSED edge (v'_i, v_j; α_i) from E'_i

mark it TRAVERSED

append it to P*

let i=j

ELSE choose one edge (v'_i, v_j; T_i) starting from v'_i

append it to P*

let i=j

ELSE choose an edge (v_i, v_k; x/y) in E from G* starting from v_i

append it to P*

let i=k

end_while

If v_i is not v_1, choose a TR(v_i,v_1) from E in G* and append it to P*

The resulting P* is an Euler tour of G* starting at v_1. ■

Theorem 17 Suppose that P* is a path of G* that is generated by Model M2 and contains every edge of $E_\alpha \cup E_c$ starting at vertex v_1. Then the label(P*), IO-sequence Q, starting at vertex v_1 is a checking sequence of M.

Proof: Suppose that P*=(u_1, u_2; L_1)(u_2, u_3; L_2)...(u_{k-1}, u_k; L_{k-1}), where u_1 is recognized as v_1. Then Q is the IO-sequence L_1L_2 ... L_{k-1} starting at v_1. Let P=(a_1, a_2; L_1)(a_2, a_3; L_2)...(a_{k-1}, a_k; L_{k-1}) be the corresponding path of G. We note that each (a_j, a_{j+1}; L_j) may be a path of G instead of a single edge of G.

We first show that a_i, i=1,2, ..., k-1, is recognized in Q. Suppose this is not true. Since P* is generated by Model M2, P* is an ordered sequence of edges in which every edge $(v_i, v'_j; x/y)$ from E_c has been followed by an α_j or T_j before the corresponding edge $(v_i, v_j; x/y)$ from E is used as a transfer sequence.

When i=1, u_1 is v_1, so a_1 is recognized as state s_1 of M, a contradiction.

Suppose $u_1, u_2, \ldots, u_{i-1}$ are the members of V* such that $a_1, a_2, \ldots, a_{i-1}$ are recognized in Q, (1<i<k), as some states of M.

We are going to prove that a_i is recognized in Q. Suppose this is not true.

If $(u_{i-1}, u_i; L_{i-1})$ is in E_T, then L_{i-1} is some T_j. From the choice of the T-set and the α-set, a_i is recognized in Q as some state a of M, a contradiction.

If $(u_{i-1}, u_i; L_{i-1})$ is in E_α, then from the choice of the α-set, a_i is recognized in Q as some state a of M, a contradiction.

If $(u_{i-1}, u_i; L_{i-1})$ is in E_c, then L_{i-1} is followed by some α_j or T_j, and thus a_i is recognized in Q as some state a of M, a contradiction.

If $(u_{i-1}, u_i; L_{i-1})$ is in E, then $(a_{i-1}, a_i; L_{i-1})$ is an edge of G where L_{i-1} =x/y for some input x. From the algorithm in Model M2 applied to form P*, P* contains an edge $(u_j, u_{j+1}; L_j)$ in G*, which occurs before $(u_{i-1}, u_i; L_{i-1})$ and has been followed by an α_i or a T_i and L_j = x/y, where a_j is t-recognized in Q as state a of M, and a_{j+1} is d-recognized in Q as $\delta(a, x)$ of M. For $(a_j, a_{j+1}; x/y)$ and $(a_{i-1}, a_i; x/y)$, since a_j and a_{i-1} are t-recognized in Q as some state a of M and a_{j+1} is d-recognized in Q as some state $\delta(a, x)$ of M, based on the definition of recognition of a vertex, a_i is t-recognized in Q as $\delta(a, x)$ of M, a contradiction.

So a_i is recognized in Q, (i=1,...,k-1).

For every edge e=(a, b; x/y) of G, there is an corresponding edge ((a, (δ(a, x))' ; x/λ(a, x)) of E_c, this edge is followed by a T_j or α_j. Without loss of generality, we assume that this edge of E_c is (u_{i-1}, u_i; L_{i-1}) in P*. Then, in Q, a_{i-1} is recognized as state a of M and $\delta(a_{i-1}, x)$ is recognized as state b of M. So (a, b; x/y) is verified. From Theorem 1, Q is a checking sequence of M. ■

4.2.4 Applications of Model M2

Consider the FSM M and its graphical representation G given in Figure 1. A distinguishing sequence for the FSM M in Figure 1 is D=ab. The transition table of M is given in Table 3. The cost of each edge of G is 1. By using this M, we will illustrate the applicability of the model to two different approaches taken by [GON78] and [HEN64], that utilize a distinguishing sequence for the construction of a checking sequence for a given FSM.

Case 1:

The α-set obtained by using Method_5 3a) and 3b) is $\{\alpha_1, \alpha_2\}$, where α_1=(ababab)/(011111), α_2=(abab)/(0000). The α-set for the example is shown in Figure 10 and Table 7.

A T-set that satisfies the property used in the definition of the T-set is $\{T_1, T_2, T_3\}$, where T_1=(ab)/(11), T_2=(ab)/(00), T_3=(ab)/(01). It can be seen in Table 8. The graph G* obtained according to the proposed model is shown in Figure 12.

The edges in G* from E can be seen in Table 9.

Table 7 Edges in E_α for Model M2 Case 1

Starting State	Ending State for α	Label of the Edge
v'_3	v_1	(ababab)/(011111)
v'_2	v_2	(abab)/(0000)

Table 8 Edges in E_T for Model M2 Case 1

Starting State	Ending State	Label of the Edge
v'_1	v_1	T_1= (ab)/(11)
v'_2	v_2	T_2= (ab)/(00)
v'_3	v_1	T_3= (ab)/(01)

Table 9 Edges in E for Model M2 Case 1

Starting State	Ending State	Label of the Edge
v_1	v_2	a/1
v_2	v_3	a/0

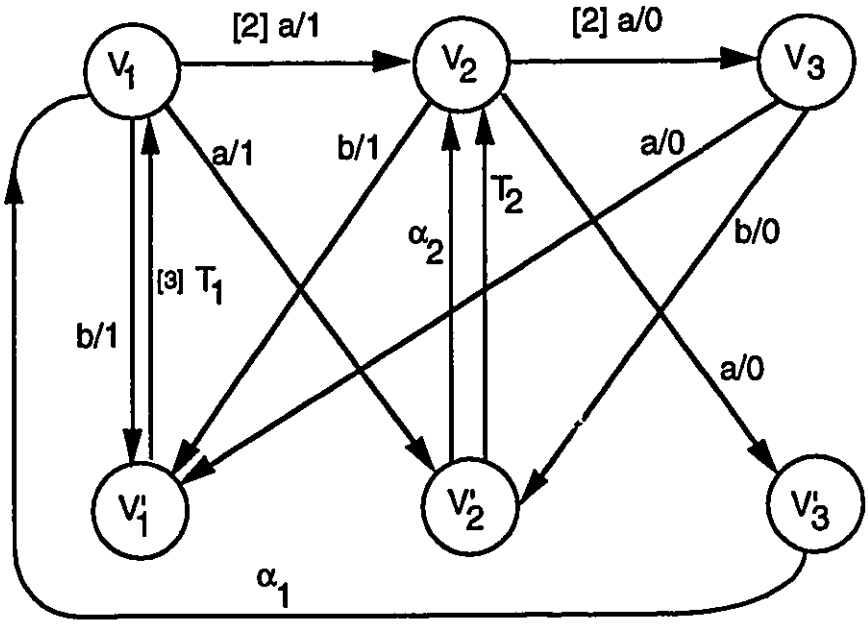


Figure 12. G* for Model M2 Case 1

A path of G* which is an RPT over $E_c \cup E_\alpha$ starting at v_1 is as follows:

$(v_1, v_2'; a/1)(v_2', v_2; \alpha_2)(v_2, v_3'; a/0)(v_3', v_1; \alpha_1)(v_1, v_1'; b/1)(v_1', v_1; T_1)$

$(v_1, v_2; a/1)(v_2, v_1'; b/1)(v_1', v_1; T_1)(v_1, v_2; a/1)(v_2, v_3; a/0)(v_3, v_2'; b/0)$

$(v_2', v_2; T_2)(v_2, v_3; a/0)(v_3, v_1'; a/0)(v_1', v_1; T_1)$

The checking sequence of M which is the label of the above path of G* is

$a/1\alpha_2$ $a/0\alpha_1$ b/1 T_1 (a/1) b/1 T_1 (a/1)(a/0) b/0 T_2 (a/0) a/0 T_1.

where (i/o) indicates transfer sequence.

The length of this sequence is 28.

Case 2:

Hennie gave a procedure in [HEN64] for constructing an I/O sequence in which every state of G is recognized. The format of this I/O sequence which starts at vertex v_1 and ends at vertex $\delta(v_1, D)$ is $(DI_1DI_2...DI_{n-1}\ DI_nD)/\lambda(v_1, DI_1DI_2...DI_{n-1}\ DI_nD)$ where each I_i is the input portion of a transfer sequence from $\delta(v_i, D)$ to v_{i+1}. Each $(DI_i)/\lambda(v_i, DI_i)$ for i=1,...,n, is used to recognize vertex v_i, and the last D is used to recognize vertex v_1 again. When this procedure is applied to the example FSM M given in Figure 1, the resulting I/O sequence (D/(11))(a/1)(D/(00))(a/0)(D/(01))(D/(11)) is shown in Figure 13. We take this I/O sequence as the only element of α-set ($\{\alpha_1\}$), α_1 = (D/(11))(a/1)(D/(00))(a/0)(D/(01))(D/(11)), and form the T-set as follows: $T_j = (DI_j)/\lambda(v_j, DI_j)$ for j=1,...,n. Thus, the T-set is T_1=(D/(11))(a/1), T_2=(D/(00))(a/0), T_3=(D/(01)). It can be seen from Table 10.

Table 10 Edges in E_T for Model M2 Case 2

Starting State	Ending State	Label of the Edge
v'_1	v_2	T_1= (aba)/(111)
v'_2	v_3	T_2= (aba)/(000)
v'_3	v_1	T_3= (ab)/(01)

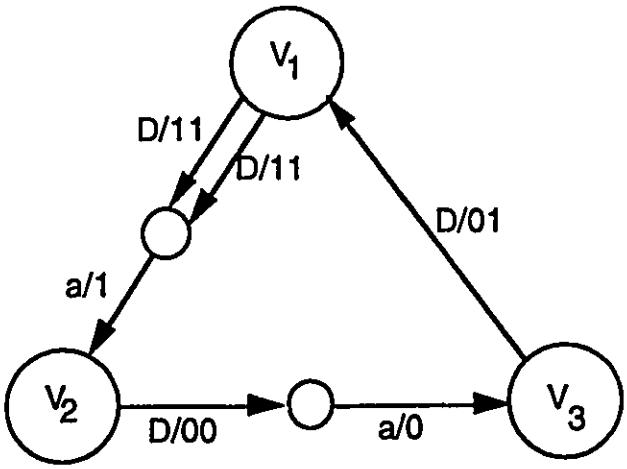


Figure 13. α-set for Model M2 Case 2

The graph G* obtained according to the proposed Model M2 is shown in Figure 14.

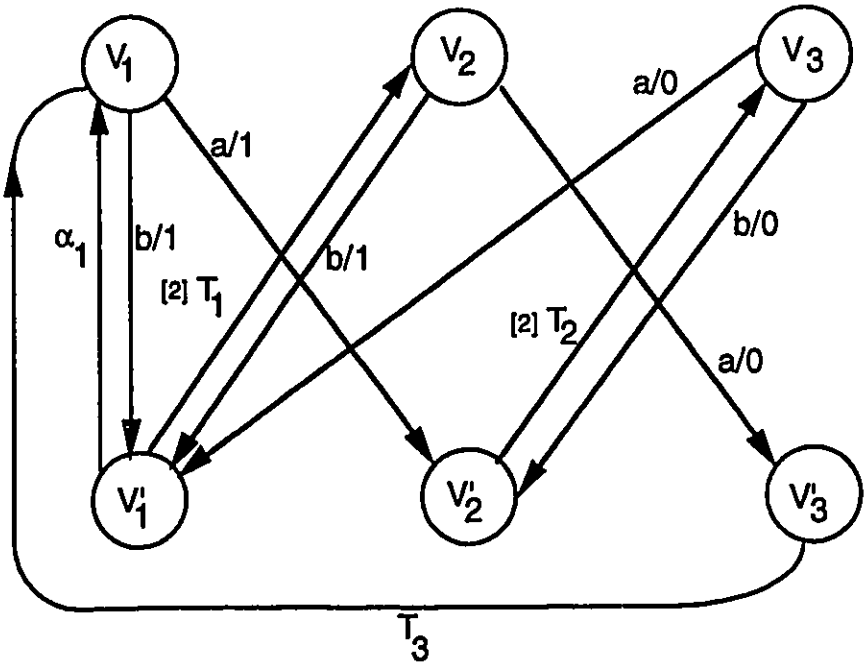


Figure 14. G* for Model M2 Case 2

A path of G* which is an RPT over $E_c \cup E_\alpha$ starting at v_1 is as follows:

$(v_1, v_1'; b/1)(v_1', v_1; \alpha_1)(v_1, v_2'; a/1)(v_2', v_3; T_2)(v_3, v_2'; b/0)(v_2', v_3; T_2)$

$(v_3, v_1'; a/0)(v_1', v_2; T_1)(v_2, v_1'; b/1)(v_1', v_2; T_1)(v_2, v_3'; a/0)(v_3', v_1; T_3)$

The checking sequence of M which is the label of the above path of G* is

$b/1\alpha_1$ $a/1T_2$ $b/0T_2$ $a/0\ T_1$ $b/1\ T_1$ $a/0\ T_3$.

The length of this sequence is 30.

4.2.5 Comparison of Model M1 and M2

From above results, it can be seen that the generation of the minimum length checking sequence has been explicitly formulated with both models. However, there exist tradeoffs between Model M1 and M2.

For Model M1, an RCPT can be formed and there are not any limitations for the FSM M, thus, an optimal length checking sequences can be produced in polynomial time. However, it has to be noted that the resulting checking sequence is a little bit longer than that generated by using Model M2.

For Model M2, there may not exist an RCPT even the FSM M has self-loop or reset feature, but an RPT can be found by adding some edges from E. The resulting checking sequence is much shorter than any other methods, compared with Method_5 and Model M1.

5. CONCLUSIONS

We have studied the problem of generation of the minimum length checking sequences for FSM-based protocol conformance testing. Under different assumptions for an FSM, the study focuses on finding the minimal length of checking sequences by using different state identification sequences, i.e. distinguishing sequence, W-set and UIO sequences.

Without interleaving state identification and transition verification sequences, four methods of generating minimum length test sequences based on D-method, W-method, Wp-method and UIOv-method all of which make the reliable reset feature assumption, have been reviewed and then based on these methods, four methods for generating minimum length checking sequences have been developed without making the reliable reset feature assumption.

It has been proved that the sequences generated by the above eight methods are checking sequences. For Method 5, it has been proven that if the FSM has the reset or the self-loop feature, the transition verification subsequences are of optimal length.

Moreover, the effects of interleaving the state identification and transition verification sequences on the length of the checking sequences have been studied. An algorithm for interleaving the state identification and transition verification sequences generated by the first four methods with reliable reset feature is proposed. It is observed that the reduction in the length of checking sequence due to interleaving is significant.

Finally, the two general models for constructing minimal length checking sequences using distinguishing sequences with interleaved state identification and transition verification sequences have been proposed. The proof for the first model to find a minimum length checking sequence in polynomial time has been established. The

sequences generated by using both models are proved to be checking sequences. Examples are provided as applications of both models. It is observed that, without reliable reset feature, the resulting checking sequences generated by using both models are significantly shorter than those using other methods without interleaving state identification and transition verification sequences.

The application area for the methods studied in this thesis covers lexical analysis, pattern matching, machine learning, telephony systems, and communication protocols. Within the context of communication protocols, the control portion of an implementation can be tested using checking sequences generated by the application of the studied methods. The complexity of generation of checking sequences and the length of these sequences increase as the size of the state space of the specification FSM increases.

It must be noted that both models are discussed only for distinguishing sequences, further research is needed to extend these models for a W-set. It is worth to study a general model for constructing a minimal checking sequence for a finite state machine without a distinguishing sequence. It is an interesting research problem to find whether there is a polynomial time algorithm to find the shortest checking sequence for a given finite-state machine that doesn't have a distinguishing sequence. Also, the study can be extended to the case where the number of states of the implementation is larger than that of the specification.

REFERENCES

- [AHO88] A.V. Aho, A.T. Dahbura, D. Lee, and M.U. Uyar, "An optimization technique for protocol conformance test sequence generation based on UIO sequences and rural Chinese postman tours," in *Protocol Specification, Testing, and Verification, XIII*, S. Aggarwal and K. Sabnani, eds., Amsterdam: North-Holland, pp.75-86, 1988.

- [BOC82] G.v. Bochmann and C.A. Sunshine, "A survey of formal methods," in *Computer Networks and Protocols*, P.E. Green, ed., New York: Plenum Press, pp.561-578, 1982.

- [BOY91] S.C. Boyd and H. Ural, "On the complexity of generating optimal test sequences," *IEEE Trans. on Software Engineering*, vol.SE-17, pp.976-978, Sep. 1991.

- [CHA89] W.Y.L. Chan, S.T. Vuong and M.R. Ito, "An improved protocol test generation procedure based on UIOs," in *Proc. ACM SIGCOMM'89 Symposium: Communication Architectures and Protocols*, pp.283-292, 1989.

- [CHO78] T. Chow, "Testing software design modeled by finite state machines," *IEEE Trans. on Software Engineering*, vol.SE-4, pp.178-187, March 1978.

- [EDM73] J. Edmonds and E.L. Johnson, "Matching, Euler tours and the Chinese postman," *Mathematical Programming*, vol.5, pp.88-124, 1973.

- [FUJ91] S. Fujiwara, G.v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, "Test selection based on finite state models," *IEEE Trans. on Software Engineering*, vol.SE-17, pp.591-603, June 1991.

- [GIL62] A. Gill, *Introduction to the Theory of Finite-State Machines*, New York: McGraw-Hill, 1962.

- [GON70] G. Gonenc, "A method for the design of fault detection experiments," *IEEE trans. on Computer*, vol. C-19, pp.551-558, June 1970.

- [HEN64] F.C. Hennie, "Fault detecting experiments for sequential circuits", *Proc. Fifth Ann. Symp. Switching Circuit Theory and Logical Design*, pp. 95-110, Princeton, N.J., 1964

- [KOH78] Z. Kohavi, *Switching and Finite Automata Theory*, New York: McGraw-Hill, 1978.

- [KUA62] M.K. Kuan, "Graphic programming using odd or even points," *Chinese Math.*, vol.1, pp.273-277, 1962.

- [LEN76] J.K. Lenstra and A.H.G. Rinnooy Kan, "On General Routing Problems," *Networks*, vol.6, pp.273-280, 1976.

- [NAI81] S. Naito and M. Tsunoyama, "Fault detection for sequential machines by transition tours," in *Proc. 11th IEEE Fault Tolerant Comput. Symp.*, IEEE Computer Soc. Press, pp.238-243, 1981.

- [RAY87] D. Rayner, "OSI conformance testing," *Computer Networks and ISDN Systems*, vol.14, no.1, pp.79-98, 1987.

- [SAB85] K.K. Sabnani and A.T. Dahbura, "A new technique for generating protocol tests," in *Proc. 9th Data Comm. Symp.*, pp.36-43, September 1985.

- [SAB88] K.K. Sabnani and A.T. Dahbura, "A protocol test generation procedure," *Computer Networks and ISDN Systems*, vol.15, no.4, pp.285-297, 1988.

- [SAR87] B. Sarikaya, G.v. Bochmann, and E. Cerny, "A test design methodology for protocol testing," *IEEE trans. on Software Engineering*, vol.SE-13, pp.518-531, May 1991.

- [SHE89] Y.N. Shen, F. Lombardi, and A.T. Dahbura, "Protocol conformance testing using multiple UIO sequences," in *Protocol Specification, Testing and Verification, IX*, E. Brinksma, G. Scollo, and C.A. Vissers, eds., Amsterdam: North-Holland, pp.131-143, 1989.

- [SID89] D.P. Sidhu and T.K. Leung, "Formal methods for protocol testing: a detailed study," *IEEE Trans. on Software Engineering*, vol.SE-15, no.4, pp.413-426, 1989.

- [SID90] D.P. Sidhu and A. Chung, "On sufficient conditions for an efficient protocol conformance test generation technique based on Rural Chinese Postman Problem," submmitted for publication, 1990.

- [URA87] H. Ural, "Test sequence selection based on static data flow analysis," *Computer Communications*, vol. 10, no.5, pp.234-242, 1987.

- [URA91a] H. Ural, "On the formal methods for test sequence generation," to appear in *Computer Communications*, 1991.

- [URA91b] H. Ural and B. Yang, "A test sequence selection method for protocol testing," *IEEE trans. on Software Engineering*, vol.SE-17, pp.514-523, April 1991.

- [UYA86] M.U. Uyar and A.T. Dahbura, "Optimal test sequence generation for protocols: the Chinese postman algorithm applied to Q.931," in *Proc. 1986 IEEE Global Telecommunications Conf.*, pp.68-72, 1986.

- [VUO89] S.T. Vuong, el al., "The UIOv-method for Protocol Test Sequence Generation" in *Proc. of the 2nd Int. Workshop on Protocol Test Systems, Berlin, Germany, Oct.3-6, 1989.*