SYNCHRONIZABLE TEST SEQUENCE GENERATION FOR PROTOCOL CONFORMANCE TESTING

by

Zhiping Wang

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.

ISBN 0-315-85796-X

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ABSTRACT

This study addresses the synchronization problem that arises during the application of a predetermined test sequence in some test architectures used for protocol conformance testing. A solution to this problem is to construct synchronizable test sequences. For FSM-based protocol specifications, a necessary and sufficient condition is given for the existence of a synchronizable test sequence for a given FSM. A polynomial time algorithm is then developed to determine the existence of such a sequence for a given FSM. A duplex digraph method is developed which transforms the order-specified digraph representing an FSM into a duplex digraph. Using the duplex digraph representation of a given FSM, a polynomial time algorithm is developed to generate a synchronizable test sequence for the FSM which checks only the output function of the FSM. The W-, D- and UIO-methods are extended to an FSM represented by an order-specified digraph by employing synchronizable W-sets, synchronizable DS sequences and synchronizable UIO sequences, respectively. Polynomial time algorithms are then developed to generate synchronizable test sequences for a given FSM which check the transfer function as well as the output function.

ACKNOWLEDGEMENTS

Sincere gratitude is extended to Professor Hasan Ural for his patience and guidance through the course of this study.

I especially wish to thank my wife Ilgar for her continuous encouragement.

I would like to acknowledge Telecommunication Research Institute of Ontario (TRIO) for its financial support and the School of Graduate Studies and Research of the University of Ottawa for the Entrance Scholarship.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The specification of a communication protocol is a set of rules which defines all possible interactions among the entities of a communication system. The implementation of a communication protocol must be tested for conformance to its specification. The objective of protocol conformance testing is to establish whether a protocol implementation under test (I) conforms to the protocol specification [24, 32].

1.1. A Test Architecture

The Open System Interconnection (OSI) model [23] developed by the International Organization for Standardization (ISO) defines a layered architecture for a communication system. A protocol implementation may correspond to a single layer or multiple layers in this model. Various test architectures have been proposed for testing the conformance of a protocol implementation as a single layer entity, as a single-layer in a multi-layer entity, or as a multi-layer entity, etc.. A common feature of these test architectures proposed for protocol conformance testing is that I is tested as a black box.

There are two main classes of such test architectures, local and external [24]. Local test architectures allow direct observation and control at the lower and upper interfaces of I. External test architectures, on the other hand, allow only indirect observation and control at the lower interface of I. The local test architectures are only applicable to in-house testing by suppliers, whereas the external test architectures are also applicable to testing carried out by users and third party test centres [24].

There are three types of external test architectures: distributed, coordinated and remote [24]. In this study, we will consider the distributed test architecture with an upper and a lower tester [24] shown in Figure 1. In the distributed test architecture, the lower interface and the upper interface of I are controlled and observed indirectly by the lower

tester (L) and directly by the upper tester (U), respectively. During testing, L and U coordinate in applying a usually pre-determined sequence of stimuli (*i.e.*, test sequence) to I and observing indirectly or directly the responses (output) of I to the applied stimuli (input).

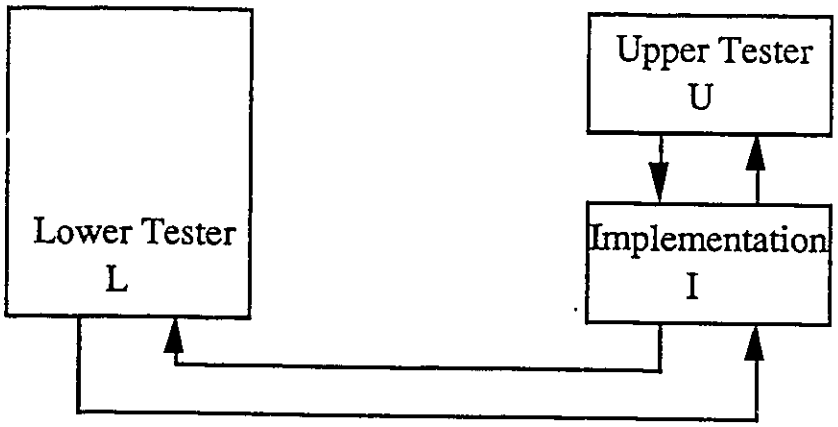


Figure 1. A Distributed Test Architecture

Due to the black box approach to protocol conformance testing, test sequences are generated from the protocol specifications. Since natural language definitions of protocols suffer from various deficiencies, such as ambiguity and difficulty with automated treatment, formal description techniques (FDTs) have been developed for the specification of protocols as well as for specification of services which include Estelle [7, 25] and Lotos [3, 22] developed by ISO, and SDL [9] developed by the International Telegraph and Telephone Consultative Committee (CCITT).

A protocol specification has two portions: control flow and data flow. The control flow is typically modelled as a finite state machine (FSM), while the data flow is typically modelled as program segments [2]. Various formal methods [1, 10, 11, 12, 15, 21, 24,

29, 30, 31, 33, 36, 37, 42, 43, 44] have been proposed for generating test sequences from a protocol specification of which the control flow is modelled as deterministic FSM [20]. The typical aim of these methods is to generate a test sequence which checks whether each transition specified in the FSM is correctly implemented by I, *i.e.*, for each transition, I generates the expected response and transfers to the expected state upon receiving the stimulus. In general, such a sequence starts and terminates at a specific state (*i.e.*, start state) of the given FSM.

Other methods have also been proposed to generate test sequences for testing the data flow aspect of the protocol implementation [35, 40] and for testing the control flow and data flow aspect of specification implementation [41].

1.2. Synchronization Problem

During the application of a predetermined test sequence in the test architecture shown in Figure 1, U and L are bound to synchronize with each other only through their interactions with I. However, this requirement may lead to a synchronization problem when L (or U) is expected to send an input to I after I responds with an output to U (or L) to an input from U (or L) but L (or U) is unable to determine whether I sent that output.

Synchronization between U and L can be achieved by any suitable test management protocol coordinating the actions of the testers [24], by a ferry protocol with inter-process communication [24], or through manual coordination (by use of a telephone or terminal connection) [32]. However, these solutions either require an additional communication channel between U and L or an additional protocol to facilitate communication between U and L. The necessity for such coordination is eliminated by constructing a synchronizable test sequence such that the corresponding sequence of transitions cause no synchronization problem.

1.3. Objectives

This study tackles the problem of generating a synchronizable test sequence for a protocol specification modelled as a deterministic and minimal finite state machine (FSM) represented by a strongly biconnected digraph. Our objectives are to

a) give a polynomial-time algorithm to determine the existence of a synchronizable test sequence for a given FSM;

b) develop a polynomial-time algorithm to generate a synchronizable test sequence to detect the output errors of a given protocol implementation;

c) extend the W-method, the D-method and the UIO-method to generate synchronizable test sequences which check the state transfer function as well as the output function of a given protocol implementation.

1.4. Organization of the Thesis

In chapter 2, we present definitions of terms used throughout the remainder of this thesis and summarize well-known methods to construct a (rural) postman tour (path) in a digraph. In chapter 3, we review the FSM-based formal methods for test sequence generation. In chapter 4, we give a necessary and sufficient condition for the existence of a synchronizable test sequence for a given FSM. Based on this condition, in chapter 5, we develop a polynomial-time algorithm to determine the existence of such a test sequence. In chapter 6, we give a polynomial-time algorithm for generating a synchronizable test sequence which checks only the output function of a given FSM. In chapter 7, by extending the W-, D-, and UIO-methods, we develop algorithms for generating synchronizable test sequences which check output and transfer functions of a given FSM.

2. PRELIMINARIES

2.1. The Finite State Machine Model

All methods discussed in this thesis assume a Mealy machine model for the control portion of a protocol specification. A Mealy machine is an FSM characterized by a quintuple [26] : (**S**, **I**, **O**, δ, λ), where

S is a set of states $\{s_0, s_1, ..., s_m\}$;

I is a set of input symbols $\{i_1, i_2, ..., i_q\}$;

O is a set of output symbols $\{o_1, o_2, ..., o_r\}$;

δ is a mapping **S** x **I** -> **S**, called *state transition function*;

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

A state transition from s_j to s_k in the FSM is denoted by t_{jk} =(s_j,s_k; i/o) where i is an input symbol, o an output symbol, and i/o is the label of the transition. The s_0 is the start state of the FSM. The reset input "ri" $\in$ **I** takes the FSM from any state to state s_0 with a single transition producing a "null" output $\in$ **O**.

An FSM is said to be *deterministic* if for each input symbol i $\in$ **I** there is at most one transition defined at each state of the FSM.

An FSM is said to be *minimal* if it contains no equivalent states [20].

An FSM is said to be *completely specified* if for each input symbol i $\in$ **I**, there is a transition defined at each state of the FSM.

2.2. Transition-level Testing Approach

A transition-level testing approach originating from sequential circuit checking experiments [26] is adopted by the FSM-based formal test sequence generation methods. Accordingly, the procedure of checking whether a transition $t_{jk}=(s_j,s_k; i/o)$ of the FSM is correctly implemented by the implementation I consists of the following steps.

1) the implementation I is brought to state s_j;

2) input i is applied to the implementation I and the output of the implementation I is checked whether it is o;

3) the state the implementation I reaches after the application of i is checked whether it is s_k.

Step 2) detects the errors in the output function. Whereas step 3) detects the errors in the state transfer function. The input sequence, called test subsequence, required to test the transition $t_{jk}=(s_j,s_k; i/o)$ will thus consists of

a) an optional transfer sequence, *i.e.*, the shortest sequence of inputs which bring the implementation I from its current state to state s_j;

b) input symbol i for stimulating transition t_{jk};

c) a checking sequence, *i.e.*, a sequence of inputs that checks whether the implementation I reaches s_k.

A *test sequence* for an FSM is then a sequence of inputs that checks whether each transition of the FSM is correctly implemented by a given implementation of the FSM.

2.3. Synchronizable Test Sequence

Formally, the synchronization problem is defined as follows:

Let each transition t_{ij} (from state i to state j) of I be one of the following types, *i.e.*, **type**(t_{ij}) =

R^{IL} : I receives an input from L and does not send any output (*i.e.*, output is null)

R^{IU} : I receives an input from U and does not send any output (*i.e.*, output is null)

$R^{IL}S^{IL}$: I receives an input from L and sends an output to L

$R^{IL}S^{IU}$: I receives an input from L and sends an output to U

$R^{IU}S^{IL}$: I receives an input from U and sends an output to L

$R^{IU}S^{IU}$: I receives an input from U and sends an output to U

$R^{IL}S^{IL,U}$: I receives an input from L and sends an output to L and U

$R^{IU}S^{IU,L}$: I receives an input from U and sends an output to U and L

Then, considering two consecutive transitions of I, one of the testers, say L (or U), faces a *synchronization problem* if L (or U) did not take part in the first transition and if the second transition requires that it sends a stimulus to I [34]. For example, a synchronization problem will occur if $R^{IL}S^{IL}$ is to be followed by R^{IU}.

Two consecutive transitions t_{ij} and t_{jk} of I form a *synchronizable pair of transitions* if t_{jk} can follow t_{ij} without generating a synchronization problem. For example, $R^{IL}S^{IU}$ forms a synchronizable pair of transitions when followed by any other transition of I. For a transition t_{ij} of an FSM, each transition t_{jk} that forms a synchronizable pair of transitions with t_{ij} is called an *eligible successor* of t_{ij}.

A transition sequence is *synchronizable* if for each two consecutive transitions of the sequence, the second transition is an eligible successor of the first one of the FSM.

A *synchronizable input sequence for state s* is an input sequence in response to which the FSM from state s produced a synchronizable transition sequence. A *synchronizable test sequence* for an FSM is then both a synchronizable input sequence for the start state of the FSM and a test sequence.

2.4. Graph Theoretic Definitions

We will represent an FSM by a digraph D = (V,E) where V is the set of vertices, each representing a state of the FSM and E is the set of directed edges, each representing a transition of the FSM. The *label* of an edge is the label of the corresponding transition. Given an edge e = (i,j) in a digraph, we call i the head of e and j the tail of e, and denote them by *head*(e) and *tail*(e), respectively. The *cost* of an edge is the number of labels associated with the edge. A *self-loop* is an edge starting and ending at the same vertex.

For each vertex v in a digraph D=(V,E), in-degree of v, $d_i(v)$, is defined as $|\{(u,v): (u,v) \in E\}|$ and out-degree of v, $d_o(v)$, is defined as $|\{(v,w): (v,w) \in E\}|$. A digraph D=(V,E) is *symmetric* if $d_i(v) = d_o(v)$, $\forall\ v \in V$.

A *path* P in a digraph D=(V,E) is a finite non-null sequence of (not necessarily distinct) consecutive edges: $P = (v_1, v_2), (v_2, v_3), ..., (v_{k-1}, v_k)$. In a digraph with no multiple edges between any pair of vertices, a path is determined by the sequence $v_1, v_2, ..., v_k$ of its vertices, and thus can be specified as $P = v_1, v_2, ..., v_k$. The *length* of a path is the number of edges included in the path. The *cost* of a path is the sum of costs of the edges in the path. We also define, for a path P = i, j, ..., p, q, *first_edge*(P) = (i,j) and *last_edge*(P) = (p,q). A path is *simple* if every edge in the path is distinct.

A digraph D=(V,E) is *strongly connected* if for any pair of distinct vertices v_i and v_j in V, there exists a path from v_i to v_j. A digraph D=(V,E) is *weakly connected* if the underlying undirected graph is connected [4]. The *edge-induced subgraph* $D[E_C]$ of

D=(V,E) for some subset $E_C \subseteq E$ is the subgraph of D whose vertex set is the set of heads and tails of edges in E_C and whose edge set is E_C [4]. $D[E_C]$ is an edge-induced *spanning* subgraph of D if its vertex set is V [1].

A *tour* in a digraph D=(V,E) is a closed path, *i.e.*, a path which starts and ends at the same vertex. An *Euler tour (path)* in a digraph D=(V,E) is a tour (path) which traverses every edge in E exactly once.

A strongly connected digraph D=(V,E) contains an Euler tour if and only if D satisfies the following symmetry condition:

Symmetry Condition I: $d_i(v) = d_o(v), \forall\ v \in V.$

A strongly connected digraph D=(V,E) contains an Euler path if and only if D satisfies the following symmetry condition:

Symmetry Condition II: $d_i(v) = d_o(v), \forall\ v \in V - \{i, j\},$

$d_i(j) = d_o(j) + 1,$ and

$d_i(i) = d_o(i) - 1,$

where i is the start vertex the j is the final vertex of the Euler path.

2.5. The Postman Tour/Path

A *postman tour* in a digraph D=(V,E) is a tour which includes each edge in E at least once. A *Chinese postman tour* in a digraph D=(V,E) is a minimum-cost postman tour of D. A *postman path* and a *Chinese postman path* in a digraph D=(V,E) are similarly defined.

The problem of finding a Chinese postman tour or path has been studied extensively [4, 17, 19, 27]. This is called the Chinese Postman Problem. The solution of

this problem is related to the the solution of the problem of finding an Euler tour or path in a graph.

2.5.1. The Postman Tour

If D=(V,E) satisfies the Symmetry Condition I, finding a Chinese postman tour is equivalent to finding an Euler tour, which can be solved by an O(V+E) algorithm [19]. If D is not symmetric, we can construct a minimum-cost symmetric augmentation D* of D [14] by replicating edges of D such that D* satisfies the Symmetry Condition I and the replicated edges are minimized. An Euler tour in D* is then a Chinese postman tour in D. The minimum-cost symmetric augmentation of D for a tour is achieved by minimizing $z = \Sigma N_e C_e$ subject to $N_e \geq 0$ and N_e is an integer, $\forall\ e \in E$, and $\Sigma(N_e: k = head(e)) - \Sigma(N_e: k = tail(e)) = d_i(k) - d_o(k),\ k \in V$, where C_e is the cost of edge e. The minimum-cost symmetric augmentation for a tour can be solved by a min-cost max-flow algorithm [1, 14, 17, 39] with a complexity of O(VElogV) [1].

2.5.2. The Postman Path

If D=(V,E) satisfies the Symmetry Condition II, finding a Chinese postman path from vertex i to vertex j in D is equivalent to finding an Euler path from i to j, which can be solved by an O(V+E) algorithm [19]. If D is not symmetric, we can construct a minimum-cost symmetric augmentation D* of D [14] by replicating edges of D such that D* satisfies the Symmetry Condition II and the replicated edges is minimized. An Euler path in D* is then a Chinese postman path in D. The minimum-cost symmetric augmentation of D for a Chinese postman path is achieved by minimizing $z = \Sigma N_e C_e$ subject to $N_e \geq 0$ and N_e is an integer, $\forall\ e \in E$, and

$$\Sigma(N_e: k = head(e)) - \Sigma(N_e: k = tail(e)) = d_i(k) - d_o(k),\ k \in V - \{i, j\},$$

$$\Sigma(N_e: j = head(e)) - \Sigma(N_e: j = tail(e)) = d_i(j) - d_o(j) - 1, \text{ and}$$

$$\Sigma(N_e: i = head(e)) - \Sigma(N_e: i = tail(e)) = d_i(j) - d_o(j) + 1.$$

The minimum-cost symmetric augmentation for a Chinese postman path can similarly be solved by a min-cost max-flow algorithm [1, 14, 17, 39] with a complexity of O(VElogV) [1].

2.6. The Rural Postman Tour/Path

A *rural postman tour* (RPT) of D=(V,E) over a set $E_C \subseteq E$ is a tour traversing every edge in E_C at least once. A *rural Chinese postman tour* (RCPT) of D=(V,E) over a set $E_C \subseteq E$ is a minimum-cost tour traversing every edge in E_C at least once. A *rural postman path* (RPP) of D=(V,E) over a set $E_C \subseteq E$ and a *rural Chinese postman path* (RCPP) of D=(V,E) over a set $E_C \subseteq E$ are similarly defined.

2.6.1. The Rural Postman Tour.

Computing an RCPT is NP-complete in the most general case [28]. Nevertheless, if D is strongly connected and the edge-induced spanning subgraph $D[E_C]$ is weakly connected, then any rural symmetric augmentation D* of $D[E_C]$ for a tour is strongly connected [1]. D*=(V*,E*) is a minimum-cost *rural symmetric augmentation* (RSA) of $D[E_C]$ for a tour if D* satisfies the Symmetry Condition I in section 2.4, $V^* \subseteq V$, and E* contains every edge in E_C at least once and every edge in $E-E_C$ zero or more times such that the total cost of edges in E* is minimized. From section 2.4, a minimum-cost RSA D* has an Euler tour which is an RCPT of D over E_C.

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

The complexity of obtaining an RPT or RCPT of a strongly connected digraph D=(V,E) is bounded by the complexity of computing an RSA D* of D[E_C] or an RSA D* of D[E_C*], which can be solved with the complexity of O(VElogV) [1].

2.6.2. The Rural Postman Path

It can be shown that if D is strongly connected and the edge-induced spanning subgraph D[E_C] is weakly connected, then any rural symmetric augmentation D* of D[E_C] for a path is strongly connected. D*=(V*,E*) is a minimum-cost *rural symmetric augmentation* (RSA) of D[E_C] for a path if D* satisfies the Symmetry Condition II in section 2.4, V* $\subseteq$ V, and E* contains every edge in E_C at least once and every edge in E-E_C zero or more times such that the total cost of edges in E* is minimized. A minimum-cost RSA D* of D[E_C] has an Euler path which is an RCPP of D over E_C.

If D[E_C] is not weakly connected, edges from E can be added to E_C to obtain E_C* such that D[E_C*] is weakly connected. An Euler path in the RSA D* of D[E_C*] is then an RPP of D over E_C, but not necessarily an RCPP.

The complexity of obtaining an RPP or an RCPP of a strongly connected digraph D=(V,E) is also bounded by the complexity of computing an RSA D* of D[E_C] or an RSA D* of D[E_C*], which can be solved with the complexity of O(VElogV) [1].

3. FSM-BASED TEST SEQUENCE GENERATION METHODS

There are four major formal testing methods for the FSM-based protocol testing, *i.e.*, the T-method, the D-method, the W-method and the UIO-method. All methods assume that the given FSM be deterministic and minimal, and its graph representation D be strongly connected.

3.1. The T-Method

In the T-method, a transition tour (TT) of an FSM is generated which takes the FSM from its start state s_o, executes every transition of the FSM at least once, and returns to the start state s_o. A minimum-length TT can be generated by using the solution of the Chinese Postman Problem [27]. Consequently, finding a minimum-length TT of an FSM becomes equivalent to finding a Chinese postman tour of the digraph D [43]. The T-method can only detect output errors. The state that the FSM reaches after the execution of a transition is not checked as specified in step 3) in the section 2.2.

In certain protocol implementations, each state of the FSM can be identified by applying a status message. A *status message* is a single input which identifies each state of the FSM by producing a unique output for the state and does not cause a state transition. In this case, step 3) of the transition-level approach in the section 2.2 can be easily incorporated into the T-method by using the "status message" input to verify the state after each transition is executed for the first time [43]. For most protocols which do not have the "status message" feature, several methods, such as the W-method, the D-method and the UIO-method, have been developed to generate test sequences which detects the state transfer errors as well as the output errors.

3.2. The W-Method

The W-method involves deriving a characterization set of a completely specified FSM [15, 20]. A *characterization set* (W-set) $\mathbf{W} = \{w_1, w_2, ..., w_k\}$ for an FSM is a set of input sequences which can distinguish the behaviours of every pair of states in the FSM. A testing-tree is constructed such that each branch of the tree represents a transition of the FSM, each node of the tree is a state of the FSM, and each transition of the FSM appears in the tree exactly once. From the testing tree, a set $\mathbf{P} = \{p_1, p_2, ..., p_n\}$ is obtained which contains all non-maximal partial paths in the tree that starts at the root of the tree (the start state s_o of the FSM). Each non-maximal partial path p_i is a minimum-length sequence of inputs that takes the FSM from its s_o to a particular state s_j from which the transition to be tested next emanates. The empty sequence p_o is considered to be a member of **P**. A test sequence can then be obtained by simply concatenating **P** and **W** as follows:

$$\text{Test Sequence} = \mathop{@}_{i=0}^{n} \; \mathop{@}_{j=1}^{k} \; (@\, r_i @\, p_i @\, w_j)$$

where @ means concatenation of two paths.

3.3. The D-Method

The D-method [21] is designed for a completely specified FSM with a *Distinguishing Sequence* (DS). An input sequence is a DS of an FSM if the output produced by the FSM in response to the DS is unique for each state of the FSM. The main idea of the D-method is to construct an i@DS-diagram denoted by D[E']=(V,E'), where E' = $\{(v_j,v_r;\ i@DS): (v_j,v_k;\ i/o) \in E,\ tail(DS) = v_r \in V\}$ and $(v_j,v_k;\ i/o)$ is the transition to be tested. Then a minimal covering of the i@DS-diagram will constitute a test sequence for the FSM [21].

A *covering* is a partition of all the edges of a digraph into simple paths which are disjoint (i.e., having no common edges). A *minimal covering* of a digraph D is a covering of D which involves the fewest possible simple paths [8]. A minimal covering of the i@DS-diagram can be constructed by either using Gonenc's approach [21] or by applying the RCPT (rural Chinese postman tour) algorithm to D over E' [42] in a way similar to the application of the RCPT to the UIO-method [1] described in the following.

3.4. The UIO-Method

This method involves constructing a test sequence for an FSM using a *unique input/output* (UIO) sequence for each state of the FSM. A UIO sequence for a state s_j of the FSM, denoted by UIO(s_j) is an input/output behaviour that is not exhibited by any other state of the FSM [33]. The minimum length UIO sequences for an FSM can be produced by an algorithm developed by Sabnani and Dahbura [33]. The procedure for obtaining a test sequence described in [33] has the following three major steps:

Step 1. Compute the shortest paths from the start state to all states using Dijkstra's Algorithm [14]. The shortest such path to state $v \in V$ is represented by P(v).

Step 2. Construct test subsequence TE(e) for each edge e:

TE(e) = P(head(e)) @ Label(e) @ UIO(tail(e)), where Label(e) is the label of e.

A test sequence is then:

$TS = @_{e \in E}$ ri/null @ TE(e)

Step 3. Optimize the test sequence generated in Step 2 by eliminating those subsequences which are completely contained in the others.

The UIO-method assumes that UIO sequences are unique for an FSM. This uniqueness may not be preserved in a faulty implementation of the FSM [10]. A modified

method, called UIOv-method is proposed [10] which adds a verification procedure to the UIO-method to ensure that UIO sequences are unique in the implementation FSM.

Recently several techniques have been proposed to improve the UIO-method with reductions in the length of the resulting test sequences. A graph traversal technique for constructing RCPT is applied in combination with UIO sequences to yield an efficient method for computing a test sequence for an FSM [1]. This method, called SUIO-method, involves constructing a digraph D'=(V',E') from D=(V,E) such that V'=V and E'=E $\cup$ E_C where E_C = {(v_j,v_r; i/o@UIO(v_k)): (v_j,v_k; i/o) $\in$ E, tail(UIO(v_k)) = v_r $\in$ V} and the cost of (v_j,v_r; i/o@UIO(v_k)) is defined as the length of UIO(v_k) plus one.

It was shown [1] that a test sequence for a given D=(V,E) can be obtained by finding an RCPT of D over E_C, which is a minimum-cost tour traversing every edge in E_C at least once. Computing such a tour is known to be NP-complete in the most general case [28]. Nevertheless, if the edge-induced spanning subgraph D[E_C] of D' is weakly connected, then any symmetric augmentation of D' is strongly connected [1] and hence contains an Euler tour. It was shown [1] that as long as D possesses a reset or self-loop feature, D[E_C] of D' is weakly connected. A digraph D=(V,E) with start vertex v_o is said to have a *reset feature* if for each vertex v $\in$ V, there is an edge (v,v_o; ri/null). A digraph D=(V,E) is said to have a *self-loop feature* if for each vertex v $\in$ V, there is a self-loop edge (v, v; i/o). A minimum-cost rural symmetric augmentation D*=(V*,E*) of D' is constructed as follows: each edge in E is included in E* zero or more times and each edge in E_C is included in E* at least once such that the total cost of edges in E* is minimized. An Euler tour of D* is an RCPT of D' over E_C which in turn is a test sequence for the FSM. Thus finding an RCPT and hence a test sequence for the FSM is equivalent to finding a minimum-cost rural symmetric augmentation of D', which can be reduced to a minimum-cost maximum-flow problem.

An FSM may have multiple UIO sequences of minimum length for each state. Only single UIO sequence for each state of the FSM is used in the UIO-method and SUIO-method described above. A judicious choice of a UIO sequence for each state in constructing the set of edges, E_C, could reduce substantially the length of a test sequence [36, 42]. By using the multiple UIO sequences, [36] has developed a method, called MUIO-method, which converts an FSM into a U-graph and generates a test sequence by the RCPT algorithm.

Both the SUIO-method and the MUIO-method generate test sequences of minimal length, given that no test subsequence for a transition overlaps with any other test subsequence [12, 36]. The length of a test sequence can further be reduced by removing overlaps of test subsequences. In [5], it was shown that achieving maximal overlapping of test subsequences in the most general case is an NP-complete problem. This result, however, does not preclude the existence of heuristics for special cases. Indeed, the technique developed in [11] derives a test sequence as a solution to a minimum cost maximum cardinality matching problem. In [30], a different heuristic technique is proposed which uses a tour having UIO sequences embedded in it. In [42, 44], multiple UIO sequences are employed in combination with overlap removal to produce significant reduction in the length of the test sequences.

4. EXISTENCE OF A CORRECTLY-ORDERED POSTMAN TOUR

In order to generate a synchronizable test sequence for an FSM, the existence of such a sequence has to be determined. In [34], the absence of a synchronizable test sequence is related to the existence of *nonsynchronizable transitions* and/or *nonsynchronizable states*. A transition t_{jk} from state j to state k is called nonsychronizable if no transition t_{ij} from any state i to state j forms a synchronizable pair with t_{jk} [34]. A nonsynchronizable state is a state which can only be reached through nonsynchronizable transitions [34]. Clearly, if there exists a nonsynchronizable transition or state, the synchronization problem can not be avoided and no synchronizable test sequence can be obtained. On the other hand, however, it was observed that the absence of nonsynchronizable transitions or states does not guarantee the existence of a synchronizable test sequence [6]. For example, the FSM represented by digraph D shown in Figure 2, taken from [6], contains no nonsynchronizable transition or state, since every edge has at least one eligible successor, yet it clearly cannot contain a synchronizable test sequence for the FSM.

We call a digraph D=(V,E) *order-specified* if for each edge e = (i,j) in E, we specify a subset of the outgoing edges of vertex j as eligible successor edges for e. Note that, for this definition, we allow multiple edges to have distinct labels. We say a path of an ordered-specified digraph D is *correctly-ordered* (CO) if for every consecutive pair of edges (i,j) and (j,k) in the path, (j,k) has been specified as an eligible successor of (i,j). The concepts of the *CO tour* , the *CO postman tour* (COPT), the *CO Euler tour* and the *CO Chinese postman tour* (COCPT) are similarly defined. Accordingly, given an order-specified digraph D=(V,E) representing an FSM, a synchronizable test sequence corresponds to a COPT which starts at a specified state (*i.e.*, start state s_0) of D; a

minimum-length synchronizable test sequence corresponds a CO Chinese postman tour which starts at state s_o.

In an order-specified digraph D = (V,E) with a specified vertex v_o, an edge (p,q) is said to be

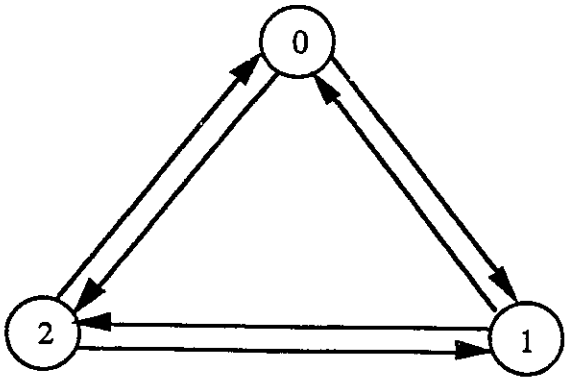
a) *reachable from* an edge (i,j) in a *correctly ordered sense* (COS) and is denoted by (i,j) $\rightarrow$ (p,q) if there is a CO path P = i,j,...p,q;

b) *reachable from (i,j) via v_o in a COS* and is denoted by (i,j) $\rightarrow 0 \rightarrow$ (p,q) if there are edges (u,v_o) and (v_o,w) such that (i,j) $\rightarrow$ (u,v_o) and $(v_o,w) \rightarrow$ (p,q).

We enumerate edges in D=(V,E) as $e_1, e_2, ..., e_n$, where n = |E|, *i.e.*, E = $\{e_i: i = 1,2,...,n\}$ and define the *edge reachability matrix* as an nxn matrix $\mathbf{R} = (r_{ij})$, where $r_{ij} = 1$ if edge $e_i \rightarrow e_j$, $r_{ij} = 0$ if $e_i \rightarrow e_j$ does not hold but $e_i \rightarrow 0 \rightarrow e_j$, and $r_{ij} = -1$ otherwise for $1 \leq i, j \leq n$. It is convenient to define $r_{ii} = 1$ for $1 \leq i \leq n$.

Given a digraph D, there exists a postman tour of D if and only if D is strongly connected [17]. By contrast, an order-specified digraph D may not contain a COPT, even if D is strongly connected in a COS (*i.e.*, if for every pair of vertices v_i and v_j, there exists a CO path from v_i to v_j). For example, it can easily be checked that the order-specified digraph D shown in Figure 2 is strongly connected in a COS.

However, an analogous necessary and sufficient condition has been developed by Boyd and Ural [6] to determine the existence of a COPT for an order-specified digraph D=(V,E) with a specified vertex v_o. This condition which is given as Theorem 4.1 below is based on the following definition: An order-specified digraph D=(V,E) with a specified vertex v_o is called *strongly biconnected* if for any two edges (i,j) and (p,q) $\in$ E, there is a CO tour in D which starts at v_o and contains (i,j) and (p,q).



Order Specifications:

i) (0,1): eligible successor (1,0)

ii) (1,0): eligible successor (0,1)

iii) (0,2): eligible successor (2,0)

iv) (2,0): eligible successor (0,2)

For all other edges there is no restriction on eligible successor edges.

Figure 2. A Digraph D containing no COPT

Theorem 4.1. An order-specified digraph D=(V,E) with a specified vertex v_o contains a COPT if and only if D is strongly biconnected.

In order to develop an efficient algorithm to determine whether a given ordered-specified digraph is strongly biconnected, we develop an equivalent condition as follows.

Theorem 4.2. An order-specified digraph D=(V,E) with a specified vertex v_o is strongly biconnected if and only if v_o is not isolated (*i.e.*, there is at least one edge in E which is incident with v_o) and for each pair of edges (v_p,v_q) and (v_s,v_t) in E, at least one of following conditions holds:

(i) $(v_p,v_q) \rightarrow (v_s,v_t)$ and $(v_s,v_t) \rightarrow (v_p,v_q)$;

(ii) $(v_p,v_q) \rightarrow (v_s,v_t)$ and $(v_s,v_t) \rightarrow 0 \rightarrow (v_p,v_q)$;

(iii) $(v_s,v_t) \rightarrow (v_p,v_q)$ and $(v_p,v_q) \rightarrow 0 \rightarrow (v_s,v_t)$.

Proof: The necessity is obvious. The sufficiency follows the following two claims.

Claim I: for any pair of edges (v_p,v_q) and (v_s,v_t) of which one is incident with v_o, there is a CO tour which starts at vertex v_o and contains edges (v_p,v_q) and (v_s,v_t).

Claim II: for any pair of edges (v_p,v_q) and (v_s,v_t) of which none is incident with v_o, there is a CO tour which starts at vertex v_o and contains edges (v_p,v_q) and (v_s,v_t).

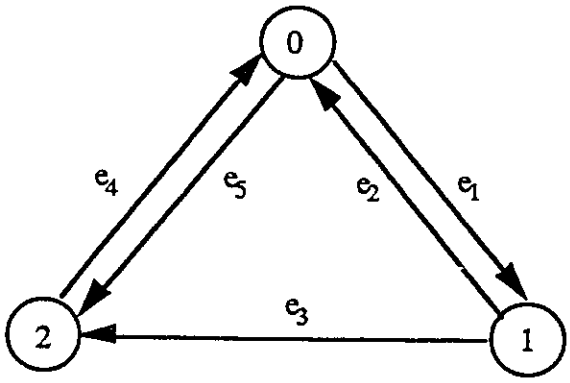
Proof of Claim I.

We have either $v_p = v_o$, $v_q = v_o$, $v_s = v_o$ or $v_t = v_o$. Suppose that $v_p = v_o$. It is obvious that condition (ii) or (iii) implies that there is a CO tour which starts at vertex v_o and contains edges (v_p,v_q) and (v_s,v_t). On the other hand, condition (i) implies that there exist CO paths $P_1 = v_o, v_q, \ldots, v_s, v_t$ and $P_2 = v_s, v_t, \ldots, v_o, v_q$. Then $T = v_o, v_q, \ldots, v_s, v_t, \ldots, v_o$ is a CO tour which starts at vertex v_o and contains edges (v_p,v_q) and (v_s,v_t). Such a tour can similarly be constructed for the case of $v_q = v_o$, $v_s = v_o$ or $v_t = v_o$.

Proof of Claim II.

Again, condition (ii) or (iii) directly implies that there is a CO tour which starts at vertex v_o and contains edges (v_p,v_q) and (v_s,v_t). Condition (i) implies that there exist CO paths $P_1 = v_p, v_q, \ldots, v_s, v_t$ and $P_2 = v_s, v_t, \ldots, v_p, v_q$. Since v_o is not isolated, there exists an edge e which is incident with v_o. From Claim I, there exists a CO tour $T_1 = v_o, v_{i_1}, \ldots,$

v_{i_j}, v_p, v_q, $v_{i_{j+1}}$, ...,v_{i_k}, v_o, which starts at v_o and contains e and (v_p,v_q). Then $T = v_o, v_{i_1}, ..., v_{i_j}, v_p, v_q, ..., v_s, v_t, ..., v_p, v_q, v_{i_{j+1}}, ..., v_{i_k}, v_o$ is a CO tour which starts at v_o and contains edges (v_p,v_q) and (v_s,v_t). ■



Edge Types and Order Specifications:

$e_1 = (0,1)$, **type** $(e_1) \in R^{IU}S^{IL}$, Eligible Successor: e_2, e_3

$e_2 = (1,0)$, **type** $(e_2) \in R^{IU}S^{IL}$, Eligible Successor: e_1, e_5

$e_3 = (1,2)$, **type** $(e_3) \in R^{IL}S^{IL}$, Eligible Successor: e_4

$e_4 = (2,0)$, **type** $(e_4) \in R^{IL}S^{IL}$, Eligible Successor: e_5

$e_5 = (0,2)$, **type** $(e_5) \in R^{IL}S^{IL}$, Eligible Successor: e_4

Figure 3. A Strongly Biconnected Digraph D

For example in the digraph D with a specified vertex $v_o = 0$ in Figure 2, neither $(2,0) \rightarrow (0,1)$ or $(2,0) \rightarrow (0,1)$ is true. Hence the conditions of Theorem 4.2 cannot be satisfied for the pair of edges (2,0) and (0,1) which indicates that D is not strongly biconnected.

For the digraph D shown in Figure 3, the reachability relations between edges are listed in Table 1. It can be verified that the conditions of Theorem 4.2. are satisfied, hence D is strongly biconnected.

$(0,1) \rightarrow (1,0)$	$(0,1) \rightarrow (1,2)$	$(0,1) \rightarrow (2,0)$	$(0,1) \rightarrow (0,2)$
$(1,0) \rightarrow (0,1)$	$(1,0) \rightarrow (1,2)$	$(1,0) \rightarrow (2,0)$	$(1,0) \rightarrow (0,2)$
$(1,2) \rightarrow 0 \rightarrow (0,1)$	$(1,2) \rightarrow 0 \rightarrow (1,0)$	$(1,2) \rightarrow (2,0)$	$(1,2) \rightarrow (0,2)$
$(2,0) \rightarrow 0 \rightarrow (0,1)$	$(2,0) \rightarrow 0 \rightarrow (1,0)$	$(2,0) \rightarrow 0 \rightarrow (1,2)$	$(2,0) \rightarrow (0,2)$
$(0,2) \rightarrow 0 \rightarrow (0,1)$	$(0,2) \rightarrow 0 \rightarrow (1,0)$	$(0,2) \rightarrow 0 \rightarrow (1,2)$	$(0,2) \rightarrow (2,0)$

Table 1. The Reachability Relations Between Edges of D in Figure 3

5. AN ALGORITHM TO DETERMINE THE EXISTENCE OF A COPT

By the definition of reachability matrix $\mathbf{R} = (r_{ij})$, Theorem 4.2 has the following direct corollary.

Corollary 5.1. An order-specified digraph D=(V,E) with a specified vertex v_o is strongly biconnected if and only if $r_{ij}+r_{ji} > 0$ for $i \neq j$ and $i,j \in \{1,2,...,n\}$.

By Corollary 5.1, whether a given digraph is strongly biconnected can be determined in time $O(|E|^2)$ if the reachability matrix **R** is given. An efficient algorithm to determine the strong biconnectivity can thus be derived by further developing an algorithm to obtain **R** with the complexity of $O(|E|^2)$.

As stated earlier, an FSM is represented by digraph D=(V,E), where each edge corresponds to a state transition and each transition is initiated by one of the two testers, upper or lower tester. Hence for each vertex v in D=(V,E) (or state v in the corresponding FSM), there are two lists of edges leaving v: the *upper leaving list* ($Leave^U[v]$) and the *lower leaving list* ($Leave^L[v]$). $Leave^U[v]$ contains the edges corresponding to the transitions initiated at state v by the upper tester and $Leave^L[v]$ contains the edges corresponding to the transitions initiated at state v by the lower tester. In other words, $Leave^U[v] = \{e \in E$: where head(e) = v and **type**(e) $\in \{R^{IU}, R^{IU}S^{IL}, R^{IU}S^{IU}, R^{IU}S^{IU,L}\}\}$ and $Leave^L[v] = \{e \in E$: where head(e) = v and **type**(e) $\in \{R^{IL}, R^{IL}S^{IU}, R^{IL}S^{IL}, R^{IL}S^{IL,U}\}\}$.

5.1. An Algorithm

Based on Corollary 5.1, we present the algorithm RMatrix in the following which determines whether a given order-specified digraph representing an FSM with two testers is strongly biconnected. The core of algorithm RMatrix is the procedure of half-breadth-

```
RMatrix(D,v_o) {To obtain the reachability matrix for D=(V,E) with a specific vertex v_o=0}
1   for every pair of edges e, f ∈ E[D] and e ≠ f
2     do R[e,f] ← -1
3   for every edge e ∈ E[D]
4     do R[e,e] ← 1
5        if e ∈ {R^IL S^IU, R^IU S^IL, R^IL S^IL,U, R^IU S^IU,L}
6             then tester-set[e] ← {0,1}        {0 stands for upper tester, 1 lower tester}
7        else if e ∈ {R^IU, R^IU S^IU}
8                  then tester-set[e] ← {0}
9             else tester-set[e] ← {1}                          {e ∈ {R^IL, R^IL S^IL}}
10  for each edge e ∈ E[D]
11     do HBFS(e)

HBFS(e)
1   for p=0 to 2(|V|-1)
2      do colour[p] ← WHITE                          {Set all half-vertices to WHITE}
3   v=tail(e)
4   for each tester ∈ tester-set[e]
5      do Do-Enqueue(2v+tester)
6   Search(e,1)                                  {Search for edges reachable in a COS}
7   if colour[0] = WHITE and colour[1] = BLACK
8      then Do-Enqueue(0)                        {Enqueue upper half-vertex of v_o}
9   else if colour[0] = BLACK and colour[1] = WHITE
10          then Do-Enqueue(1)                   {Enqueue lower half-vertex of v_o}
11  Search(e,0)                          {Search for edges reachable in a COS via v_o}

Search(e,index)
1   while Q ≠ Ø
2      do q ← head(Q)
3         for each edge f ∈ Leave[q] - {e}
4            do R[e,f] ← index
5               v ← tail(f)
6               for each tester ∈ tester-set[f]
7                     do Do-Enqueue(2v+tester)    {2v+tester: a half-vertex of vertex v}
8         Dequeue(Q)
9         colour[q] ← BLACK

Do-Enqueue(h)                                                {h is a half-vertex}
1   if colour[h] = WHITE
2      then colour[h] ← GRAY
3           Enqueue(Q,h)
4   else do nothing
```

first-search HBFS, which is similar to breadth-first search [38]. Though both expand the frontier between discovered and undiscovered objects uniformly across the breadth of the frontier, the objects of breadth-first search are vertices but the objects of half-breadth-first search are half-vertices. Each vertex v is divided into upper and lower halves such that the upper half can conceptually be thought of the origin of $Leave^U[v]$ and the lower half the origin of $Leave^L[v]$.

Procedure RMatrix assumes that the input graph D=(V,E) with a specific vertex v_o is represented using lists of leaving edges such that each half-vertex h is associated with a list of leaving edges Leave[h]. The vertices are numbered from 0 to |V|-1 with v_o=0. The half-vertices are numbered from 0 to 2(|V|-1) such that for each vertex v, the upper half-vertex is assigned 2v and the lower half-vertex 2v+1. $Leave[2v] = Leave^U[v]$ and $Leave[2v+1] = Leave^L[v]$.

Procedure RMatrix initializes the entries of R to be -1's, sets diagonal entries to be 1's, determines the tester-set for each edge and calls procedure HBFS for each edge.

Given an edge e, half-breadth-first search starts from the half-vertex (or half-vertices) of tail(e). When half-breadth-first search continues from a half-vertex, the adjacent edges of the half-vertex are swept to fill corresponding entries of the reachability matrix **R** and to explore its adjacent half-vertices. After the search for the edges reachable from e in a COS, the search continues for the edges reachable from e via v_o in a COS from a half-vertex of v_o.

To keep track of progress, half-breadth-first search colours half-vertices white, gray and black. The colour of each half-vertex h is stored in the variable colour[h]. The algorithm also employs a FIFO queue Q to manage the gray half-vertices. All half-vertices start out white and may later become gray then black. A half-vertex is discovered the first

time it is encountered during the search and becomes gray. A gray half-vertex becomes and remains black after all its adjacent half-vertices are gray. Thus gray half-vertices represent the frontier between discovered and undiscovered half-vertices. The distinction of the discovered half-vertices ensures that the search proceeds in a breadth-first manner.

Procedure HBFS with input edge e works as follows. Lines 1-2 paint every half-vertices white, lines 3-4 call procedure Do-Enqueue once or twice depending on the tester-set(e) to enqueue the half-vertex of tail(e), line 5 sets v to tail(e), line 6 calls procedure Search for edges reachable from e in a COS, lines 7-10 call procedure Do-Enqueue at most once to enqueue the half-vertex of v_o, and line 11 calls procedure Search for edges reachable from e in a COS via v_o.

The main loop of procedure Search iterates as long as there remain gray half-vertices. Line 2 determines half-vertex q at the head of the queue Q. The **for** loop of lines 3-7 consider each edge in Leave[q]. This loop also contains another **for** loop of line 6-7, which calls Do-Enqueue once or twice depending one or two half-vertices in tester-set[q]. When all the adjacency edges of half-vertex p have been examined, p is removed from queue Q and blackened in lines 8-9.

Procedure Do-Enqueue checks the colour of half-vertex q and, if white, paints q gray and places q at the tail of queue Q.

5.2. An Example

Applying the algorithm RMatrix to the order-specified digraph D in Figure 3, we will obtain the reachability matrix **R** as follows.

$$\mathbf{R} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The matrix **R** satisfy the condition of Corollary 5.1, which indicates again that D is strongly biconnected.

To show how the algorithm works, we will follow the process in filling the entries in the third row of **R** by calling the procedure HBFS(e_3). Before we start, sets of Leave[x] and tester-set[e] are listed below, where x is a half vertex and e is an edge.

Leave[0] = Leave^U[0] = $\{e_1\}$
Leave[1] = Leave^L[0] = $\{e_5\}$
Leave[2] = Leave^U[1] = $\{e_2\}$
Leave[3] = Leave^L[1] = $\{e_3\}$
Leave[4] = Leave^U[2] = $\emptyset$
Leave[5] = Leave^L[2] = $\{e_4\}$

tester-set[e_1] = {0,1}
tester-set[e_2] = {0,1}
tester-set[e_3] = {1}
tester-set[e_4] = {1}
tester-set[e_5] = {1}

The change of variables is traced step by step in the following and the change of colours of half vertices is shown in Figure 4(i) - Figure 4(viii), where each vertex v of D is labelled with its upper half vertex 2v and lower half vertex 2v+1.

Step 1. Before HDFS(e_3) is called (Figure 4(i)):

$\mathbf{R}[e_3,e_3] = 1$.

Step 2. After processing line 5 of HBFS(e_3) (Figure 4(ii)):

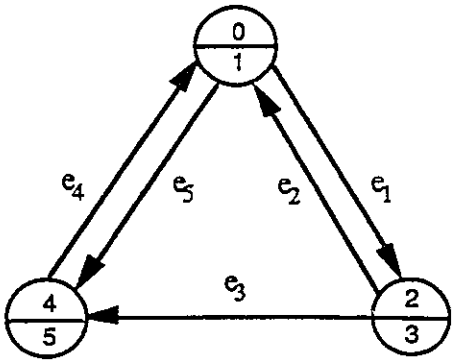
v = 2, Q = (5), colour[0] = colour [1] = colour[3] =colour[4] = WHITE, colour[5] = GRAY.

Step 3. After processing the first loop of **while** in Search(e_3, 1) (Figure 4(iii)):

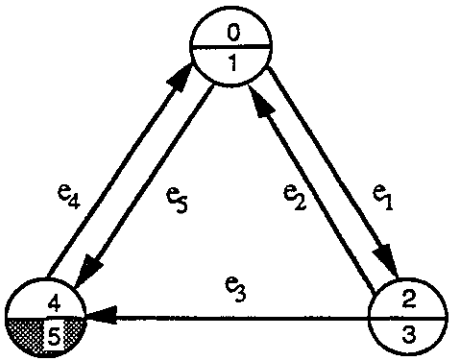
q = 5, $\mathbf{R}[e_3,e_4] = 1$, v = 0, Q = (1), colour [1] = GRAY, colour[5] = BLACK.

Step 4. After processing the second loop of **while** in Search(e_3, 1) (Figure 4(iv)):

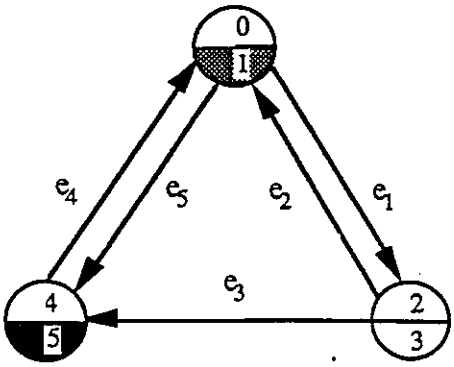
q = 1, $\mathbf{R}[e_3,e_5] = 1$, Q = $\emptyset$, colour [1] = BLACK.



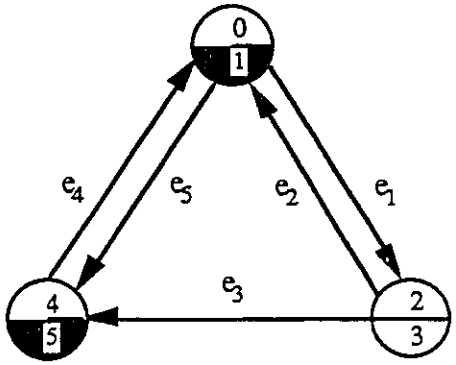
(i)



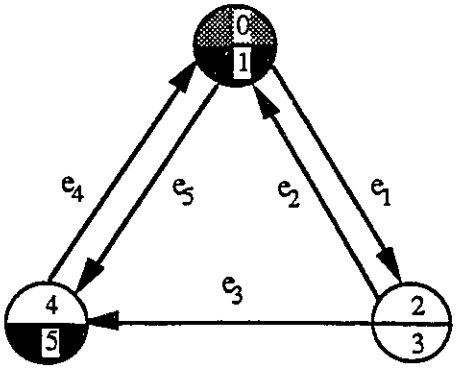
(ii)



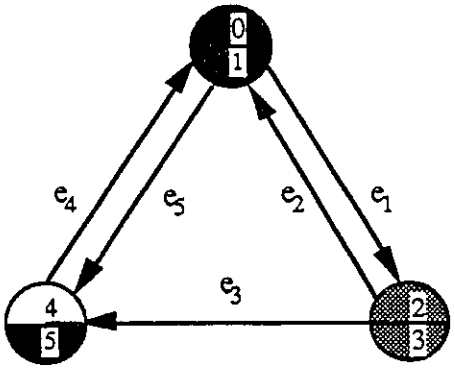
(iii)



(iv)



(v)



(vi)

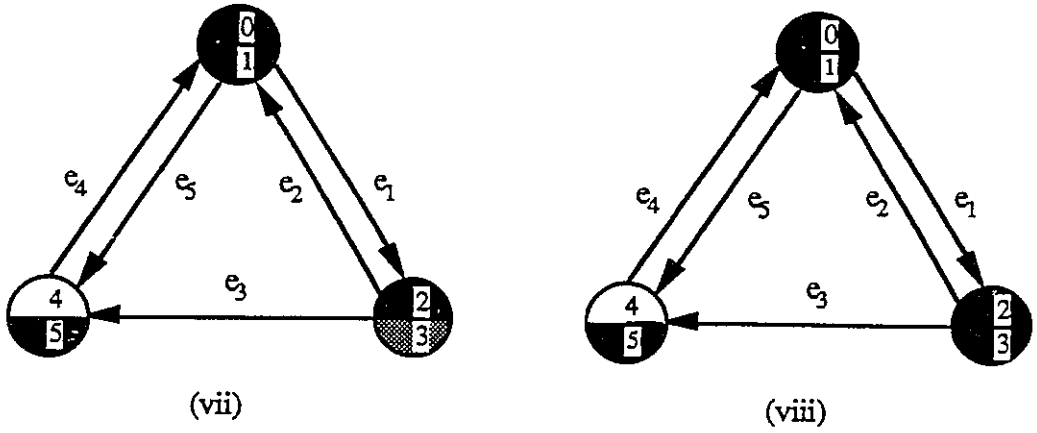


Figure 4 (i - viii). Tracing the Process of the Algorithm RMatrix

Step 5. After processing line 10 of HBFS(e_3) (Figure 4(v)):

Q = (0), colour [0] = GRAY.

Step 6. After processing the first loop of **while** in Search(e_3, 0) (Figure 4(vi)):

q = 0, $\mathbf{R}[e_3,e_1]$ = 0, v = 1, Q = (2,3), colour [2] = colour[3] = GRAY, colour[0] = BLACK.

Step 7. After processing the second loop of **while** in Search(e_3, 0) (Figure 4(vii)):

q = 2, $\mathbf{R}[e_3,e_2]$ = 0, v = 0, Q = (3), colour [2] = BLACK.

Step 7. After processing the third loop of **while** in Search(e_3, 0) (Figure 4(viii)):

q = 3, Q = Ø, colour [3] = BLACK.

5.3. Complexity Analysis of the Algorithm

We first consider procedure HBFS. After initialization, no vertex is ever whitened, and thus the test of line 1 in procedure Do-Enqueue ensures that each half-vertex is enqueued and hence dequeued at most once. The operations of enqueuing and dequeuing take O(1) time, so the total time devoted to queue operations is O(V). Each list of leaving

edges is scanned at most once since it is done only when the half-vertex is dequeued. During the time each leaving edge is examined, the **for** loop (lines 6-7 in HBSF) only iterates at most twice because there are only two testers. Hence examination of each edge takes O(1) time. Since the total sum of the lengths of the leaving edge lists is $\Theta(E)$, at most O(E) time is spent on scanning leaving edge lists. The overhead for initialization is O(V). Thus, HBFS takes O(V+E) time.

RMatrix calls HBFS O(E) times. The overhead of RMatrix is $O(E^2)$. Thus, the complexity of the procedure of RMatrix is $O(EV + E^2)$. If we assume there is no isolated vertex, we have $|E| \geq |V|$. The complexity is then $O(E^2)$.

6. SYNCHRONIZABLE TEST SEQUENCES FOR OUTPUT FUNCTIONS

In this chapter, we will develop a duplex digraph method, described in section 6.1 below, to generate synchronizable test sequences. A duplex digraph technique was originally described by Chen *et al.* [13], in which a digraph D=(V,E) is converted into a duplex digraph D'=(V',E') by the following two steps:

(1) for a vertex u in V, a pair of vertices u^U and u^L in V' are created,

(2) for an edge (u,v) in E, an edge is created in E' which is

(u^L,v^L), if **type**((u,v)) = $R^{IL}S^{IL}$, or

(u^L,v^U), if **type**((u,v)) = $R^{IL}S^{IU}$, or

(u^U,v^L), if **type**((u,v)) = $R^{IU}S^{IL}$, or

(u^U,v^U), if **type**((u,v)) = $R^{IU}S^{IU}$.

For example, the conversion of D in Figure 3 into the duplex digraph D' by the above two steps is shown in Figure 5. A Chinese postman tour in D', if it exists, is then a minimum length synchronizable test sequence for the FSM represented by D.

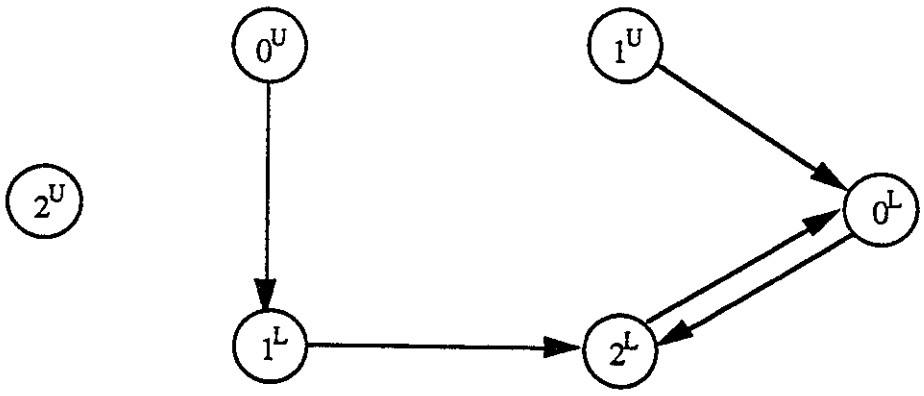


Figure 5. A Duplex Digraph of D in Figure 3 by the Method in [13]

However, this method suffers two deficiencies. First, a postman tour (and hence a Chinese postman tour) in D' does not, in general, exist even when there is a synchronizable test sequence. For example in Figure 3, T = 0,1,0,1,2,0,2,0, is a COPT in D and hence a synchronizable test sequence, but no postman tour exists in the duplex digraph D' in Figure 5. Second, some types of transitions are not covered, *i.e.*, the conversion of edges of types R^{IU}, R^{IL}, $R^{IU}S^{IU,L}$ and $R^{IL}S^{IL,U}$ is not given.

The duplex digraph method we developed in section 6.1 below overcomes the above deficiencies. Our method differs from the original duplex digraph method in two ways:

(1) we create either an edge or three edges in D' for each edge e in D according to the type of e, *i.e.*, **type**(e);

(2) we consider all types of edges , *i.e.*, any edge e with **type**(e) $\in$ $\{R^{IU}, R^{IU}S^{IU}, R^{IL}, R^{IL}S^{IL}, R^{IU}S^{IL}, R^{IL}S^{IU}, R^{IU}S^{IU,L}, R^{IL}S^{IL,U}\}$.

6.1. The Duplex Digraph

We define, for each vertex v in D=(V,E), $Arrive^{U}[v] = \{e \in E: tail(e) = v$ and **type**(e) $\in \{R^{IU}, R^{IU}S^{IU}\}\}$, $Arrive^{L}[v] = \{e \in E: tail(e) = v$ and **type**(e) $\in \{R^{IL}, R^{IL}S^{IL}\}\}$ and $Arrive^{U,L}[v] = \{e \in E: tail(e) = v$ and **type**(e) $\in \{R^{IU}S^{IL}, R^{IL}S^{IU}, R^{IU}S^{IU,L}, R^{IL}S^{IL,U}\}\}$. Recalling the definitions of $Leave^{U}[v]$ and $Leave^{L}[v]$, we know that edges in $Leave^{U}[v]$ are eligible successors of edges in $Arrive^{U}[v]$ and $Arrive^{U,L}[v]$, edges in $Leave^{L}[v]$ are eligible successors of edges in $Arrive^{L}[v]$ and $Arrive^{U,L}[v]$. Furthermore, edges in $Leave^{U}[v]$ are the only eligible successors of edges in $Arrive^{U}[v]$, and edges in $Leave^{L}[v]$ are the only eligible successors of edges in $Arrive^{L}[v]$.

The existence of a COPT in D=(V,E) starting from v_o, or, equivalently, a strongly biconnected digraph D=(V,E) with a specified vertex v_o, ensures that, for each vertex $v \in V-\{v_o\}$,

if $Leave^U[v] = \emptyset$ then $Arrive^U[v] = \emptyset$, and

if $Leave^L[v] = \emptyset$ then $Arrive^L[v] = \emptyset$.

Note that, in the above cases, $Arrive^{U,L}[v]$ may or may not be empty.

We create a duplex digraph D'=(V',E'), where $V' = V^U \cup V^L \cup H$ and $E' = E_C \cup F$, from a strongly biconnected digraph D=(V,E) with a specified vertex v_o as follows.

(a) For each vertex v in V, there are two sets of edges leaving v: $Leave^U[v]$ and $Leave^L[v]$. We create a vertex v^U in V^U if $Leave^U[v] \neq \emptyset$ and a vertex v^L in V^L if $Leave^L[v] \neq \emptyset$. In addition for vertex v_o, if $Leave^U[v_o] = \emptyset$ but $Arrive^U[v_o] \neq \emptyset$, we create vertex $v_o{}^U$ in V^U; similarly, if $Leave^L[v_o] = \emptyset$ but $Arrive^L[v_o] \neq \emptyset$, we create vertex $v_o{}^L$ in V^L.

(b) For each edge $(u,v) \in Arrive^U[v]$ (that implies $(u,v) \in Leave^U[u]$), we create a directed edge, in E_C, from u^U to v^U. Similarly, for each edge $(u,v) \in Arrive^L[v]$, we create a directed edge, in E_C, from u^L to v^L.

c) For each edge $(u,v) \in Arrive^{U,L}[v]$, one of following is performed:

i) In the case that vertex v^U exists but vertex v^L does not, we create a directed edge, in E_C, from u^U to v^U, if $(u,v) \in Leave^U[u]$; if $(u,v) \in Leave^L[u]$, we create a directed edge, in E_C, from u^L to v^U.

ii) In the case that vertex v^L exists but vertex v^U does not, we create a directed edge, in E_C, from u^U to v^L, if $(u,v) \in Leave^U[u]$; if $(u,v) \in Leave^L[u]$, we create a directed edge, in E_C, from u^L to v^L.

iii) In the case that both v^U and v^L exist, we have two subcases: If $(u,v) \in Leave^U[u]$, we create, in H, a vertex $U_{u,v}$, a directed edge $(u^U, U_{u,v})$ in E_C, and two directed edges $(U_{u,v}, v^U)$ and $(U_{u,v}, v^L)$ in F. We call $(u^U, U_{u,v})$ the *parent edge* of $(U_{u,v}, v^U)$ and $(U_{u,v}, v^L)$. We call $(U_{u,v}, v^U)$ and $(U_{u,v}, v^L)$ the *child edges* of $(u^U, U_{u,v})$. $(U_{u,v}, v^U)$ and $(U_{u,v}, v^L)$ are called *twins*. Similarly if $(u,v) \in Leave^L[u]$, we create, in H, a vertex $L_{u,v}$, a directed edge $(u^L, L_{u,v})$ in E_C, and two directed edges $(L_{u,v}, v^U)$ and $(L_{u,v}, v^L)$ in F. The parent edges, the child edges and twin are similarly defined.

d) Each edge in E_C is given cost one and each edge in F is given cost zero.

Note that vertex v_o is treated differently in (a). As we have just indicated, for each $v \in V-\{v_o\}$, if $Leave^U[v] = \emptyset$ then $Arrive^U[v] = \emptyset$. In this case, we do not want to create vertex v^U. Otherwise, either v^U would become isolated or any path entering v^U would have no way out. Hence vertex v^U is created only when $Leave^U[v] \neq \emptyset$. Similarly, vertex v^L is created only when $Leave^L[v] \neq \emptyset$. However, vertex v_o is different from other vertex such that we can have $Arrive^U[v_o] \neq \emptyset$ while $Leave^U[v_o] = \emptyset$. In this case, we need to create vertex $v_o{}^U$ in D' in order for edges in $Arrive^U[v_o]$ to be preserved in D' by the way described in (b). Similar argument applies to the case when $Leave^L[v_o] = \emptyset$ but $Arrive^L[v_o] \neq \emptyset$.

We call an edge e in D' an *inherent edge* if e is neither a parent edge nor a child edge.

6.2. Characteristics of the Duplex Digraph

Lemma 6.1. If an ordered-specified digraph D=(V,E) with a specified vertex v_o is strongly biconnected, then there exists a path P in the corresponding duplex digraph D'=(V',E'), where $V'=V^U \cup V^L \cup H$ and $E' = E_C \cup F$,

which visits every vertex in V' and contains every edge in E_C and at least one edge for each pair of twins.

Proof: From the way the duplex digraph D' is created, we know that each edge e in D gives rise to, in D', either an inherent edge or a parent edge and a pair of child edges (*i.e.*, twins). In the latter case, the twins have the same head vertex but with different tail vertices. Hence, as long as a tail vertex in D' is specified, a segment in D' corresponding to an edge in D can be determined, which is either a single inherent edge or a parent edge followed by the child edge with the specified tail vertex.

From Theorem 4.1, there exists a COPT T in D starting at v_o. Let e_i be the ith edge of T for i= 1,2,...,k, where k is the length of T. We first assume that first_edge(T) = $e_1 \in \text{Leave}^U[v_o]$. To obtain a path P starting at v_o^U, we construct segments $\{s_i: i=1,2,...,k\}$ as follows.

Step 1. If e_k gives rise to an inherent edge in D', we select the tail vertex of the inherent edge (*i.e.*, v_o^U or v_o^L) as the last vertex of the path P, *i.e.*, the vertex at which P ends. Otherwise we select v_o^U as the last vertex of P. Initially, let i=k and v_{last} be the last vertex of P. Go on to Step 2.

Step 2. Construct segment s_i in D' corresponding to edge e_i in D with the specified tail vertex v_{last}. Go on to Step 3.

Step 3. If i=1 then stop. Otherwise, update v_{last} to be the head vertex of s_i and decrement i by one, then go to Step 2.

The path P is readily constructed from segments $\{s_i: i=1,2,...,k\}$ as

$$P = s_1 @ s_2 @ ... @ s_k$$

Since each edge e' in E_C is constructed for some edge e_i in E ($1 \le i \le k$), e' is contained in s_i. Hence P contains every edge in E_C. Since each pair of twins have their parent edge e' in E_C and tail(e') has the twins as the only outgoing edges, P must also contain at least one of the twins.

Now, we will prove that P visits all the vertices in V' by showing that every vertex in V' is contained in some segment s_i ($1 \le i \le k$).

For each vertex $v^U \in V^U$, we have $Leave^U[v] \neq \emptyset$ since it is the condition under which v^U in $V^U - \{v_o^U\}$ is created and we assume $e_1 \in Leave^U[v_o]$. Let e_i be an edge in $Leave^U[v]$ for some i ($1 \le i \le k$). Then v^U is the head vertex of segment s_i. Similarly, each vertex $v^L \in V^L - \{v_o^L\}$ is contained in some s_i ($1 \le i \le k$).

For vertex v_o^L, if it exists and $Leave^L[v_o] = \emptyset$, then $Arrive^L[v_o] \neq \emptyset$ and $e_k \in Arrive^L[v_o]$, hence the tail vertex of s_k is v_o^L. Otherwise if $Leave^L[v_o] \neq \emptyset$, we have just shown that it implies that v_o^L is contained in some s_i ($1 \le i \le k$).

For each vertex $x \in H$, either $x = U_{u,v}$ or $L_{u,v}$ for some edge $e_i = (u,v)$ ($1 \le i \le k$), s_i contains the parent edge e' in D' derived from edge (u,v) in D and hence the vertex x since tail(e') = x.

Similarly, if first_edge(T) = $e_1 \in Leave^L[v_o]$, we can construct the path P starting at v_o^L. ■

Since the existence of a path P visiting every vertex in digraph D' implies that D' is weakly connected, we have the following corollary of Lemma 6.1.

Corollary 6.1. If an ordered-specified digraph D=(V,E) with a specified vertex v_o is strongly biconnected, then the corresponding duplex digraph D'=(V',E') is weakly connected.

We define a given order-specified digraph D with a specified vertex v_o to be *robustly biconnected* if any edge in D is reachable from any other edge in a COS. In terms of reachability matrix $\mathbf{R} = (r_{ij})$, D is robustly biconnected if and only if $r_{ij} = 1$ for all i and j. Obviously, if D is robustly biconnected then D is strongly biconnected.

Lemma 6.2. If an order-specified digraph D with specified vertex v_o is robustly biconnected, then the duplex digraph D'=(V',E') is strongly connected.

Proof: From Theorem 4.1 and the definition of robustly biconnected, there exists a COPT T_o in D starting at v_o. Since D is robustly biconnected, there exists a CO path Q containing last_edge(T_o) as the first edge and first_edge(T_o) as the last edge. Concatenation of T_o and Q gives a COPT T in D such that first_edge(T) is an eligible successor of last_edge(T). By the proof of the Lemma 6.1, we can construct a path P in D' from T such that P visits every vertex in D'. Furthermore, since last_edge(T) can form a synchronizable pair of transitions with first_edge(T), we can choose the first vertex of the path P as the last vertex of P, that is P is also a tour. The existence of such a tour in D' implies that D' is strongly connected. ■

From section 2.6, if the duplex digraph D' is strongly connected, we can obtain an RPT in D' over E_C, which corresponds to a COPT in the digraph D. In general, the strong biconnectivity of the order-specified digraph D with specified vertex v_o does not guarantee the strong connectivity of the duplex digraph D'. However, any digraph can be

decomposed into strongly connected components [4, 18]. The *strongly connected components* of a digraph G are the equivalence classes of vertices under the mutually reachable relation. Two vertices x, y are *mutually reachable from* each other if there exist two paths in G such that one is from x to y and the other is from y to x. For two strongly components G_1 and G_2 in a digraph G, we define that G_2 is *reachable from* G_1, denoted by $G_1 \rightarrow G_2$, if a vertex in G_2 is reachable from a vertex in G_1. Obviously, for any two strongly connected components G_1 and G_2, where $G_1 \neq G_2$, both relations $G_1 \rightarrow G_2$ and $G_2 \rightarrow G_1$ cannot hold at the same time. Otherwise, vertices in $G_1 \cup G_2$ would be mutually reachable from each other and hence G_1 and G_2 would be in the same strongly connected component, which contradicts with the facts the both G_1 and G_2 are strongly connected components.

The decomposition of the duplex digraph D' into strongly connected components has the following characteristics.

Lemma 6.3. If an order-specified digraph D with specified vertex v_o is strongly biconnected, then the duplex digraph D' can be uniquely decomposed into strongly connected components $G_1, G_2, ..., G_k$ for some k, $k \geq 1$, such that $G_i \rightarrow G_j$ for $1 \leq i < j \leq k$. $\{G_i : i=1,2,...,k\}$ is called the *ordered decomposition* of D'.

Proof: Suppose $\{H_i : i=1,2,...,k\}$ is a decomposition of D' into strongly connected components. From Lemma 6.1, there exists a path in D' which visits every vertex in D'. This implies that for each pair of H_i and H_j, where $i \neq j$, either $H_i \rightarrow H_j$ or $H_j \rightarrow H_i$. Sorting $\{H_i : i=1,2,...,k\}$ by the relation of $\rightarrow$ will thus give rise to $\{G_i : i=1,2,...,k\}$. Obviously, such an ordered decomposition is unique. ■

For digraph D'=(V',E'), we define $\Delta(G_i,G_j)$ to be edges in E' directed from G_i to G_j for i<j and i,j=1,2,...,k. We also define subgraph D_i of D' as $D_i=(G_i,E_i)$, where E_i = $\{(u,v) \in E': u,v \in G_i\}$, for i=1,2,...,k.

Lemma 6.4. Suppose that an order-specified digraph D with specified vertex v_o is strongly biconnected and D'=(V',E') is the duplex digraph where $V'=V^U \cup V^L \cup H$ and $E' = E_C \cup F$. Let $\{G_i: i=1,2,...,k\}$ be the ordered decomposition of D'. Then

(i) $\Delta(G_j,G_i) = \emptyset$ for $k \geq j > i \geq 1$;

(ii) If $(x,y) \in \Delta(G_i,G_j)$ for some i and j such that $i+1 < j$, then (x,y) is a child edge and has its twin (x,z) in $D_i \cup \Delta(G_i,G_{i+1})$;

(iii) $\Delta(G_i,G_{i+1}) \neq \emptyset$ for i=1,2,..,k-1;

(iv) There is at most one edge in $E_C \cap \Delta(G_i,G_{i+1})$ for i=1,2,..,k-1;

(v) If there is an edge e in $E_C \cap \Delta(G_i,G_{i+1})$ for some i, where $1 \leq i < k$, then every edge in $\Delta(G_i,G_{i+1})$ - {e} is a child edge and has its twin in D_i;

(vi) If $E_C \cap \Delta(G_i,G_{i+1}) = \emptyset$ for some i, where $1 \leq i < k$, we have three subcases:

(a) every edge in $\Delta(G_i,G_{i+1})$ has its twin in D_i,; or

(b) there is one edge e which has its twin in $\Delta(G_i,G_j)$ for some $j > i+1$ and every edge in $\Delta(G_i,G_{i+1})$ - {e} has its twin in D_i; or

(c) there is only one pair of child edges e and e' in $\Delta(G_i,G_{i+1})$ such that e and e' are twins and every edge in $\Delta(G_i,G_{i+1})$ - {e, e'} has its twin in D_i.

Proof. By Lemma 6.1, there exists a path P in digraph D' such that P visits every vertex in D' and contains every edge in E_C and at least one edge for each pair of twins.

(i): (i) is implied by the fact that $G_i \rightarrow G_j$ but not $G_j \rightarrow G_i$ for $k \geq j > i \geq 1$.

(ii): the path P can not contain any edge in $\Delta(G_s,G_t)$ for $1 \leq s,t \leq k$ and $s+1<t$, otherwise, from (i), P would not be able to visit the vertices in component G_{s+1}. Hence (x,y) can only be a child edge and has its twin (x,z) in the path P, furthermore (x,z) can not be in any $\Delta(G_i,G_t)$ for $t > i+1$. That is, we have that $(x,z) \in D_i \cup \Delta(G_i,G_{i+1})$ and hence (ii) holds. We called $(x,y) \in \Delta(G_i,G_j)$ a *redundant edge.*

(iii): Since $\{G_i : i=1,2,...,k\}$ is the ordered decomposition of D', the path P should visit every vertex in G_i before any vertex in G_j for $1 \leq i<j \leq k$. Therefore, P must go across G_i to G_{i+1} exactly once, for i=1,2,...,k-1. Hence (iii) holds.

(iv): Since P must go across G_i to G_{i+1} exactly once and P contains every edge in E_C, (iv) is true.

(v): We know that P must go across $\Delta(G_i,G_{i+1})$ by e and from (iv), $\Delta(G_i,G_{i+1})$ - {e} contains only child edges. If $\Delta(G_i,G_{i+1})$ - {e} contains a pair of twins, P has to include at least one of them. Then P must go across G_i to G_{i+1} more than once. This is a contradiction. Hence $\Delta(G_i,G_{i+1})$ - {e} does not contain any pair of twins. Suppose (x,y) is a child edge in $\Delta(G_i,G_{i+1})$ - {e} and

(x,z) is its twin. We know $z \notin G_{i+1}$. Since P does not include (x,y), P must include (x,z). We have $z \notin G_j$ for $j>i+1$, from the proof of (ii) and $z \notin D_j$ for $j<i$ from (i). Hence $z \in G_i$, *i.e.*, $(x,z) \in D_i$. We call $(x,y) \in \Delta(G_i,G_{i+1}) - \{e\}$ a *redundant edge*.

(vi): If there are two pairs of twins in $\Delta(G_i,G_{i+1})$, P should include at least one edge from each pair of twin edges, which again leads to the contraction that P must go across G_i to G_{i+1} more than once. Hence $\Delta(G_i,G_{i+1})$ contains at most one pair of twins. It can be shown that any other edge should have its twin in D_i by the same method used in the proof of (v). Hence, we have three subcases of (a), (b) or (c). In the case of (a), we arbitrarily select an edge e from $\Delta(G_i,G_{i+1})$ and call edges in $\Delta(G_i,G_{i+1})-\{e\}$ *redundant edges*; in the case of (b), we call edges in $\Delta(G_i,G_{i+1})-\{e\}$ *redundant edges*; in the case of (c), we arbitrarily select an edge from e and e', say e, and call edges in $\Delta(G_i,G_{i+1})-\{e\}$ *redundant edges*. ■

From Lemma 6.4, we know that no edge in $\Delta(G_i,G_j)$ for $i+1 < j$ is included in any COPT in D and that exactly one edge from $\Delta(G_i,G_{i+1})$ $(1 \le i < k)$ is included in any COPT in D. A digraph D"=(V",E"), where V"=V', is obtained from D' by removing redundant edges from E' as follows:

(i) Redundant edges in $\Delta(G_i,G_j)$ for $i+1 < j$ are removed from D'.

(ii) Redundant edges in $\Delta(G_i,G_{i+1})$ can be removed by the following steps.

Step 1. If there is only one edge in $\Delta(G_i,G_{+1})$ then stop, otherwise go on to Step 2.

Step 2. Remove an edge from $\Delta(G_i,G_{i+1})$ which has a twin in D_i. If no such an edge exists in $\Delta(G_i,G_{+1})$, $\Delta(G_i,G_{i+1})$ contains only a pair of twin edges and we remove one of them. Go to Step 1.

We know that

(a) from Lemma 6.4, D" is still weakly connected;

(b) $\{G_i : i=1,2,...,k\}$ is also an ordered decomposition of D" and D_i is also a subgraph of D", for i=1,2,...,k;

(c) for each pair of G_i and G_{i+1} where $1 \le i < k$, there is exactly one edge in D" directed from G_i to G_{i+1}, which we call a *link* from G_i to G_{i+1};

(d) $E'' \supseteq E_C$ and E" includes at least one edge of each pair of twin edges in D';

(e) each edge in D" except links is contained in some strongly connected subgraph D_i $(1 \le i \le k)$.

6.3. An Algorithm for Synchronizable Test Sequence Generation

With the above properties of D", we can now construct an RPT in D" over E_C starting at either v_o^U or v_o^L if k=1, or an RPP in D" over E_C either from v_o^U to v_o^L or from v_o^L to v_o^U if k>1. In the case when k>1, we have either $v_o^U \in G_1$ and $v_o^U \in G_k$, or $v_o^L \in G_1$ and $v_o^U \in G_k$.

If k=1 (the case when D is robustly biconnected), we have that D' = D" = D_1, *i.e.*, D' is itself strongly connected. Hence there exists an RPT T* in D' over E_C starting at either v_o^U or v_o^U, which corresponds to a COPT T in the digraph D. T is obtained from T* by removing vertices in H from T* and eliminating the superscripts from the upper and the lower vertices.

We now consider the case when k>1. Let $E_{iC} = E_i \cap E_C$ and let (x_i,y_i) be the link from G_i to G_{i+1} for i=1,2,...k-1. Let $y_o = v_o^U$ and $x_k = v_o^L$ when $v_o^U \in G_1$ and $v_o^L \in$

G_k, otherwise let $y_o = v_o^L$ and $x_k = v_o^U$. Then the pair $\{y_{i-1}, x_i\}$ are in the same G_i for i=1,2,...,k. Since D_i is strongly connected, there exists an RPP P_i in D_i over E_{iC} from y_{i-1} to x_i for i=1,2,...,k. Then $T^* = P_1 @ x_1,y_1 @ P_2 @ x_2,y_2 @ ... @ P_{k-1} @ x_{k-1},y_{k-1} @ P_k$ is an RPP in D" over E_C, where @ stands for concatenation of two paths, P_1 starts at y_o, and P_k terminates at x_k. The path T^* then corresponds to a COPT in D starting at v_o.

We can now give an algorithm to construct a COPT in D as follows:

Step 1. Construct the duplex digraph D' from D.

Step 2. Find the strongly connected components $\{H_i : i=1,2,...,k\}$ of D'. If k=1 then go to step 3 else go to step 4.

Step 3. Construct an RPT T^* in D' over E_C starting either from v_o^U or v_o^L according to the method described in section 2.6.1. Since T^* corresponds to a COPT in D, stop.

Step 4. Sort $\{H_i : i=1,2,...,k\}$ into $\{G_i : i=1,2,...,k\}$, the ordered decomposition of D'; Obtain subgraph $D_i=(G_i,E_i)$, where $E_i = \{(u,v) \in E' : u,v \in G_i\}$, and $E_{iC} = E_i \cap E_C$ for i=1,2,...,k.

Step 5. Determine $\Delta(G_i,G_{i+1})$ and remove redundant edges from $\Delta(G_i,G_{i+1})$ for i=1,2,...,k-1.

Step 6. Identify the links (x_i,y_i) from G_i toward G_{i+1} for i=1,2,...k-1. Let $y_o = v_o^U$ and $x_k = v_o^L$ when $v_o^U \in G_1$ and $v_o^L \in G_k$, otherwise let $y_o = v_o^L$ and $x_k = v_o^U$.

Step 7. Construct an RPP P_i in D_i over E_{iC} from y_{i-1} to x_i for i=1,2,...,k according to the method described in section 2.6. Then construct an RPP T^* in D" over E_C:

$T^* = P_1 @x_1,y_1@P_2 @x_2,y_2@...@P_{k-1}@x_{k-1},y_{k-1}@P_k$,

which corresponds a COPT in D.

6.4. Examples

Example 6.1:

Consider the digraph D=(V,E) in Figure 3, we have

$Leave^U(v_o)$	= e_1	$Arrive^{U,L}(v_o)$	= e_2
$Leave^L(v_o)$	= e_5	$Arrive^U(v_o)$	= Ø
$Leave^U(v_1)$	= e_2	$Arrive^L(v_o)$	= e_4
$Leave^L(v_1)$	= e_3	$Arrive^{U,L}(v_1)$	= e_1
$Leave^U(v_2)$	= Ø	$Arrive^U(v_1)$	= Ø
$Leave^L(v_2)$	= e_4	$Arrive^L(v_1)$	= Ø
		$Arrive^{U,L}(v_2)$	= Ø
		$Arrive^U(v_2)$	= Ø
		$Arrive^L(v_2)$	= e_3, e_5

A COPT of D=(V,E) in Figure 3 is constructed by the algorithm as follows:

(Step 1). The duplex digraph D'=(V',E') is constructed as shown in Figure 6, where $V' = V^U \cup V^L \cup H$, $V^U = \{0^U, 1^U\}$, $V^L = \{0^L, 1^L, 2^L\}$, $H = \{U_{0,1}, U_{1,0}\}$, $E' = E_C \cup F$, $E_C = \{(0^U,U_{0,1}), (0^L,2^L), (1^U,U_{1,0}), (1^L,2^L), (2^L,0^L)\}$, and $F = \{ (U_{0,1},1^U), (U_{0,1},1^L), (U_{1,0}, 0^U), (U_{1,0}, 0^L)\}$. Edges in E_C are boldfaced.

(Step 2). The strongly connected components of D' are $\{0^U, 1^U, U_{0,1}, U_{1,0}\}$, $\{1^L\}$ and $\{0^L, 2^L\}$. Since k=3, Step 3 is skipped.

(Step 4). The strongly connected components of D' are ordered to obtain the ordered decomposition of D' which is $\{G_1, G_2, G_3\}$, where $G_1 = \{0^U, 1^U, U_{0,1}, U_{1,0}\}$, $G_2 = \{1^L\}$ and $G_3 = \{0^L, 2^L\}$. $D_i = (G_i, E_i)$, where $E_1 = \{(0^U,U_{0,1}), (1^U,U_{1,0}), (U_{0,1},1^U), (U_{1,0}, 0^U)\}$, $E_2 = \emptyset$ and $E_3 = \{(0^L,2^L), (2^L,0^L)\}$.

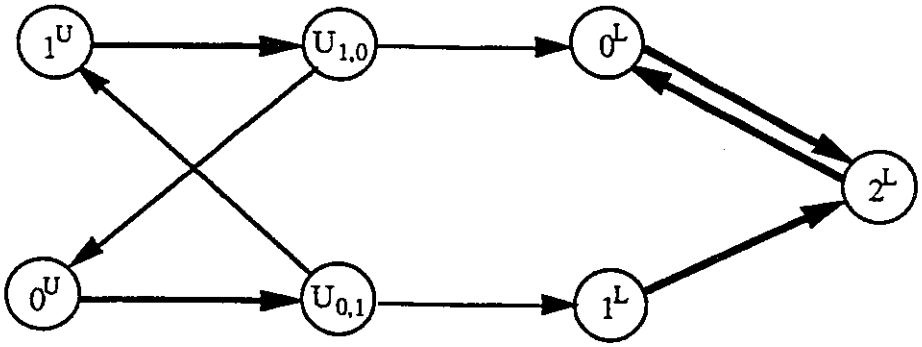


Figure 6. The Duplex Digraph D' of D in Figure 3

Boldfaced edges are edges in E_C.

(Ste 5). $\Delta(G_1,G_2) = \{(U_{0,1},1^L)\}$, $\Delta(G_2,G_3) = \{(1^L,2^L)\}$. There are no redundant edges to be removed from these sets.

(Step 6). Edges $(U_{0,1},1^L)$ and $(1^L,2^L)$ are links, where $(U_{0,1},1^L)$ is from G_1 to G_2 and $(1^L,2^L)$ is from G_2 to G_3.

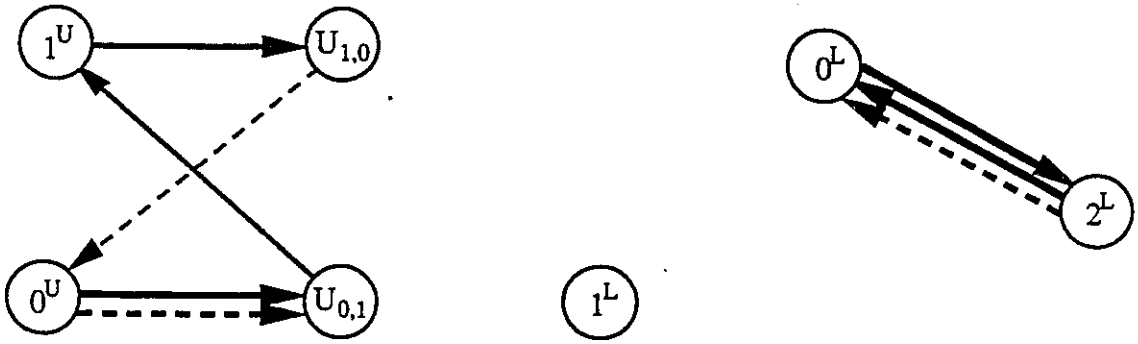


Figure 7. Rural Symmetric Augmentations of Subgraphs of D' in Figure 6.

(i) RSA D_1^* of $D[E_{1C}^*]$; (ii) RSA D_2^* of $D[E_{2C}]$; (iii) RSA D_3^* of $D[E_{3C}]$. Dashed edges are replicates.

(Step 7). Assume that $y_o = 0^U$ and $x_k = 0^L$. We have $E_{1C} = \{(1^U, U_{1,0}), (0^U, U_{0,1})\}$, $E_{2C} = \emptyset$, $E_{3C} = \{(0^L, 2^L), (2^L, 0^L), (1^L, 2^L)\}$. $D[E_{1C}]$ is not weakly connected, edge $(U_{0,1}, 1^U)$ is added to E_{1C} to obtain E_{1C}^*,

which are shown as solid edges in Figure 7(i). An RSA D_1* of $D[E_{1C}$*] for constructing a path P_1 from 0^U to $U_{0,1}$ is shown in Figure 7(i) and $P_1 = 0^U, U_{0,1}, 1^U, U_{1,0}, 0^U, U_{0,1}$; D_2 has no edge, hence $P_2 = \emptyset$; An RSA D_3* of $D[E_{3C}]$ for constructing a path P_3 from 2^L to 0^L is shown in Figure 7(iii) and $P_3 = 2^L, 0^L, 2^L, 0^L$. An RPP T* in D"=(V",E") over E_C is as follows, where V"=V' and $E" = E_1 \cup E_2 \cup E_3 \cup \{(U_{0,1},1^U), (U_{0,1},1^L), (U_{1,0}, 0^U)\}$:

$T^* = 0^U, U_{0,1}, 1^U, U_{1,0}, 0^U, U_{0,1}, 1^L, 2^L, 0^L, 2^L, 0^L$

which then gives rise to a COPT T in D, where

T = 0,1,0,1,2,0,2,0

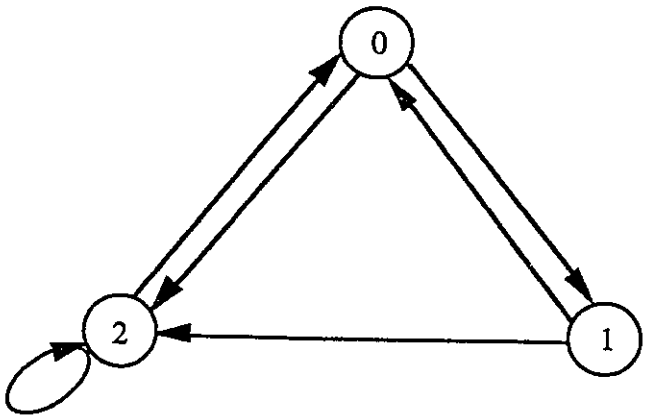
Note that since P_1 is in fact an RCPP in D_1 over E_{1C}, P_3 is also an RCPP in D_3 over E_{3C} and no edge is removed from either $\Delta(G_1,G_2)$ or $\Delta(G_2,G_3)$, T* is also an RCPP in D" and hence T is also a COCPT in D.

Example 6.2:

Figure 8 is another order-specified digraph D=(V,E), where V={0,1,2} and $E=\{e_1,e_2,e_3,e_4,e_5,e_6\}$. Using the algorithm RMatrix in section 5.1, we can obtain the edge reachability matrix of which all entries are 1's. That is, D is robustly biconnected. Hence there exists a COPT in D. In fact, TS = 0, 1, 0, 1, 2, 2, 0, 2, 2, 0 is a COCPT in D.

The steps of obtaining a COPT in D by the algorithm in section 6.1 is as follows.

(Step 1). The duplex digraph D'=(V',E') of D=(V,E) in Figure 8 is shown in Figure 9, where $V' = V^U \cup V^L \cup H$, $V^U = \{0^U, 1^U, 2^U\}$, $V^L = \{0^L, 1^L, 2^L\}$, $F = \{U_{0,1}, U_{2,0}, L_{2,2}\}$ and $E' = E_C \cup F$, $E_C = \{(0^U, U_{0,1}), (1^U, 0^U), (1^L, 2^L), (2^L, L_{2,2}), (2^U, U_{2,0}), (0^L, 2^L)\}$, $F = \{(U_{0,1}, 1^U), (U_{0,1}, 1^L), (L_{2,2}, 2^L), (L_{2,2}, 2^U), (U_{2,0}, 0^U), (U_{2,0}, 0^L)\}$.



Edge Types and Order Specifications:

$e_1 = (0,1)$, **type** $(e_1) = R^{IU}S^{IL}$, Eligible Successor: e_2, e_3

$e_2 = (1,0)$, **type** $(e_2) = R^{IU}S^{IU}$, Eligible Successor: e_1

$e_3 = (1,2)$, **type** $(e_3) = R^{IL}S^{IL}$, Eligible Successor: e_4

$e_4 = (2,2)$, **type** $(e_4) = R^{IL}S^{IU}$, Eligible Successor: e_4, e_5

$e_5 = (2,0)$, **type** $(e_5) = R^{IU}S^{IL}$, Eligible Successor: e_1, e_6

$e_6 = (0,2)$, **type** $(e_6) = R^{IL}S^{IL}$, Eligible Successor: e_4

Figure 8. A Robustly Biconnected Digraph D

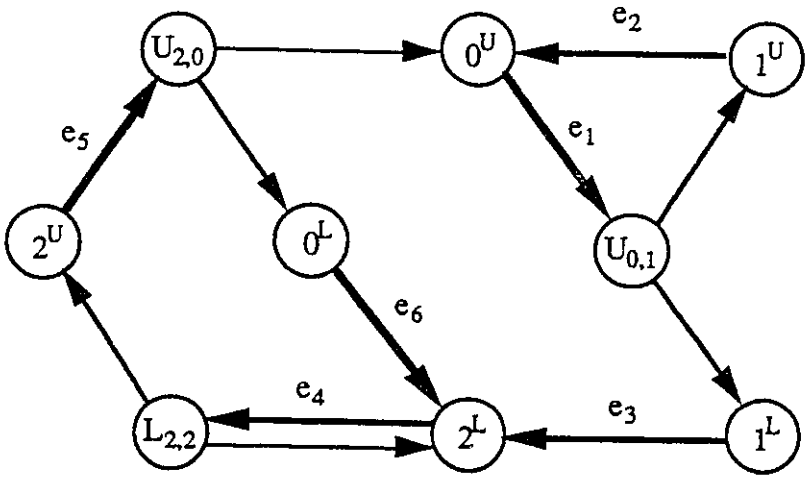


Figure 9. The Duplex Digraph D' of D in Figure 8

Edges E_C in are boldfaced.

(Step 2). Since D' is strongly connected, we continue with Step 3.

(Step 3). Since $D[E_C]$ is not weakly connected, Edges $(U_{0,1}, 1^L)$ and $(L_{2,2}, 2^U)$ are added to E_C to obtain E_C^* (shown as solid edges in Figure 10) such that $D[E_C^*]$ is weakly connected. An RSA D* of $D[E_C^*]$ is shown in Figure 10, where the dashed edges are replicates. Since D* is strongly connected, an Euler tour T* in D* is a RPT in D' over E_C^* and hence corresponds to a COPT T in D, where

$T^* = 0^U, U_{0,1}, 1^U, 0^U, U_{0,1}, 1^L, 2^L, L_{2,2}, 2^U, U_{2,0}, 0^L, 2^L, L_{2,2}, 2^U, U_{2,0}, 0^U$

and

T = 0, 1, 0, 1, 2, 2, 0, 2, 2, 0

Note that T* is in fact an RCPT in D' over E_C, T is thus a COCPT in D.

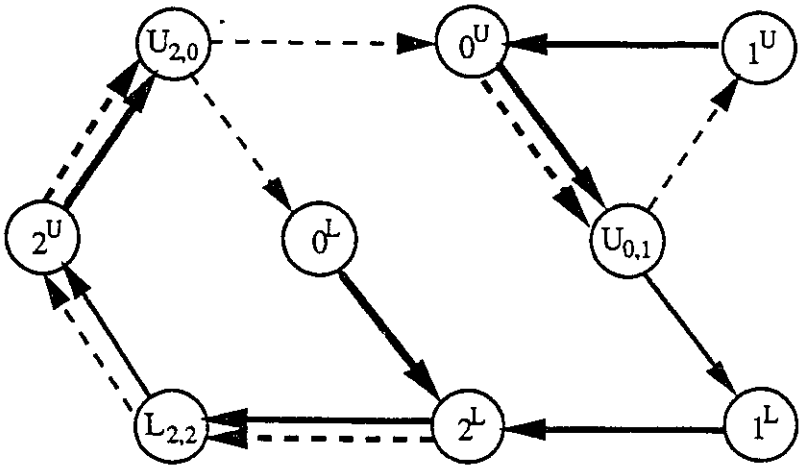


Figure 10. An RSA D* of D' in Figure 9

Solid edges are edges in E_C^*;

Dashed edges are replicates.

6.5. Complexity Analysis and Summary

The complexity of each of the above steps is given as follows:

Step 1: $\Theta(V'+E')$

Step 2: $\Theta(V'+E')$ [14, 38]

Step 3: $O(V'E'\log V')$

Step 4: $O(k\log k)$, or $O(V'\log V')$

Step 5: $O(E')$

Step 6: $O(E')$

Step 7: $O(V''E'' \log V'')$

Since $V'' = V'$, $E'' = \Theta(E')$, $V' = \Theta(V+E)$ and $E' = \Theta(E)$, the complexity of the algorithm is $O((V+E) E \log(V+E))$.

The following theorem summarizes the above results.

Theorem 6.1. Given that an order-specified digraph D=(V,E) with specified vertex v_o is strongly biconnected, a COPT T in D starting at v_o can be constructed in polynomial time ($O((V+E) E \log(V+E))$).

The tour T might not necessarily a COCPT in D. However, the above method gives a reasonably short test sequence with $O(qV^2)$ [16] as the upper bound on length, where q is the number of input symbols of the FSM. The COPT T in D gives rise to a synchronizable test sequence which checks output functions of the FSM represented by D.

7. SYNCHRONIZABLE TEST SEQUENCES FOR OUTPUT & TRANSFER FUNCTIONS

In chapter 6, we developed a test sequence for an FSM represented by digraph D=(V,E) by constructing a COPT in D. However the test sequence thus obtained can only detect output errors. The state that the FSM is expected to reach after the execution of a transition is not checked by that test sequence. To test the correctness of transfer functions as well as output functions, characterizing sets, distinguishing sequences, or unique input/output sequences are used to derive test sequences which check both output and transfer functions of an FSM. In the subsequent sections, we will describe how the W-method, the D-method and the UIO-method can be extended to construct synchronizable test sequences for an FSM.

In the following sections, $\mathbf{I}^U$ and $\mathbf{I}^L$ denote the inputs from the upper tester and the lower tester, respectively.

7.1. Extensions of the W-Method

An approach of using characterizing sets for the construction of a synchronizable test sequence is proposed in [34]. This approach ignores the synchronization problem during the construction of a W-set (**W**) and a testing tree (**P**) for the given FSM. All subsequences of **P@W** are then checked for synchronization problems. If a subsequence of **P@W** has synchronization problems, a different W-set or a different testing tree is constructed and used. This process is repeated until a W-set and a testing tree are found such that all subsequences of **P@W** have no synchronization problem.

Our approach described below avoids the exhaustive search for a W-set and a testing tree by constructing two synchronizable W-sets and a synchronizable testing tree.

Therefore, in our approach, replacement of W-sets or testing trees is not needed in the subsequent construction of a synchronizable test sequence.

A set of input sequences is called a *synchronizable characterizing set* or SW-set if

(i) the transition sequences produced by the FSM from any state in response to the input sequences are synchronizable, and

(ii) the set can distinguish the behaviours of every pair of states in the FSM.

Even though we have a strong assumption that any input sequence in an SW-set is applicable to each state, a synchronization problem can still arise when a certain type of transition is followed by the application of an input sequence in the SW-set. For example, if such a sequence is initiated by the upper tester (*i.e.*, the first input symbol is $\mathbf{I}^{U}$), any transition of type of R^{IL} or $R^{IL}S^{IL}$ preceding the application of the input sequence will cause synchronization problem. To solve such a problem, we need two SW-sets, one contains input sequences initiated by the upper tester ($\mathbf{SW}^{U}$) and the other one contains input sequences initiated by the lower tester ($\mathbf{SW}^{L}$). Any transition of the FSM can then be checked by applying input sequences from either $\mathbf{SW}^{U}$ or $\mathbf{SW}^{L}$.

In order for an FSM to have an $\mathbf{SW}^{U}$, the FSM should not contain indistinguishable states with respect to the upper tester. Two states v_i and v_j in an FSM are *indistinguishable with respect to the upper tester* if

(i) any synchronizable input sequence initiated by the upper tester for state v_i is also a synchronizable input sequence for state v_j, and vice versa,

(ii) the FSM produces identical output sequences from either state in response to any synchronizable input sequence initiated by the upper tester.

Indistinguishability with respect to the lower tester is similarly defined.

An FSM is called *strongly distinguishable* if it contains no indistinguishable states with respect to either the upper or the lower tester. Obviously, a strongly distinguishable FSM is also minimal.

In the W-method, we assume that the FSM is strongly distinguishable, strongly biconnected, completely specified, and possesses two SW-sets, $\mathbf{SW}^U = \{w^U_1, w^U_2, ..., w^U_g\}$ and $\mathbf{SW}^L = \{w^L_1, w^L_2, ..., w^L_h\}$. We also assume that the FSM has a reset input ri^U initiated by the upper tester and a reset input ri^L initiated by the lower tester, where both reset inputs can be applied at each state of the FSM and can be followed by any input from the upper or lower tester without causing synchronization problems, *i.e.*, $\mathbf{type}(ri^U) = R^{IU}S^{IU,L}$ and $\mathbf{type}(ri^L) = R^{IL}S^{IL,U}$.

7.1.1. Extended W-Method for a Strongly Distinguishable FSM

Given an FSM satisfying the assumptions stated in the previous subsection and represented by a digraph D=(V,E), a minimal $\mathbf{SW}^U$ can be computed by applying multiple-experiments in the same way as a W-set is constructed [20] except that each input sequence should start with an input in $\mathbf{I}^U$ and the consecutive transitions should not cause a synchronization problem. A minimal $\mathbf{SW}^U$ is similarly constructed.

To obtain a synchronizable test sequence from the SW-sets, a synchronizable testing-tree (ST-tree), similar to a testing-tree described in section 3.2 [20], is constructed such that the root of the tree is the start state of the FSM, each branch of the ST-tree represents a transition in the FSM, each node of the ST-tree is a state of the FSM, each transition of the FSM appears in the ST-tree exactly once and each pair of consecutive transitions in the ST-tree is synchronizable. From the ST-tree, we can derive a set of paths $\mathbf{P} = \{p_1, p_2, ..., p_n\}$ in the ST-tree, such that

(i) each p_i is a minimum-length input sequence which takes the FSM from its start state to a particular state from which the transition to be tested next emanates;

(ii) the transition sequence produced by the FSM in response to each p_i is synchronizable.

The empty sequence is also considered a member of **P**. For each p_i, we define set $\mathbf{SW}(p_i) = \{w_j \in \mathbf{SW}^U \cup \mathbf{SW}^L$: the transition sequence produced by the FSM in response to p_i @ w_j is synchronizable$\}$. Let *Last_TR*(p_i) stands for the last transition of the transition sequence produced by the FSM in response to p_i. We have, in fact, that if **type**(Last_TR$(p_i)) \in \{R^{IU}, R^{IU}S^{IU}\}$ then $\mathbf{SW}(p_i) = \mathbf{SW}^U$; if **type**(Last_TR$(p_i)) \in \{R^{IL}, R^{IL}S^{IL}\}$ then $\mathbf{SW}(p_i) = \mathbf{SW}^L$; if **type**(Last_TR$(p_i)) \in \{R^{IL}S^{IU}, R^{IU}S^{IL}, R^{IL}S^{IL,U}, R^{IU}S^{IU,L}\}$ then $\mathbf{SW}(p_i) = \mathbf{SW}^U \cup \mathbf{SW}^L$. Let $|i_s|$ be the number of input symbols of the input sequence i_s, and $|I^U \cup I^L|$ be $\sum\{|i_s|: i_s \in I^U \cup I^L\}$. We define $\mathbf{SW}(p_i)^*$ as follows:

(i) $\mathbf{SW}(p_i)^* = \mathbf{SW}(p_i)$ if $\mathbf{SW}(p_i) = \mathbf{SW}^U$ or $\mathbf{SW}(p_i) = \mathbf{SW}^L$;

(ii) $\mathbf{SW}(p_i)^* = \mathbf{SW}^U$ if $\mathbf{SW}(p_i) = \mathbf{SW}^U \cup \mathbf{SW}^L$ and $|\mathbf{SW}^U| \leq |\mathbf{SW}^L|$;

(iii) $\mathbf{SW}(p_i)^* = \mathbf{SW}^L$ if $\mathbf{SW}(p_i) = \mathbf{SW}^U \cup \mathbf{SW}^L$ and $|\mathbf{SW}^U| > |\mathbf{SW}^L|$.

A synchronizable test sequence (STS) which checks output and transfer functions of the FSM can then be constructed as follows.

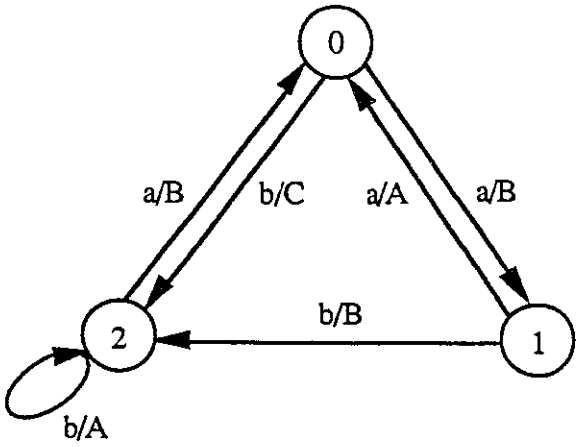
$$\mathrm{STS} = \mathop{@}_{i=1}^{n} \; \mathop{@}_{w_j \in \mathbf{SW}(p_i)^*} (p_i @ w_j @ r_i)$$

where, in each subsequence p_i @ w_j @ r_i, reset input r_i stands for r_i^U if Last_TR$(p_i$ @ $w_j)$ is of type R^{IU}, $R^{IU}S^{IU}$, $R^{IL}S^{IU}$, $R^{IL}S^{IL,U}$, or $R^{IU}S^{IU,L}$, otherwise r_i stands for r_i^L.

The complexity of constructing **P** and **SW** is $O(qm^2)$ [15] with q input symbols and m states in an FSM. Hence the complexity of the W-method is $O(qm^2)$.

7.1.2. An Example

Consider the digraph D in Figure 11, which is the same as that in Figure 8 with explicit input/output labelling of each edge. We have $\mathbf{I}^U = \{a\}$ and $\mathbf{I}^L = \{b\}$.



Edge Types and Order Specifications:

$e_1 = (0,1; a/B)$, **type** $(e_1) = R^{IU}S^{IL}$, Eligible Successor: e_2, e_3

$e_2 = (1,0; a/A)$, **type** $(e_2) = R^{IU}S^{IU}$, Eligible Successor: e_1

$e_3 = (1,2; b/B)$, **type** $(e_3) = R^{IL}S^{IL}$, Eligible Successor: e_4

$e_4 = (2,2; b/A)$, **type** $(e_4) = R^{IL}S^{IU}$, Eligible Successor: e_4, e_5

$e_5 = (2,0; a/B)$, **type** $(e_5) = R^{IU}S^{IL}$, Eligible Successor: e_1, e_6

$e_6 = (0,2; b/C)$, **type** $(e_6) = R^{IL}S^{IL}$, Eligible Successor: e_4

Figure 11. A Digraph D Representing a Strongly Distinguishable FSM

It can be verified that the digraph D in Figure 11 is strongly distinguishable. By applying a multiple-experiment on the FSM represented by D, we obtain two SW-sets: $\mathbf{SW}^U = \{a, ab\}$ and $\mathbf{SW}^L = \{b\}$. An ST-tree is shown in Figure 12. From ST-tree, we obtain the set $\mathbf{P} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$, where $p_1 = \emptyset$, $p_2 = a$, $p_3 = aa$, $p_4 = ab$, $p_5 = b$, $p_6 = bb$, $p_7 = bba$ and $\mathbf{SW}(p_1) = \mathbf{SW}^U \cup \mathbf{SW}^L$, $\mathbf{SW}(p_2) = \mathbf{SW}^U \cup \mathbf{SW}^L$, $\mathbf{SW}(p_3) = \mathbf{SW}^U$, $\mathbf{SW}(p_4) = \mathbf{SW}^L$, $\mathbf{SW}(p_5) = \mathbf{SW}^L$, $\mathbf{SW}(p_6) = \mathbf{SW}^U \cup \mathbf{SW}^L$, $\mathbf{SW}(p_7) =$

$SW^U \cup SW^L$. Since $|SW^U| > |SW^L|$, we have $SW(p_1)^* = SW(p_2)^* = SW(p_4)^* = SW(p_5)^* = SW(p_6)^* = SW(p_7)^* = \{b\}$ and $SW(p_3)^* = \{a, ab\}$. Thus a synchronizable test sequence is as follows:

b ri ab ri aaa ri aaab ri abb ri bb ri bbb ri bbab ri

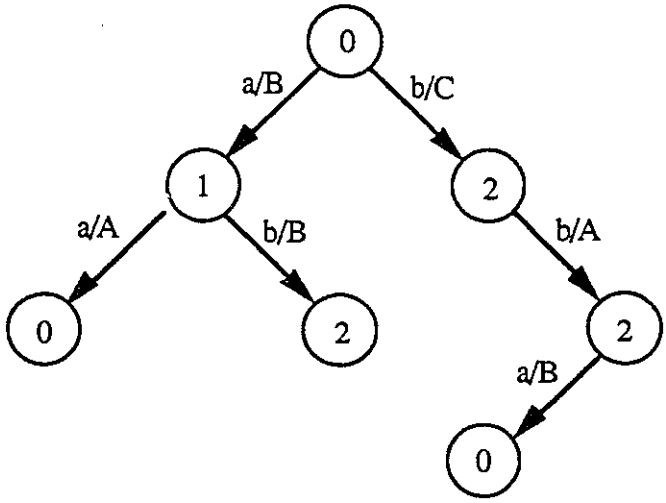


Figure 12. An ST-tree for D in Figure 11

7.1.3. Extended W-Method for a Distinguishable FSM

It can be verified that the indistinguishability relation between states with respect to upper (lower) tester is an equivalence relation for an FSM. Given an FSM which is not strongly distinguishable, the states of the FSM can be partitioned into $\{US_i: i=1,2,..,s\}$, the sets of indistinguishable states of the FSM with respect to upper tester. That is $V = \cup_{i=1,2,..,s} US_i$ and $US_i \cap US_j = \emptyset$ for $i \neq j$. Similarly, we have $V = \cup_{i=1,2,..,t} LS_i$ and $LS_i \cap LS_j = \emptyset$ for $i \neq j$, where $\{LS_i: i=1,2,..,t\}$ are sets of indistinguishable states of the FSM with respect to lower tester. The FSM is *distinguishable* if the intersection $US_i \cap LS_j$ contains at most one state of the FSM for i=1,2,..,s and j=1,2,..,t. It is obvious that a strongly distinguishable FSM is a distinguishable FSM and a distinguishable FSM is a minimal FSM.

In an FSM, a set of input sequences is called a *sub-synchronizable characterizing set with respect to the upper tester*, denoted by $\mathbf{SSW^U}$ if

(i) the transition sequences produced by the FSM from any state in response to the input sequences are synchronizable, and

(ii) the set can distinguish the behaviours of every pair of states s and s' where $s \in US_i$ and $s' \in US_j$ for $i \neq j$.

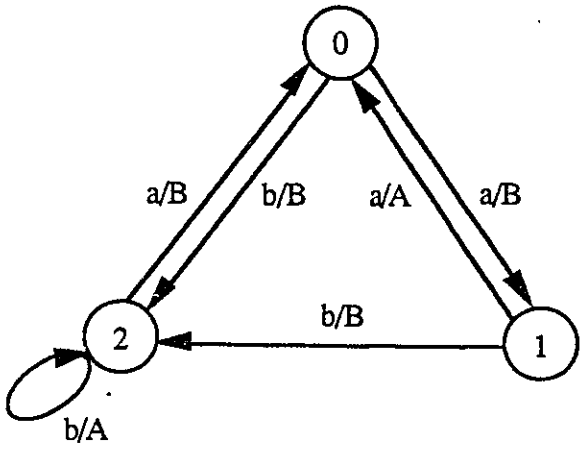
A *sub-synchronizable characterizing set with respect to the lower tester*, denoted by $\mathbf{SSW^L}$, is similarly defined. Obviously, if there exists an $\mathbf{SSW^U}$ and an $\mathbf{SSW^L}$ for a given distinguishable FSM, the pair of output strings (o^U, o^L) produced by the FSM in response to the pair of $(\mathbf{SSW^U}, \mathbf{SSW^L})$ is different for each starting state. Hence the pair $(\mathbf{SSW^U}, \mathbf{SSW^L})$ can be used in place of $(\mathbf{SW^U}, \mathbf{SW^L})$ when the FSM is not strongly distinguishable.

Given a distinguishable FSM, a minimal $\mathbf{SSW^U}$ can be computed in the same way as an $\mathbf{SW^U}$ is computed except that sets $\{US_i: i=1,2,..,s\}$ are to be distinguished instead of vertices. A minimal $\mathbf{SSW^L}$ is similarly computed. **P** is also similarly derived. For each p_i, we define $\mathbf{SSW}(p_i) = \{w_j \in \mathbf{SSW^U} \cup \mathbf{SSW^L}$: the transition sequence produced by the FSM in response to $p_i @ w_j$ is synchronizable$\}$.

Once a minimal $\mathbf{SSW^U}$ and a minimal $\mathbf{SSW^L}$ are computed, a synchronizable test sequence is constructed by the procedure described in section 7.1.1 provided that $\mathbf{SW^U}$ and $\mathbf{SW^L}$ are replaced by $\mathbf{SSW^U}$ and $\mathbf{SSW^L}$ respectively, and that $\mathbf{SW}(p_i)^*$ is replaced by $\mathbf{SSW}(p_i)$ for i=1,2,...,n. Note that, to check each transition in a strongly distinguishable FSM, input sequences from either $\mathbf{SSW^U}$ or $\mathbf{SSW^L}$ but not both are applied to the transition. However, when an FSM is only distinguishable, input sequences from both $\mathbf{SSW^U}$ and $\mathbf{SSW^L}$ are applied to the transition, this is indicated by the fact that $\mathbf{SW}(p_i)^*$ is replaced by $\mathbf{SSW}(p_i)$ for i=1,2,...,n.

7.1.4. An Example

Consider the digraph D in Figure 13, which is the same as that in Figure 11 except that the label of e_6 is "b/B" instead of "b/C". The digraph in Figure 13 is not strongly distinguishable since, for instance, state 0 and state 1 are indistinguishable with respect to lower tester. The only input symbol from the lower tester is "b" which brings both state 0 and state 1 to state 2 with the same output symbol "B".



Edge Types and Order Specifications:

e_1 = (0,1; a/B), **type** (e_1) = $R^{IU}S^{IL}$, Eligible Successor: e_2, e_3

e_2 = (1,0; a/A), **type** (e_2) = $R^{IU}S^{IU}$, Eligible Successor: e_1

e_3 = (1,2; b/B), **type** (e_3) = $R^{IL}S^{IL}$, Eligible Successor: e_4

e_4 = (2,2; b/A), **type** (e_4) = $R^{IL}S^{IU}$, Eligible Successor: e_4, e_5

e_5 = (2,0; a/B), **type** (e_5) = $R^{IU}S^{IL}$, Eligible Successor: e_1, e_6

e_6 = (0,2; b/B), **type** (e_6) = $R^{IL}S^{IL}$, Eligible Successor: e_4

Figure 13. A Digraph D Representing a Non-Strongly Distinguishable FSM

The states can be partitioned into sets of indistinguishable states with respect to the upper tester as {{0}, {1}, {2}} and into sets of indistinguishable states with respect to the lower tester as {{0,1}, {2}}. Since {0} ∩ {0,1} = {0}, {0} ∩ {2} = Ø, {1} ∩ {0,1} = {1}, {1} ∩ {2} = Ø, {2} ∩ {0,1} = Ø, {2} ∩ {2} = {2}, the FSM represented by D is distinguishable. A pair of ($\mathbf{SSW^U}$, $\mathbf{SSW^L}$) is ({a, aa}, {b}). From section 7.1.2, p_1 = Ø, p_2 = a, p_3 = aa, p_4 = ab, p_5 = b, p_6 = bb, p_7 = bba. We have $\mathbf{SSW}(p_1) = \mathbf{SSW^U} \cup \mathbf{SSW^L}$, $\mathbf{SSW}(p_2) = \mathbf{SSW^U} \cup \mathbf{SSW^L}$, $\mathbf{SSW}(p_3) = \mathbf{SSW^U}$, $\mathbf{SSW}(p_4) = \mathbf{SSW^L}$, $\mathbf{SSW}(p_5) = \mathbf{SSW^L}$, $\mathbf{SSW}(p_6) = \mathbf{SSW^U} \cup \mathbf{SSW^L}$, $\mathbf{SSW}(p_7) = \mathbf{SSW^U} \cup \mathbf{SSW^L}$. A synchronizable test sequence is then as follows:

a ri aa ri b ri aa ri aaa ri ab ri aaa ri aaaa ri abb ri

bb ri bba ri bbaa ri bbb ri bbaa ri bbaaa ri bbab ri

7.2. Extensions of the D-Method

An approach for using distinguishing sequences in the construction of a synchronizable test sequence is proposed in [34]. This approach ignores the synchronization problem during the construction of a DS for the given FSM. For each transition e, the transition segment e@DS is then checked for synchronization problems during the construction of a test sequence. If a transition segment e@DS has synchronization problems, a different DS is generated to replace the previous one. The process is repeated until either no different DS can be generated or a DS is found such that all transition segments have no synchronization problem. If such a DS is found, all transition segments are linked by transfer functions and checked for synchronization as a whole. In the case of synchronization problem, a different order of the subsequence and/or different transfer functions are considered.

Our approach described below again avoids the exhaustive search for a DS, a synchronizable ordering of transition segments, and proper transfer functions. Instead, we

construct two synchronizable DS's and employ the duplex digraph method to generate a synchronizable test sequence which includes all transition segments.

In an FSM, an input string is said to be a *distinguishing sequence* (DS) [26] if the output string produced by the FSM in response to the DS is different for each starting state. The DS is used to check whether the FSM reaches the expected state after the execution of each transition.

An input string s is said to be a *synchronizable distinguishing sequence* (SDS) in an FSM represented by an order-specified digraph if
(i) the transition sequence ts produced by the FSM in response to s is synchronizable and
(ii) the output string of ts is different for each starting state.

The argument in the section 7.1 about the synchronization problem associated with the application of the input sequence in the SW-set applies similarly to the SDS and suggests that we need two SDS sequences, one initiated by the upper tester (SDS^U) and one by the lower tester (SDS^L). Any transition of the FSM can then be checked by applying either SDS^U or SDS^L.

To apply the D-method to an FSM represented by an order-specified digraph D=(V,E), we assume that the FSM is strongly distinguishable, robustly biconnected, completely specified and possesses two synchronizable distinguishing sequences, SDS^U and SDS^L. We also assume that the FSM has reset inputs ri^U and ri^L such that **type**(ri^U) = $R^{IU}S^{IU,L}$ and **type**(ri^L) = $R^{IL}S^{IL,U}$.

7.2.1. Extended D-Method for a Strongly Distinguishable FSM

Given a strongly distinguishable FSM represented by a digraph D=(V,E), an SDS^U of minimum-length can be computed by constructing an *upper distinguishing tree* from digraph D=(V,E), which is similar to the distinguishing tree described by Kohavi [26]

except that only those input symbols which will not cause any synchronization problem are applied to the uncertainty vectors [26] at each level of the tree. An uncertainty at level 0 is an initial uncertainty vector which is the subset of V containing the start state. An uncertainty at level i (i>0) is a collection of states which are end states after the application of a specific input to the states in an uncertainty vector at level i-1.

An SDS^L of minimum-length can be computed similarly by constructing a *lower distinguishing tree* from digraph D=(V,E).

To obtain a synchronizable test sequence by using SDS^U and SDS^L sequences, we construct the duplex digraph D'=(V',E') from digraph D=(V,E) by the method described in chapter 6 with v_o^U or v_o^L as the start vertex. From Lemma 6.2, the robust biconnectivity of D ensures that D' is strongly connected. A graph traversal technique [1, 29] for constructing an RPT can be applied in combination with SDS^U and SDS^L sequences to obtain an RPT T* of a strongly connected digraph $D^{\sim}=(V^{\sim},E^{\sim})$ over a set of core edges E_C, where $D^{\sim}=(V^{\sim},E^{\sim})$ is derived from D'=(V',E'). The RPT T* gives rise to a COPT T in digraph D=(V,E) and hence a synchronizable test sequence for the FSM represented by D which checks both output and transfer functions of the FSM. The following steps describe our method:

Step 1: Construct the duplex digraph D'=(V',E') from digraph D=(V,E) by the method described in section 6.1 with v_o^U or v_o^L as the start state, where $V' = V^U \cup V^L \cup H$ and $E' = E_C \cup F$. Let each edge e' in E_C has the label of the edge e in E which gives rise to e' and each child edge in F has the label of "null/null". Each edge in E_C is given cost one and each edge in F is given cost zero.

Step 2: Construct an i@SDS-diagram as $D^{\wedge}=(V^{\wedge},E^{\wedge})$, such that $V^{\wedge} = V' \cup Z = V^U \cup V^L \cup H \cup Z$ and $E^{\wedge} = E' \cup L_C \cup L = E_C \cup F \cup L_C \cup L$, where Z, L_C and L are constructed as follows:

For each edge $e = (v_p, v_q: i_1/o_1) \in E$, let $e' = (x_j, x_k: i_1/o_1) \in E_C$ be the corresponding inherent or parent edge. We have the following two cases:

Case 1: e' is an inherent edge, then $x_k \in V^U \cup V^L$. Let e_{last} be the last edge in E in response to the input sequence i_1@SDS applied at state v_p, where SDS stands for SDS^U if $x_k \in V^U$, otherwise, SDS stands for SDS^L. Let $(x_s, x_t; i_k/o_k) \in E_C$ be the inherent or parent edge corresponding to e_{last}. We create an edge $(x_j, x_t; i_1@SDS)$ in L_C and let the edge has cost Length(SDS)+1, where Length(SDS) stands for the length of SDS.

Case 2: e' is a parent edge, then $x_k \in H$, *i.e.*, $x_k = U_{p,q}$ or $x_k = L_{p,q}$. Let $e_{U\text{-}last}$ and $e_{L\text{-}last}$ be the last edges in E in response to the input sequences $i_1@SDS^U$ and $i_1@SDS^L$ applied at state v_p, respectively. Let $(x_s, x_t; i_h/o_h)$, $(x_u, x_v; i_k/o_k) \in E_C$ be the inherent edge or the parent edge corresponding to $e_{U\text{-}last}$ and $e_{L\text{-}last}$, respectively.

If $x_k = U_{p,q}$, we create a vertex $X_{p,q}$ in Z; an edge $(x_j, X_{p,q}; i_1/o_1)$ in L_C; an edge $(X_{p,q}, x_t; SDS^U)$ in L; and an edge $(X_{p,q}, x_v; SDS^L)$ in L. Let edge $(x_j, X_{p,q}; i_1/o_1)$ has cost one and edges $(X_{p,q}, x_t; SDS^U)$ and $(X_{p,q}, x_v; SDS^L)$ have cost $Length(SDS^U)$ and $Length(SDS^L)$, respectively.

If $x_k = L_{p,q}$, we create a vertex $Y_{p,q}$ in Z; an edge $(x_j, Y_{p,q}; i_1/o_1)$ in L_C; an edge $(Y_{p,q}, x_t; SDS^U)$ in L; and an edge $(Y_{p,q}, x_v; SDS^L)$ in L. Let edge $(x_j, Y_{p,q}; i_1/o_1)$ has cost one and edges $(Y_{p,q}\ x_t; SDS^U)$ and $(Y_{p,q}, x_v; SDS^L)$ have cost $Length(SDS^U)$ and $Length(SDS^L)$, respectively.

Since D' is strongly connected, due to the manner D^ is obtained, D^ will be strongly connected.

Step 3: The edge-induced subgraph $D[L_C]$ might not be a spanning subgraph of D^, since some vertex in H might not be incident with any edge in L_C. We create a digraph

$D^{\sim}=(V^{\sim},E^{\sim})$ from $D^{\wedge}=(V^{\wedge},E^{\wedge})$ by removing those vertices which are not incident with any edge in L_C as follows:

1) Every vertex in $V^{\wedge}$ which is incident with some edge in L_C is included in $V^{\sim}$;

2) Every edge in $E^{\wedge}$ with both head vertex and tail vertex in $V^{\sim}$ is included in $E^{\sim}$.

3) For each vertex x in $V^{\wedge}$ but not in $V^{\sim}$, we know that $v \in H$. There are a pair of child edges (x,y; null/null) and (x,z; null/null) incident with x and they are the only edges leaving vertex x. For each incoming edge (w,x; label), we create two edges (w,y; label) and (w,z; label) in $E^{\sim}$ and let the costs of these two edges be the cost of edge (w,x; label).

Thus, we have that $D^{\sim}$ is also strongly connected, $E^{\sim} \supseteq L_C$ and $D[L_C]$ is a spanning subgraph of $D^{\sim}$. Now finding a synchronizable test sequence which checks every transition by either SDS^U or SDS^L is equivalent to finding a tour in $D^{\sim}$ which includes every edge in L_C at least once for the digraph $D^{\sim}=(V^{\sim}, E^{\sim})$. Finding such a tour is reduced to finding an RPT of $D^{\sim}$ over L_C [1].

Step 4: If $D[L_C]$ is not weakly connected, edges from $E^{\sim}$ are added to L_C to obtain L_C^* until $D[L_C^*]$ is weakly connected.

Step 5: Construct an RSA D* of D~ over L_C^* and find an Euler tour T* in D*, which is then an RPT T* of $D^{\sim}$ over L_C^*, where each edge in T* is represented by the label of the edge. By removing the label of "null/null" in T* will give rise to a synchronizable test sequence T in D which checks output and transfer functions. Note that if $D[L_C]$ is itself weakly connected, T* is also an RCPT in $D^{\sim}$ over L_C and T is then a minimum-length synchronizable test sequence.

The complexity analysis of the method is as follows.

(In Step 1): $O(V'+E')$, or $O(V+E)$ since $|V'| = O(V+E_C)$, $|E_C| = O(E)$ and $|E'| = \Theta(E)$,.

(In Step 2): $O(V^{\wedge}+E^{\wedge})$, or $O(V+E)$ since $|V^{\wedge}| = O(V'+E_C)$ and $|E^{\wedge}| = \Theta(E')$.

(In Step 3): $O(V^{\sim}+E^{\sim})$, or $O(V+E)$ since $|V^{\sim}| = \Theta(V^{\wedge})$ and $|E^{\sim}| = \Theta(E^{\wedge})$.

(In Step 4): $O(E^{\sim})$, or $O(E)$.

(In Step 5): $O(V^{\sim}E^{\sim}\log V^{\sim})$ or $O((V+E)\ E\ \log(V+E))$ since $|E^{\sim}| = \Theta(E)$ and $|V^{\sim}| = O(V+E)$.

Hence, given SDS^U and SDS^L for an FSM represented by D=(V,E), the complexity of constructing a synchronizable test sequence for the FSM is $O((V+E)\ E\ \log(V+E))$, or $O((m+n)\ n\ \log(m+n))$, where m is the number of states and n is the number of transitions in the FSM.

7.2.2. An Example

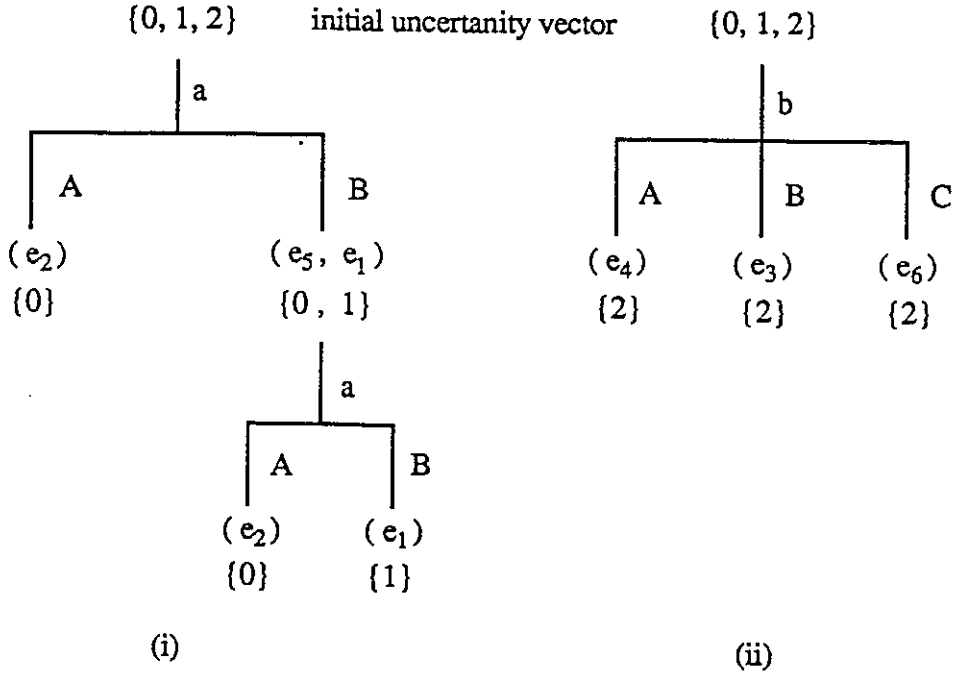


Figure 14. Distinguishing Trees for D in Figure 9

(i) An upper distinguishing tree.

(ii) A lower distinguishing tree.

Consider the digraph D=(V,E) in Figure 11. The upper distinguishing tree is shown in Figure 14(i) and the lower distinguishing tree is shown in Figure 14(ii). For the convenience of searching for subsequent synchronizable input symbol from a state, the edge by which the state is reached is shown right above the state.

From the trees, an SDS^U can be obtained as "aa" and an SDS^L as "b". The duplex digraph D'=(V',E') of D is shown in Figure 15.

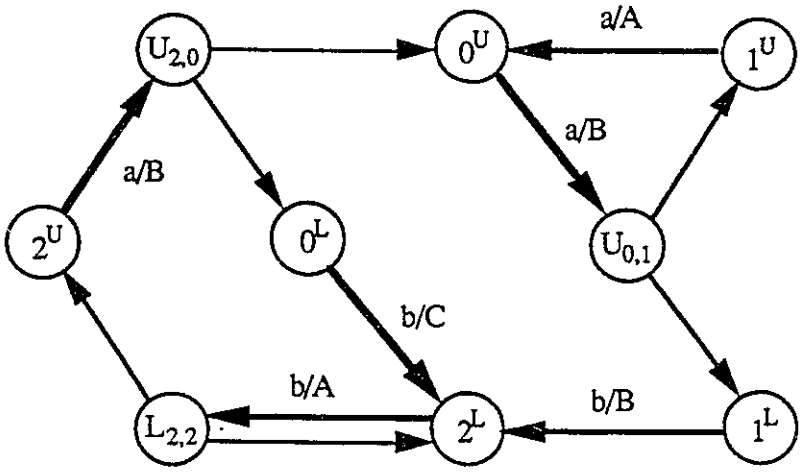


Figure 15. The Duplex Digraph D'=(V',E') of D in Figure 11

Unlabelled edges have "null/null" labels.

Boldfaced edges are edges in E_C.

The i@SDS-diagram D^=(V^,E^) for D is shown in Figure 16. Edges in L_C are boldfaced, where L_C = {(0^U, $X_{0,1}$; a/B), (0^L, $L_{2,2}$; b@b), (1^U, 0^U; a@aa), (1^L, $L_{2,2}$; b@b), (2^U, $X_{2,0}$; a/B), (2^L, $Y_{2,2}$; b/A)}. D[L_C] is not a spanning subgraph of D^, since vertices $U_{0,1}$ and $U_{2,0}$ are not incident with any edge in L_C. Thus, we create a digraph D~=(V~,E~) by following the procedure in step 3.

The digraph D~=(V~,E~) is shown in Figure 17. Since D[L_C] is not weakly connected in D~, we add edges to L_C to obtain L_C*, where L_C* = L_C ∪ {(0^U, 1^L; a/B),

$(X_{0,1}, 2^L; b), (2^U, 0^L; a/B)\}$ (Figure 18). $D[L_C^*]$ is weakly connected. The RSA D^* of $D^\sim$ over L_C^* is shown in Figure 19. An Euler tour T^* of D^* starting at 0^U is

a/B b b/A aa a@aa a/B b@b null/null a/B b@b null/null a/B aa

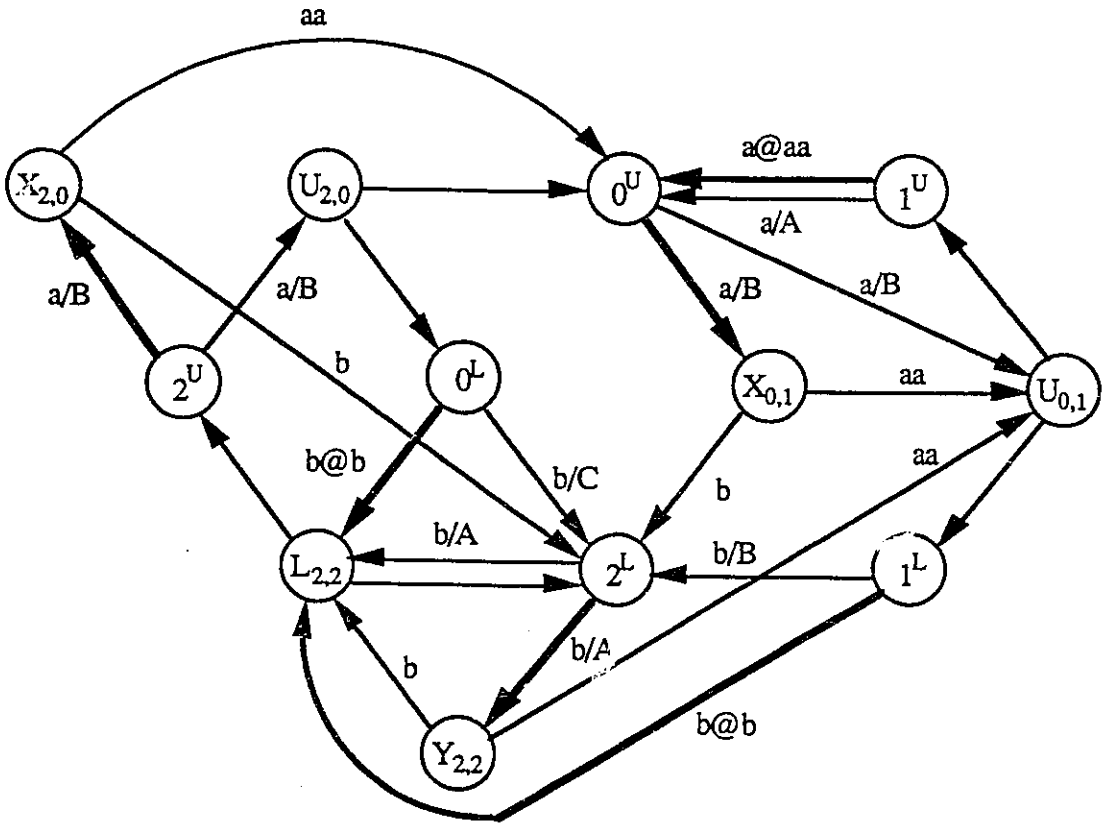


Figure 16. An i@SDS-diagram $D^\wedge$ for D' in Figure 15

Boldfaced edges are edges in L_C.

By removing "null/null" labels, we obtain

a/B b b/A aa a@aa a/B b@b a/B b@b a/B aa

Hence, a synchronizable test sequence T for the FSM represented by D=(V,E) is:

a@b b@aa a@aa a b@b a b@b a@aa

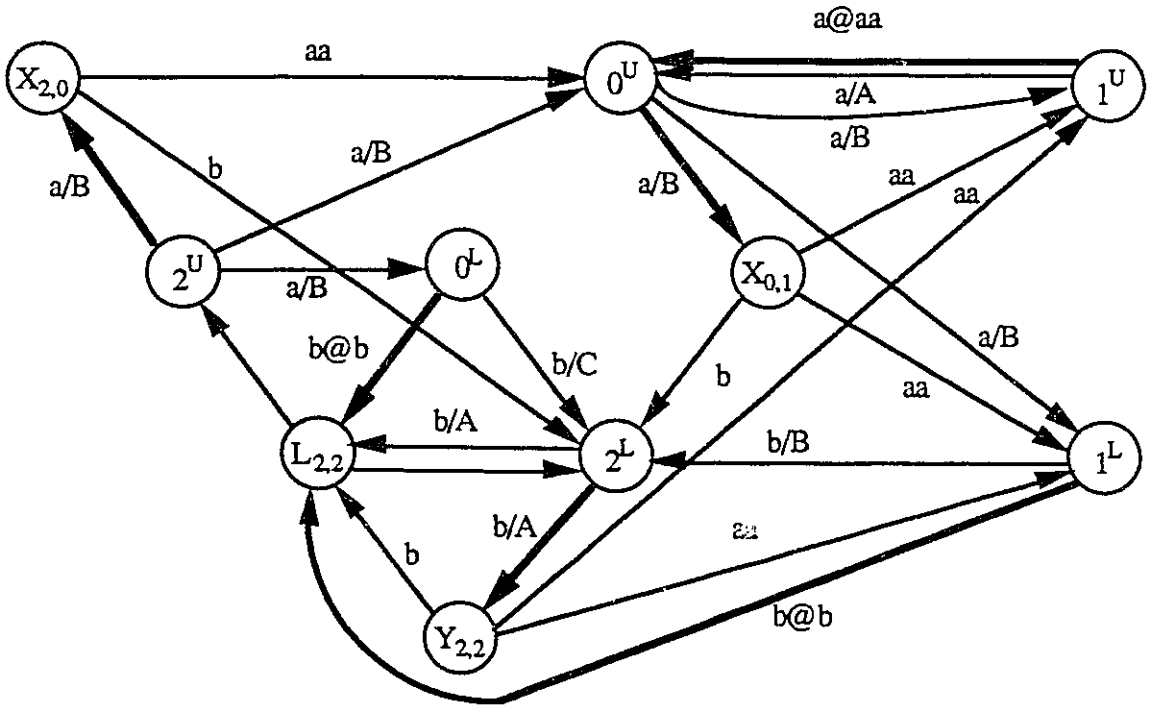


Figure 17. A Digraph $D^{\sim}$ Obtained from $D^{\wedge}$ in Figure 16

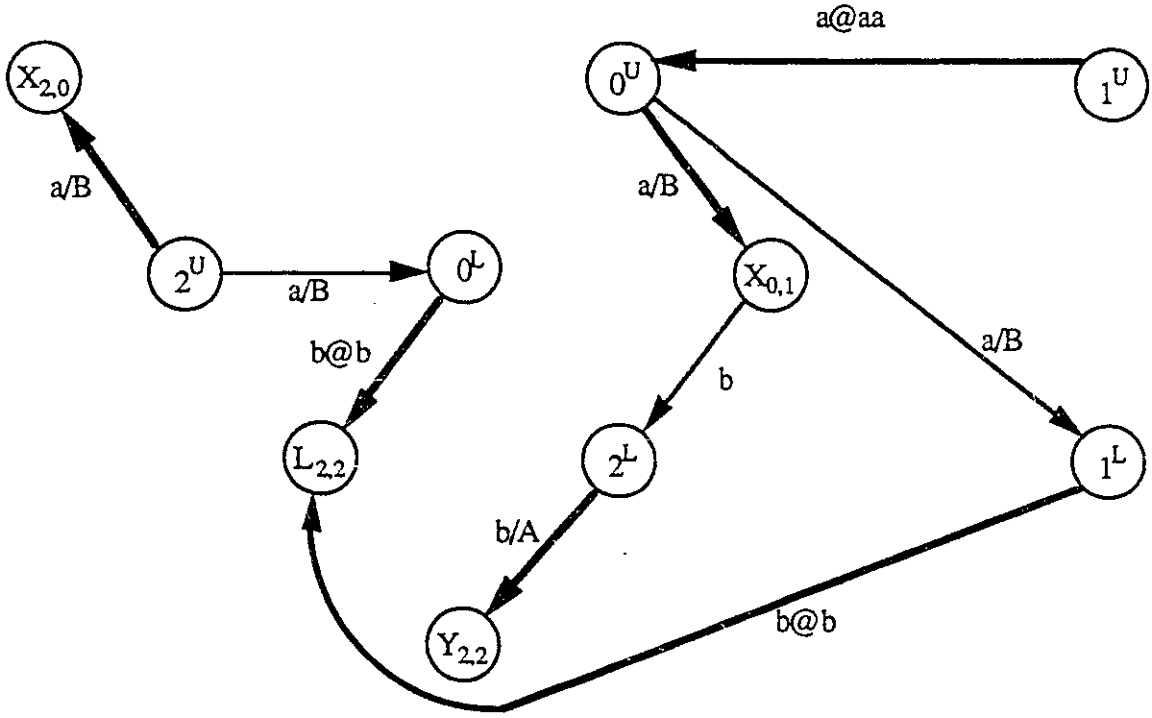


Figure 18. L_C* and the Edge-induced Spanning Subgraph D[L_C*]

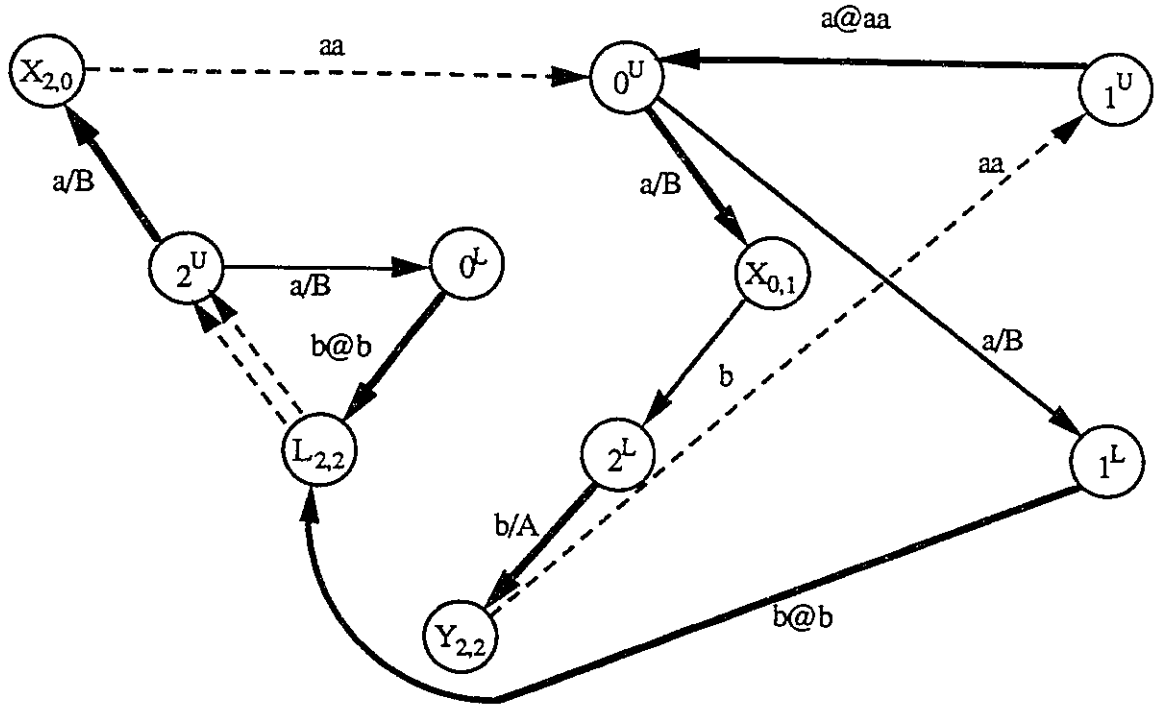


Figure 19. An RSA D* of $D^{\sim}$ over L_C*

Dashed edges are replicates.

7.2.3. Extended D-Method for a Distinguishable FSM

Similar to the method described in section 7.1.3 for an FSM which is only distinguishable but not strongly distinguishable, we partition vertex set V into equivalence sets. That is $V = \cup_{i=1,2,..,s} US_i = \cup_{i=1,2,..,t} LS_i$, where $US_i \cap US_j = \emptyset$ and $LS_i \cap LS_j = \emptyset$ for $i \neq j$, and the intersection $US_i \cap LS_j$ contains at most one single state of the FSM for i=1,2,..,s and i=1,2,..,t.

In an FSM, a synchronizable input sequence is said to be a *sub-synchronizable distinguishing sequence with respect to upper tester*, denoted by $SSDS^U$ if the output string produced by the FSM in response to the $SSDS^U$ is different for states in different set US_i, $1 \leq i \leq s$. a *sub-synchronizable distinguishing sequence with respect to lower tester*, denoted by $SSDS^L$ is also similarly defined. Obviously, if there exists an $SSDS^U$ and an $SSDS^L$ for a given distinguishable FSM, the pair of output strings (o^U, o^L) produced by the FSM in

response to the pair of sub-synchronizable distinguishing sequences ($SSDS^U$,$SSDS^L$) is different for each starting state. Hence, the pair ($SSDS^U$, $SSDS^L$) can be used in replace of (SDS^U, SDS^L) when the FSM is not strongly distinguishable.

Given a distinguishable FSM, an $SSDS^U$ of minimum length can be computed by constructing a similar upper distinguishing tree from digraph D=(V,E) except that, instead of being a vector of vertices, the uncertainty is a vector of indistinguishable sets with respect to either upper or lower tester and the uncertainty at level 0 includes only the indistinguishable sets with respect to upper tester. An $SSDS^L$ of minimum length can be similarly computed.

Corresponding to the pair of ($SSDS^U$,$SSDS^L$), an i@SSDS-diagram $D^\wedge$ can be obtained and the RPT algorithm is then applied to $D^\sim$ derived from $D^\wedge$ to obtain a synchronizable test sequence. The steps in developing such a test sequence are the same as those described in section 7.2.1, except that Step 2 in section 7.2.1 should be modified, since in a distinguishable FSM, both $SSDS^U$ and $SSDS^L$ are required to apply to a transition where applicable. The modified Step 2 is as follows:

Step 2: Construct an i@SSDS-diagram as $D^\wedge=(V^\wedge,E^\wedge)$, such that $V^\wedge = V' = V^U \cup V^L \cup H$ and $E^\wedge = E' \cup L_C = E_C \cup F \cup L_C$, where L_C is constructed in the following.

For each edge $e = (v_p, v_q: i_1/o_1) \in E$, let $e' = (x_j, x_k: i_1/o_1) \in E_C$ be the corresponding inherent or parent edge. We have the following two cases:

Case 1: e' is an inherent edge, then $x_k \in V^U \cup V^L$. Let e_{last} be the last edge in E in response to the input sequence i@SSDS applied at state v_p, where SSDS stands for SDS^U if $x_k \in V^U$, otherwise, SSDS stands for $SSDS^L$. Let (x_s, x_t; i_k/o_k) $\in E_C$ be the inherent or parent edge corresponding to e_{last}. We create an edge (x_j, x_t; i_1@SSDS) in L_C and let the edge has cost Length(SSDS)+1.

Case 2: e' is a parent edge, then $x_k \in H$, *i.e.*, $x_k = U_{p,q}$ or $x_k = L_{p,q}$. Let $e_{U\text{-last}}$ and $e_{L\text{-last}}$ be the last edges in E in response to the input sequences i_1@SSDS^U and i_1@SSDS^L applied at state v_p, respectively. Let $(x_s, x_t; i_h/o_h)$, $(x_u, x_v; i_k/o_k) \in E_C$ be the inherent edge or the parent edge corresponding to $e_{U\text{-last}}$ and $e_{L\text{-last}}$, respectively. We create edges $(x_k, x_t; i_1$@SSDS$^U)$, $(x_k, x_v; i_1$@SSDS$^L)$ in L_C and let edges $(x_k, x_t; i_1$@SSDS$^U)$, $(x_k, x_v; i_1$@SSDS$^L)$ have cost Length(SSDS^U)+1 and Length(SSDS^L)+1, respectively.

Due to the manner D^ is obtained, D^ will be strongly connected.

The total complexity is also O((m+n) n log(m+n)), where m is the number of states and n is the number of transitions in the FSM.

7.2.4. An Example

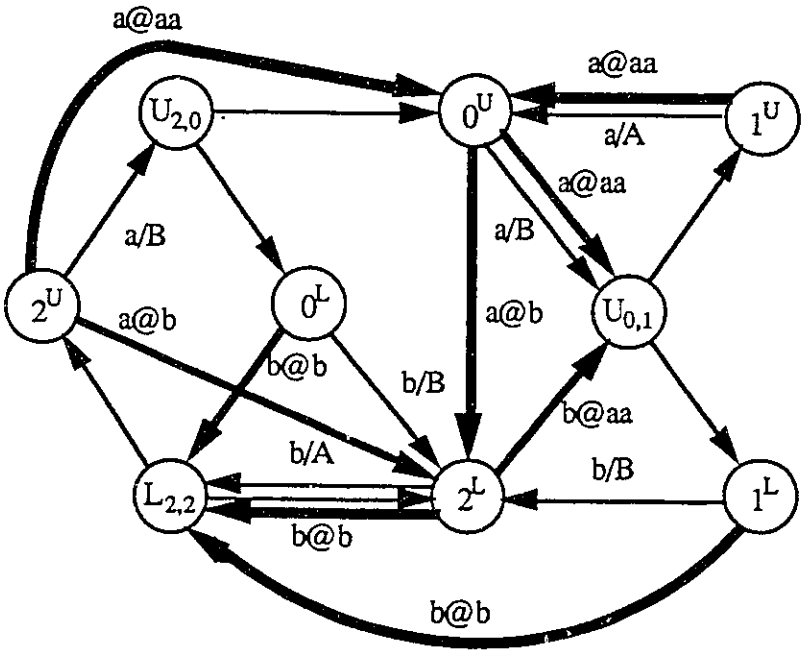


Figure 20. An i@SSDS-diagram D^

Boldfaced edges are edges in L_C.

Consider the digraph of Figure 13, which is not strongly distinguishable but distinguishable. From section 7.1.4, sets of indistinguishable states with respect to upper tester are {0}, {1}, {2} and sets of indistinguishable states with respect to lower tester are {0,1}, {2}. A pair of ($SSDS^U$,$SSDS^L$) is (aa, b). The duplex digraph D'=(V',E") is the same as that in Figure 15 except the label of edge (0^L, 2^L) is "b/B" instead of "b/C". An i@SSDS-diagram D^ is shown in Figure 20. Edges in L_C are boldfaced in Figure 20, where L_C = {(0^U, $U_{0,1}$; a@aa), (0^U, 2^L; a@b), (0^L, $L_{2,2}$; b@b), (1^U, 0^U; a@aa), (1^L, $L_{2,2}$; b@b), (2^U, 0^U; a@aa), (2^U, 2^L; a@b), (2^L, $U_{0,1}$; b@aa), (2^L, $L_{2,2}$; b@b)}.

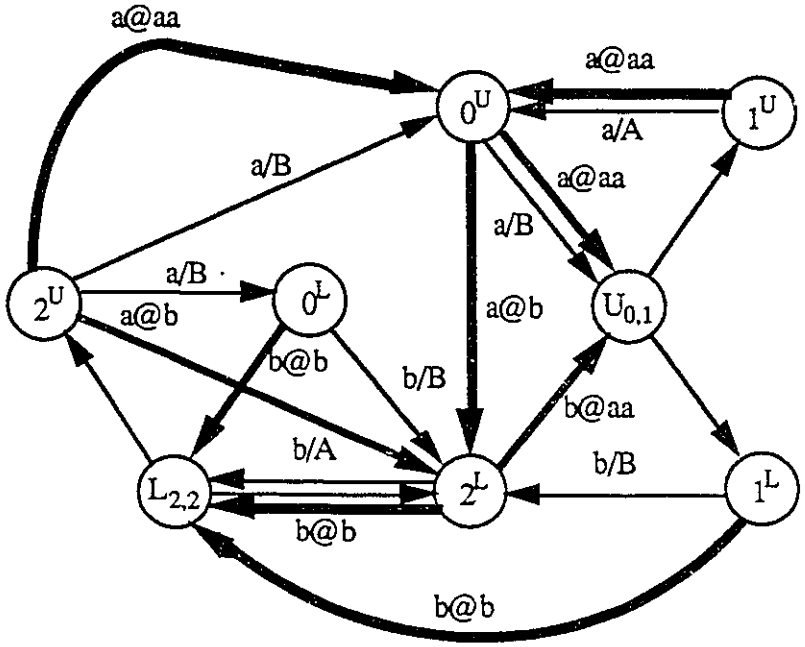


Figure 21. A Digraph D~ Obtained from D^ in Figure 20

We then derive D~ from D^, as is shown in Figure 21, by removing vertex $U_{2,0}$, and edges (2^U, $U_{2,0}$; a/B), ($U_{2,0}$, 0^U; null/null) and ($U_{2,0}$, 0^L; null/null), and adding edges

$(2^U, 0^U; a/B)$ and $(2^U, 0^L; a/B)$. The edge-induced subgraph $D[L_C]$ is then a spanning subgraph of $D^{\sim}$.

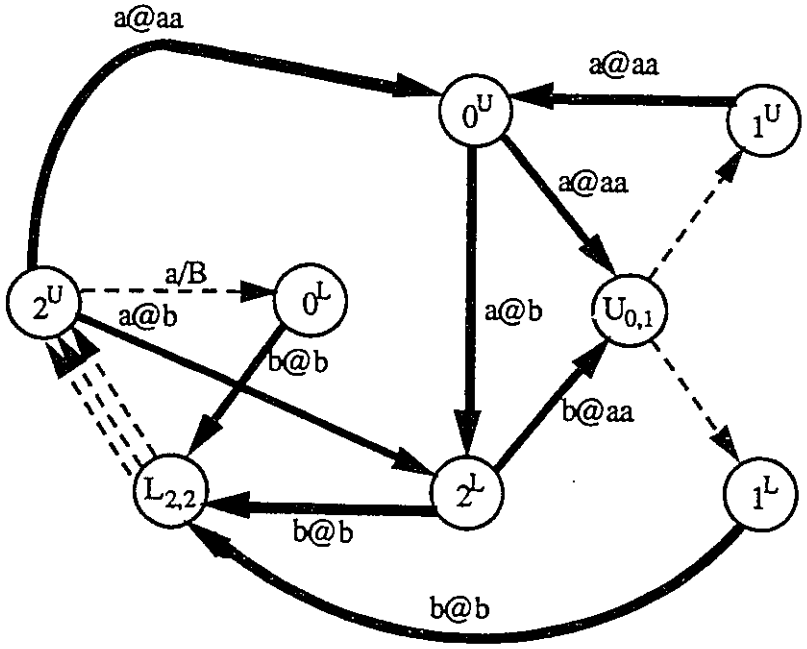


Figure 22. An RSA D* of $D^{\sim}$ in Figure 21

Dashed edges are replicates.

From Figure 21, we know that $D[L_C]$ is also weakly connected. Hence the RSA D* of $D^{\sim}$, as is shown in Figure 22, is strongly connected. An Euler tour starting at 0^U in D* is

a@aa null/null a@aa a@b b@aa null/null b@b null/null
a@b b@b null/null a/B b@b null/null a@aa

By removing the "null/null" labels, we obtain

a@aa a@aa a@b b@aa b@b a@b b@b a/B b@b a@aa

Hence, synchronizable test sequence T for the FSM represented by D=(V,E) is

a@aa a@aa a@b b@aa b@b a@b b@b a b@b a@aa

which is, in fact, a minimum-length synchronizable test sequence, since $D[L_C]$ is itself weakly connected.

7.3. Extensions of the UIO-Method

In an FSM, a *unique input/output* (UIO) [33] for a state of the FSM is an I/O behaviour that is not exhibited by any other state of the FSM. The UIO sequences are used to check whether the FSM reaches the expected state after the execution of each transition.

An I/O sequence is said to be a *synchronizable unique input/output* (SUIO) for a state of the FSM if

(i) it is a synchronizable transition sequence in the FSM and

(ii) the I/O behaviour is unique (*i.e.*, not exhibited by any other state of the FSM).

The argument in the section 7.1 about the synchronization problem associated with the application of SDS applies similarly to the SUIO sequence and suggests that we need two SUIO sequences for each state v_j, denoted by $SUIO^U(v_j)$ and $SUIO^L(v_j)$. $SUIO^U(v_j)$ is initiated by the upper tester (*i.e.*, the input symbol of the first transition is from the upper tester) and $SUIO^L(v_j)$ is initiated by the lower tester. Any transition which transfers the FSM to the expected state v_j can then be checked by applying either $SUIO^U(v_j)$ or $SUIO^L(v_j)$.

7.3.1. Extended UIO-Method for a Strongly Distinguishable FSM

In order for an FSM to have an $SUIO^U(v_j)$ ($SUIO^L(v_j)$) for each state v_j, the FSM should not contain indistinguishable states with respect to upper (lower) tester, that is the FSM should be strongly distinguishable.

We assume that the FSM is strongly distinguishable, robustly biconnected, and possesses two synchronizable unique input/output sequences for each state v_j, $SUIO^U(v_j)$

and $SUIO^L(v_j)$. We also assume that the FSM has reset inputs ri^U and ri^L such that **type**$(ri^U) = R^{IU}S^{IU,L}$ and **type**$(ri^L) = R^{IL}S^{IL,U}$.

Sabnani and Dahbura [33] gave a procedure to compute a minimum length UIO sequence for each state. The procedure can also be used to compute $SUIO^U(v)$ for each state v except that initially only those transitions initiated by the upper tester are computed and that when the input/output sequences of length ℓ are computed from those of length $(\ell - 1)$, only those synchronizable successive transitions are considered. $SUIO^U(v)$ for each state v is similarly computed.

Similar to the method developed in section 7.2.1, a COPT T in digraph D=(V,E) is obtained by using a graph traversal technique [1] to construct an RPT T* of a strongly digraph $D^{\sim}=(V^{\sim},E^{\sim})$ over edges L_C defined below. The COPT T is then a synchronizable test sequence for the FSM represented by D which checks both output and transfer functions of the FSM. The following steps describe the method:

Step 1: Construct the duplex digraph D'=(V',E') from digraph D=(V,E) by the method described in section 6.1 with v_o^U or v_o^L as the start state, where $V' = V^U \cup V^L \cup H$ and $E' = E_C \cup F$. Let each edge e' in E_C has the label of the edge e in E which gives rise to e' and each child edge in F has the label of "null/null". Each edge in E_C is given cost one and each edge in F is given cost zero.

Step 2: Construct a digraph $D^{\wedge}=(V^{\wedge},E^{\wedge})$, such that $V^{\wedge} = V' \cup Z = V^U \cup V^L \cup H \cup Z$ and $E^{\wedge} = E' \cup L_C \cup L = E_C \cup F \cup L_C \cup L$, where Z, L_C and L are constructed as follows:

For each edge $e = (v_p, v_q: i_1/o_1) \in E$, let $e' = (x_j, x_k: i_1/o_1) \in E_C$ be the corresponding inherent or parent edge. We have the following two cases:

Case 1: e' is an inherent edge, then $x_k \in V^U \cup V^L$. Let e_{last} be the edge in E corresponding to the last transition of the sequence $i_1/o_1@SUIO(x_k)$, where $SUIO(x_k)$ stands for either $SUIO^U(v_q^U)$ if $x_k=v_q^U$ or $SUIO^L(v_q^L)$ if

$x_k = v_q^L$. Let $(x_s, x_t; i_k/o_k) \in E_C$ be the inherent or parent edge corresponding to e_{last}. We create an edge $(x_j, x_t; i_1/o_1@SUIO(x_k))$ in L_C and let the edge has cost Length(SUIO(x_k))+1, where Length(SUIO(x_k)) stands for the length of SUIO(x_k).

Case 2: e' is a parent edge, then $x_k \in H$, *i.e.*, $x_k = U_{p,q}$ or $x_k = L_{p,q}$. Then e' has two child edges (x_k, v_q^U) and (x_k, v_q^L). Let $e_{U\text{-}last}$ and $e_{L\text{-}last}$ be the edges in E corresponding to the last transitions of the sequences $i_1/o_1@SUIO^U(v_q^U)$ and $i_1/o_1@SUIO^L(v_q^L)$, respectively. Let $(x_s, x_t; i_h/o_h)$, $(x_u, x_v; i_k/o_k) \in E_C$ be the inherent edge or the parent edge corresponding to $e_{U\text{-}last}$ and $e_{L\text{-}last}$, respectively.

If $x_k = U_{p,q}$, we create a vertex $X_{p,q}$ in Z; an edge $(x_j, X_{p,q}; i_1/o_1)$ in L_C; an edge $(X_{p,q}, x_t; SUIO^U(v_q^U))$ in L; and an edge $(X_{p,q}, x_v; SUIO^L(v_q^L))$ in L. Let edge $(x_j, X_{p,q}; i_1/o_1)$ has cost one and edges $(X_{p,q}, x_t; SUIO^U(v_q^U))$ and $(X_{p,q}, x_v; SUIO^L(v_q^L))$ have cost Length($SUIO^U(v_q^U)$) and Length($SUIO^L(v_q^L)$), respectively.

If $x_k = L_{p,q}$, we create a vertex $Y_{p,q}$ in Z; an edge $(x_j, Y_{p,q}; i_1/o_1)$ in L_C; an edge $(Y_{p,q}, x_t; SUIO^U(v_q^U))$ in L; and an edge $(Y_{p,q}, x_v; SUIO^L(v_q^L))$ in L. Let edge $(x_j, Y_{p,q}; i_1/o_1)$ has cost one and edges $(Y_{p,q}\ x_t; SUIO^U(v_q^U))$ and $(Y_{p,q}, x_v; SUIO^L(v_q^L))$ have cost Length($SUIO^U(v_q^U)$) and Length($SUIO^L(v_q^L)$), respectively.

Since D' is strongly connected, due to the manner D^ is obtained, D^ will be strongly connected.

Step 3: The edge-induced subgraph $D[L_C]$ might not be a spanning subgraph of D^, since some vertex in H might not be incident with any edge in L_C. We create a digraph $D^{\sim}=(V^{\sim},E^{\sim})$ from $D^{\wedge}=(V^{\wedge},E^{\wedge})$ by removing those vertices which are not incident with any edge in L_C as follows:

1) Every vertex in $V^{\wedge}$ which is incident with some edge in L_C is included in $V^{\sim}$;

2) Every edge in $E^{\wedge}$ with both head vertex and tail vertex in $V^{\sim}$ is included in $E^{\sim}$.

3) For each vertex x in $V^{\wedge}$ but not in $V^{\sim}$, we know that $v \in H$. There are a pair of two child edges (x,y; null/null) and (x,z; null/null) incident with x and they are the only edges leaving vertex x. For each incoming edge (w,x; label), we create two edges (w,y; label) and (w,z; label) in $E^{\sim}$ and let the costs of these two edges be the cost of edge (w,x; label).

We have that $D^{\sim}$ is also strongly connected, $E^{\sim} \supseteq L_C$ and $D[L_C]$ is a spanning subgraph of $D^{\sim}$. Finding a test sequence which includes every edge in L_C at least once for the digraph $D^{\sim}=(V^{\sim}, E^{\sim})$ is reduced to finding an RPT of $D^{\sim}$ over L_C [1].

Step 4: If $D[L_C]$ is not weakly connected, edges from $E^{\sim}$ are added to L_C to obtain L_C^* until $D[L_C^*]$ is weakly connected.

Step 5: Construct an RSA D^* of $D^{\sim}$ over L_C^* and find an Euler tour T^* in D^*, which is then an RPT T^* of $D^{\sim}$ over L_C^*, where each edge in T^* is represented by the label of the edge. By removing the label of "null/null" in T^* will give rise to a synchronizable test sequence T in D which checks output and transfer functions. Note that if $D[L_C]$ is itself weakly connected, T^* is also an RCPT in $D^{\sim}$ over L_C and T is then a minimum-length synchronizable test sequence.

The complexity of the above steps are the same as those in section 7.2.1. Hence, given $SUIO^U$ and $SUIO^L$ sequences for an FSM represented by D=(V,E), the complexity of constructing a synchronizable test sequence for the FSM is O((V+E) E log(V+E)), or O((m+n) n log(m+n)), where m is the number of states and n is the number of transitions in the FSM.

7.3.2. An Example

Consider the digraph in Figure 11. It can verified that $SUIO^U(0)$ = a/B,a/A; $SUIO^U(1)$ = a/A; $SUIO^U(2)$ = a/B,a/B; $SUIO^L(0)$ = b/C; $SUIO^L(1)$ = b/B; $SUIO^L(2)$ = b/A.

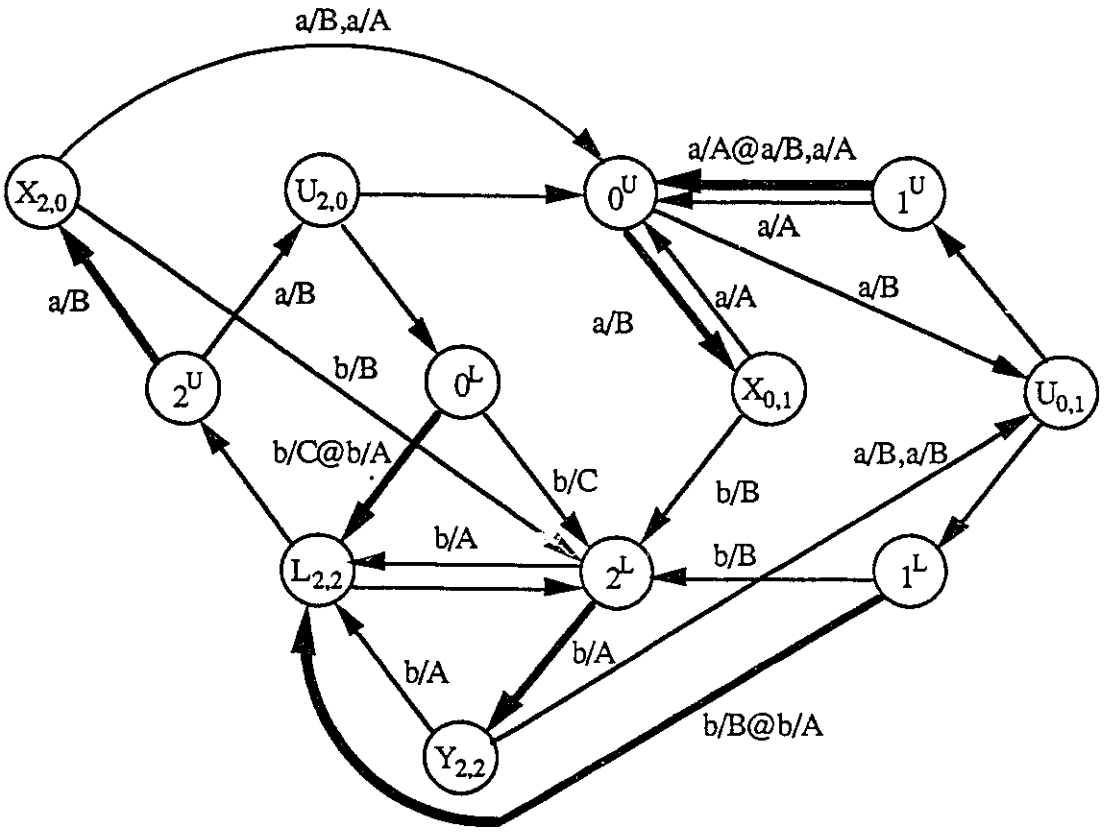


Figure 23. A Digraph D^ Obtained from D' in Figure 15

Boldfaced edges are edges in E_C.

A digraph D^=(V^,E^) shown in Figure 23 is constructed from D'=(V',E') in Figure 15 such that $V^ = V' \cup \{X_{0,1}, X_{2,0}, Y_{2,2}\}$ and $E^ = E' \cup L_C$, where L_C = $\{(0^U,X_{0,1}; a/B), (0^L, I_{2,2}; b/C@b/A), (1^U, 0^U; a/A@a/B,a/A), (1^L, L_{2,2}; b/B@b/A), (2^U,X_{2,0}; a/B), (2^L, Y_{2,2}; b/A)\}$. Edges in L_C are boldfaced in Figure 23.

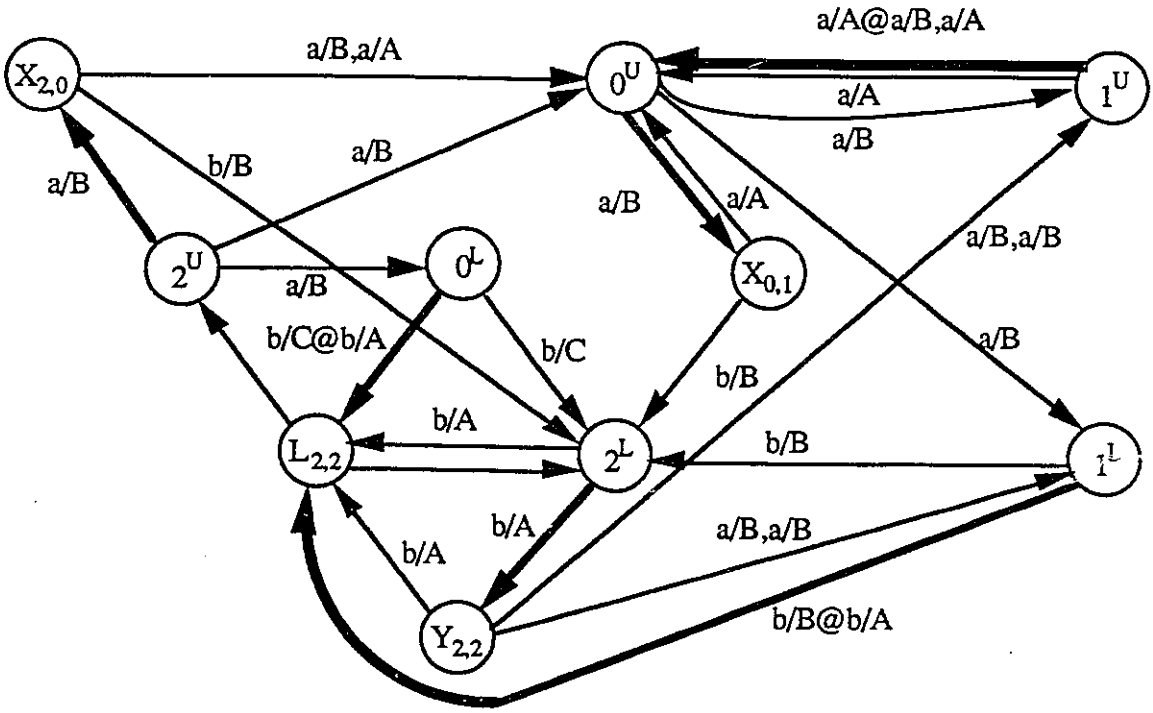


Figure 24. A Digraph D~ Obtained from D^ in Figure 23

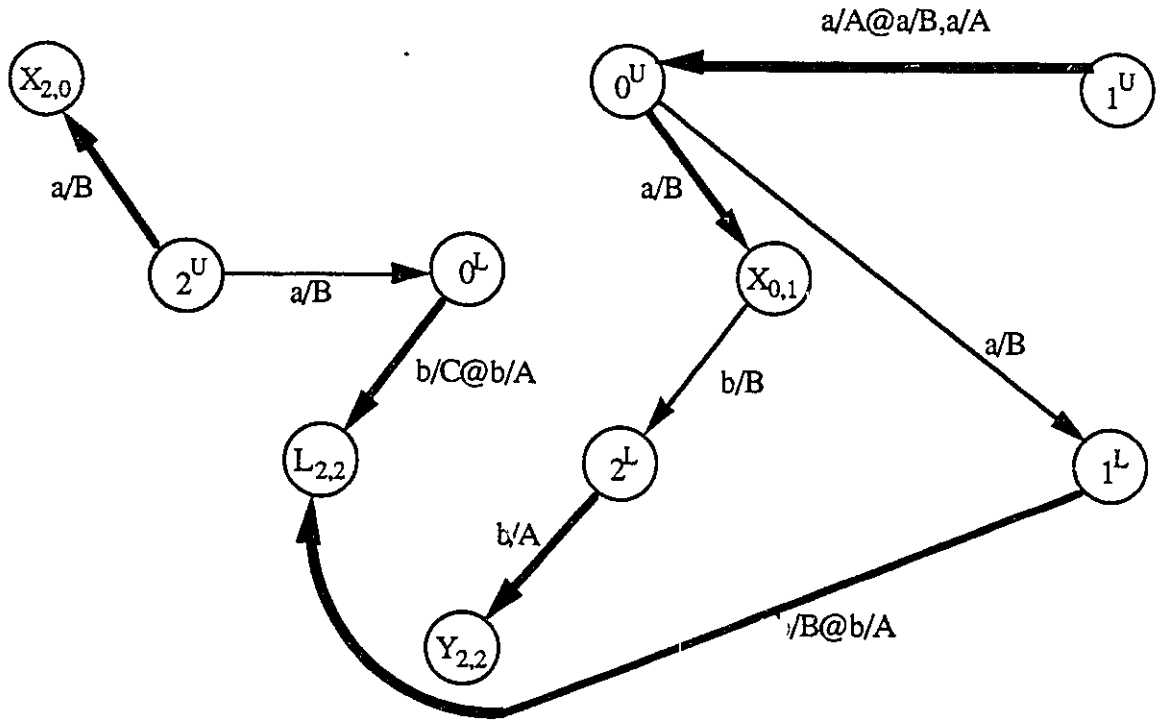


Figure 25. An Edge-induced Spanning Subgraph D[L_C*]

$D[L_C]$ is not a spanning subgraph of $D^\wedge$ since vertices $U_{0,1}$ and $U_{2,0}$ are not incident with any edge in L_C. Thus, we create a digraph $D^\sim=(V^\sim,E^\sim)$ by the procedure in step 3. The digraph $D^\sim=(V^\sim,E^\sim)$ is shown in Figure 24. Since $D[L_C]$ is not weakly connected, we add edges $(0^U, 1^L; a/B)$, $(X_{0,1}, 2^L; b/B)$ and $(2^U, 0^L; a/B)$ to L_C to obtain L_C^*, which is shown in Figures 25. An RSA D^* of $D^\wedge$ is shown in Figure 26.

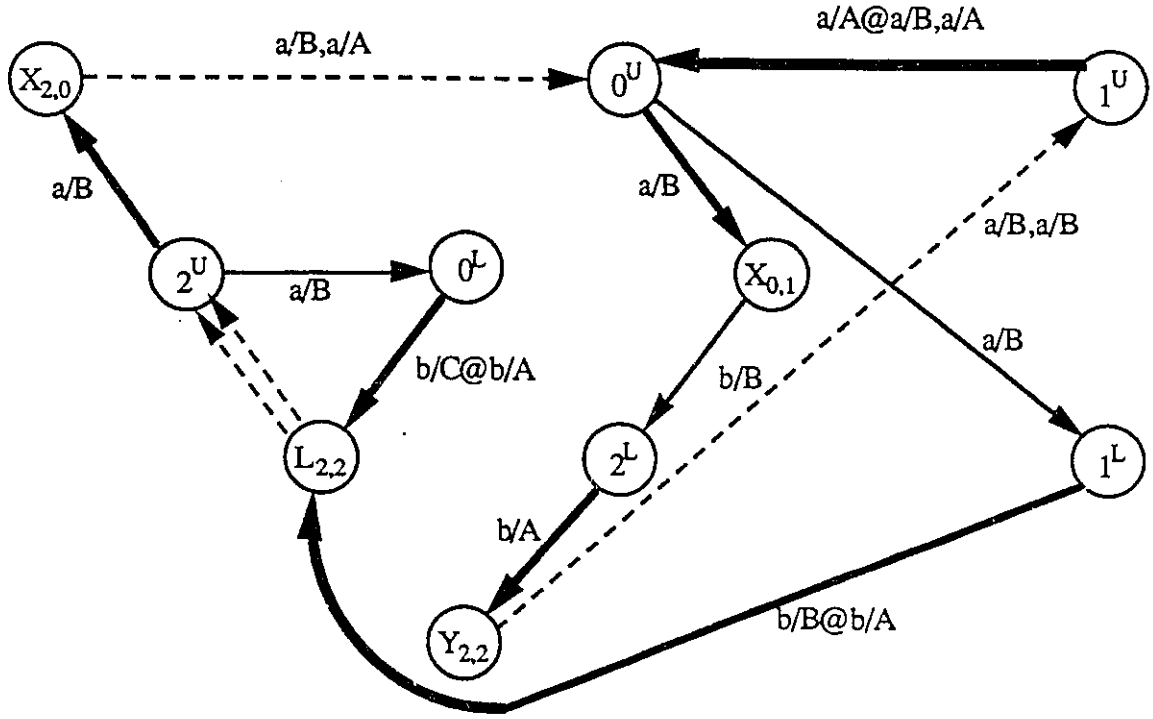


Figure 26. An RSA D^* of $D^\sim$ in Figure 25

Dashed edges are replicates.

An Euler tour starting at 0^U in D^* of Figure 26 is

a/B	b/B	b/A	a/B,a/B	a/A@a/B,a/A	a/B	b/B@b/A
null/null		a/B	b/C@b/A	null/null	a/B	a/B,a/A

By removing null/null label, we obtain the following sequence:

a/B b/B b/A a/B,a/B a/A,a/B,a/A a/B b/B,b/A a/B
b/C,b/A a/B a/B,a/A

Hence a synchronizable test sequence for the FSM represented by digraph D in Figure 11 is

a@b b@aa a@aa a@bb a@bb a@aa

7.3.3. Extended UIO-Method for a Distinguishable FSM

Similar to the method described in 7.1.2 for an FSM which is only distinguishable but not strongly distinguishable, we partition vertex set V into equivalence sets. That is $V = \cup_{i=1,2,..,s} US_i = \cup_{i=1,2,..,t} LS_i$, where $US_i \cap US_j = \emptyset$ and $LS_i \cap LS_j = \emptyset$ for $i \neq j$, and the intersection $US_i \cap LS_j$ contains at most one single state of the FSM for i=1,2,..,s and i=1,2,..,t.

An I/O sequence is said to be a *sub-synchronizable unique input/output sequence from upper tester for set* US_i if

(i) the sequence is initiated by upper tester,

(ii) the sequence is synchronizable input sequence for any state in US_i, and

(iii) the input/output behavior is not exhibited by any state other than those in US_i. We denote such a sequence by $SSUIO^U(US_i)$.

A *sub-synchronizable unique input/output sequence from lower tester for set* LS_i, denoted by $SSUIO^L(LS_j)$, is similarly defined.

The procedure of computing $SSUIO^U(v)$ for each state v described in section 7.3.1 can be similarly used to compute $SSUIO^U(US_i)$ for each set US_i except that now our objective is sets of vertices instead of vertices.

Given $SSUIO^U(US_i)$ and $SSUIO^L(LS_j)$ for each US_i and LS_j, respectively, for i=1,2.,...,s, j=1,2,...,t, the I/O behavior of the FSM from a state v in response to the pair $(SSUIO^U(US_i), SSUIO^L(LS_j))$ is unique, where $\{v\} = US_i \cap LS_j$ for some i and j. Hence

$\{(SSUIO^U(US_i), SSUIO^L(LS_j)): i=1,2,...,s; j=1,2,...t\}$ can be employed in replace of $\{SSUIO^U(v_i), SSUIO^L(v_i): i=1,2,...,m\}$ when the FSM is not strongly distinguishable. The procedure of generating a synchronizable test sequence is the same as that described in the section 7.3.1 except that Step 2 is modified as follows:

Step 2: Construct a digraph $D^\wedge=(V^\wedge,E^\wedge)$, such that $V^\wedge = V' = V^U \cup V^L \cup H$ and $E^\wedge = E' \cup L_C = E_C \cup F \cup L_C$, where L_C is constructed in the following.

For each edge $e = (v_p, v_q: i_1/o_1) \in E$, let $e' = (x_j, x_k: i_1/o_1) \in E_C$ be the corresponding inherent or parent edge. We have the following two cases:

Case 1: e' is an inherent edge, then $x_k \in V^U \cup V^L$.

If $x_k = v_q{}^U$ and $v_q \in US_f$ for some f,, let e_{last} be the edge in E corresponding to the last transition of the sequence $i_1/o_1@SSUIO^U(US_f)$ and let $(x_s, x_t; i_k/o_k) \in E_C$ be the inherent or parent edge corresponding to e_{last}. We create an edge $(x_j, x_t; i_1/o_1@SSUIO^U(US_f))$ in L_C and let the edge has cost $Length(SSUIO^U(US_f))+1$.

Otherwise, if $x_k=v_q{}^L$ and $v_q \in LS_f$ for some f, let e_{last} be the edge in E corresponding to the last input/output of the sequence $i_1/o_1@SSUIO^L(LS_f)$ and let $(x_s, x_t; i_k/o_k) \in E_C$ be the inherent or parent edge corresponding to e_{last}. We create an edge $(x_j, x_t; i_1/o_1@SSUIO^L(LS_f))$ in L_C and let the edge has cost $Length(SSUIO^L(LS_f))+1$.

Case 2: e' is a parent edge, then $x_k \in H$, *i.e.*, $x_k = U_{p,q}$ or $x_k = L_{p,q}$ and $(x_k, v_q{}^U)$ and $(x_k, v_q{}^L)$ are the child edges of e', where $v_q \in US_f$ and $v_q \in LS_g$ for some f and g. Let $e_{U\text{-}last}$ and $e_{L\text{-}last}$ be the edges in E corresponding to the last transitions of the sequences $i_1/o_1@SSUIO^U(US_f)$ and $i_1/o_1@SSUIO^L(LS_g)$, respectively. Let $(x_a, x_b; i_h/o_h)$, $(x_u, x_v; i_k/o_k) \in E_C$ be the inherent edge or the parent edge corresponding to $e_{U\text{-}last}$ and $e_{L\text{-}last}$, respectively. We create edges $(x_j, x_b; i_1/o_1@SSUIO^U(US_f))$ and $(x_j, x_v;$

i_1/o_1@SSUIO^L(LS_g)) in L_C and let edges (x_j,x_b; i_1/o_1@SSUIO^U(US_f)) and (x_j, x_v; i_1/o_1@SSUIO^L(LS_g)) have cost Length(SSUIO^U(US_f))+1 and Length(SSUIO^L(LS_g))+1, respectively.

Since D' is strongly connected, D^ is strongly connected.

The total complexity is also O(q (m+n) n log(m+n)), where q is the number of input symbols m is the number of states and n is the number of transitions in the FSM.

7.3.4. An Example

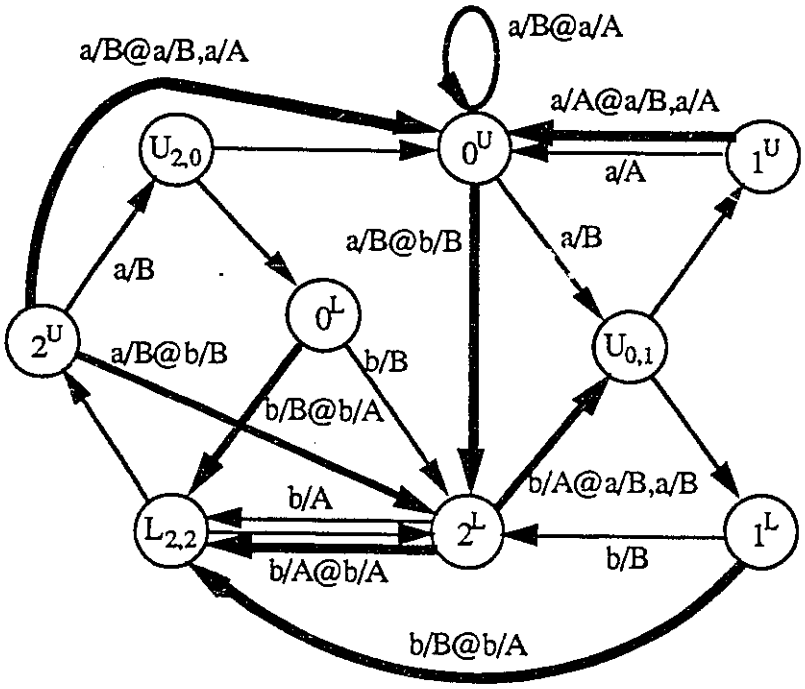


Figure 27. A Digraph D^ Obtained from D' Derived from D in Figure 13

Boldfaced edges are edges in L_C.

Consider the digraph in Figure 13, which is not strongly distinguishable but distinguishable. From section 7.1.4, sets of indistinguishable states with respect to upper tester are {0}, {1}, {2} and sets of indistinguishable states with respect to lower tester are {0,1}, {2}. We have that $SSUIO^U(\{1\})$ = a/A, $SSUIO^U(\{0\})$ = a/B,a/A, $SSUIO^U(\{2\})$ = a/B,a/B and $SSUIO^L(\{0,1\})$ = b/B, $SSUIO^L(\{2\})$ = b/A. The duplex digraph D'=(V',E') is the same as that in Figure 15 except the label of edge $(0^L, 2^L)$ is "b/B" instead of "b/C". A digraph $D^\wedge$ is shown in Figure 27. Edges in L_C are boldfaced in Figure 27, where L_C = {$(0^U, 0^U$; a/B@a/A), $(0^U, 2^L$; a/B@b/B), $(0^L, L_{2,2}$; b/B@b/A), $(1^U, 0^U$; a/A@a/B,a/A), $(1^L, L_{2,2}$; b/B@b/A), $(2^U, 0^U$; a/B@a/B,a/A), $(2^U, 2^L$; a/B@b/B), $(2^L, U_{0,1}$; b/A@a/B,a/B), $(2^L, L_{2,2}$; b/A@b/A)}.

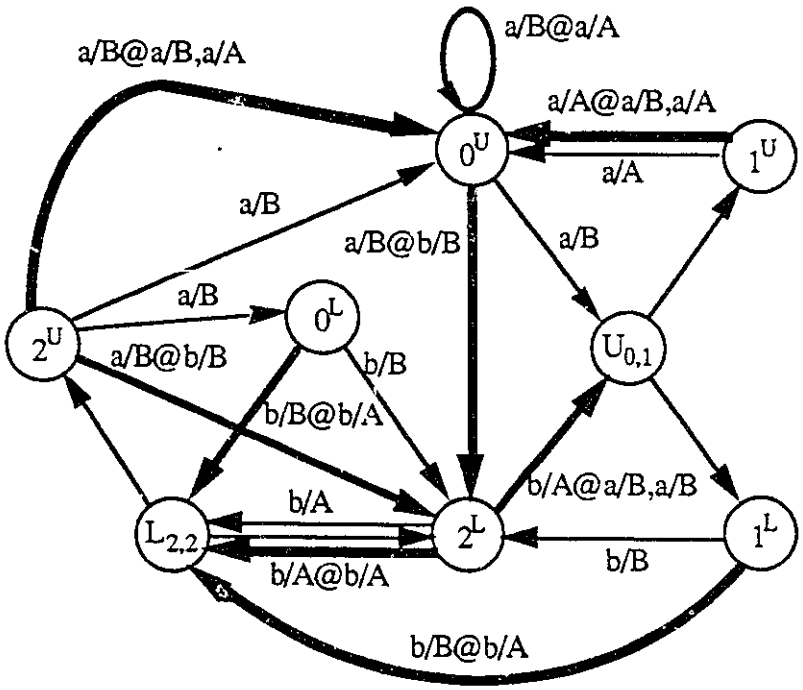


Figure 28. A Digraph $D^\sim$ Obtained from $D^\wedge$ in Figure 27

The Digraph $D^{\sim}$ in Figure 28 is derived from $D^{\wedge}$ by removing vertex $U_{2,0}$, and edges $(2^U, U_{2,0}; a/B)$, $(U_{2,0}, 0^U;$ null/null) and $(U_{2,0}, 0^L;$ null/null), and add edges $(2^U, 0^U; a/B)$ and $(2^U, 0^L; a/B)$. The edge-induced subgraph $D[L_C]$ is then a spanning subgraph of $D^{\sim}$.

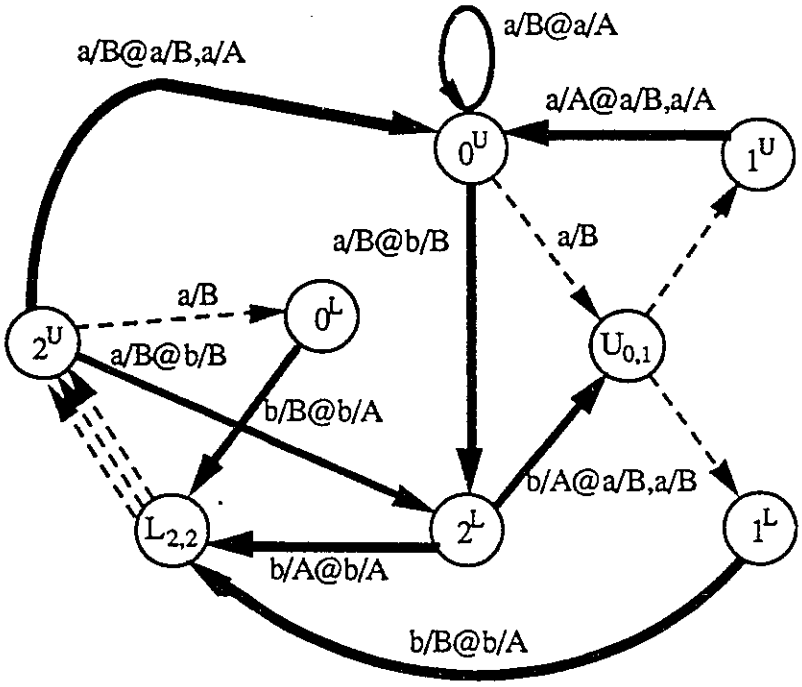


Figure 29. An RSA D* of $D^{\sim}$ in Figure 28

Dashed edges are replicates.

From Figure 28, we know that $D[L_C]$ is also weakly connected. Hence the RSA D* of $D^{\sim}$, as is shown in Figure 29, is strongly connected. An Euler tour starting at 0^U in D* is

a/B@a/A a/B null/null a/A@a/B,a/A a/B@b/B b/A@a/B,a/B
null/null b/B@b/A null/null a/B@b/B b/A@b/A
null/null a/B b/B@b/A null/null a/B@a/B,a/A

By removing the "null/null" labels, we obtain

a/B@a/A a/B a/A@a/B,a/A a/B@b/B b/A@a/B,a/B b/B@b/A
a/B@b/B b/A@b/A a/B b/B@b/A a/B@a/B,a/A

Hence, a synchronizable test sequence T for the FSM represented by D=(V,E) is

a@a a a@aa a@b b@aa b@b a@b b@b a b@b a@aa

which is, in fact, a minimum-length synchronizable test sequence, since $D[L_C]$ is itself weakly connected.

8. CONCLUSIONS

We have studied the problem of the synchronizable test sequence generation for a protocol specification modeled as a deterministic FSM. By defining an order-specified digraph [3] to represent the FSM, we are able to rephrase the problem as finding a correctly-ordered postman tour in the digraph.

With the aid of the order-specified digraph, we have given a necessary and sufficient condition for the existence of a synchronizable test sequence for a given FSM. We have also developed an algorithm with the complexity of $O(n^2)$ to verify the condition, where n is the number of transitions in the FSM. Therefore, we can determine if a synchronizable test sequence exists for a given FSM in $O(n^2)$ time.

To generate synchronizable test sequences for a given FSM, we have described a duplex digraph method which converts an order-specified digraph into a duplex digraph. The well-developed graph techniques to generate postman tour can then be applied to obtain synchronizable test sequences. We have developed an algorithm with the complexity of $O((m+n)\ n \log(m+n))$ to generate a synchronizable test sequence which checks only output functions of the FSM, where m is the number of states in the FSM.

The formal methods to generate test sequences for the protocol without synchronization issue include the W-method, D-method and UIO-method. The test sequences thus obtained can detect transfer errors as well as output errors. We have incorporated these methods into the protocol testing with synchronization issue to generate synchronizable test sequences which check both output and transfer errors. We defined synchronizable characterizing sets (*i.e.*, $\mathbf{SW}^U$ and $\mathbf{SW}^L$), synchronizable distinguishing sequences (*i.e.*, SDS^U and SDS^L) and synchronizable input/output sequences (*i.e.*, $SUIO^U$ and $SUIO^L$) which can be applied to a strongly distinguishable FSM. These synchronizable

sequences can be obtained in similar ways in constructing a W-set, a DS and UIO sequences [15,20,33] but only those sequences cause no synchronization problem are considered. We also defined sub-synchronizable characterizing sets (*i.e.*, $\mathbf{SSW}^U$ and $\mathbf{SSW}^L$), sub-synchronizable distinguishing sequences (*i.e.*, $SSDS^U$ and $SSDS^L$) and sub-synchronizable unique input/output sequences (*i.e.*, $SSUIO^U$ and $SSUIO^L$) which can be constructed, respectively, in the same way as $\mathbf{SW}^U$, $\mathbf{SW}^L$ SDS^U, SDS^L, $SUIO^U$ and $SUIO^L$ are constructed. The new sequences can then be applied to a non-strongly distinguishable but distinguishable FSM.

For an FSM represented by a robustly biconnected digraph, a synchronizable test sequence can be generated with a polynomial-time algorithm by each method which checks both the state transfer errors and the output errors. Given $\mathbf{SW}^U$ and $\mathbf{SW}^L$ (or, $\mathbf{SSW}^U$ and $\mathbf{SSW}^L$) in the W-method, the complexity is $O(qm^2)$. Given SDS^U and SDS^L (or, $SSDS^U$ and $SSDS^L$) in the D-method, the complexity is $O((m+n)\ n \log(m+n))$. Given $SUIO^U$ and $SUIO^L$ (or, $SSUIO^U$ and $SSUIO^L$) in the UIO-method, the complexity is $O((m+n)\ n \log(m+n))$.

However, for an FSM represented by a strongly biconnected digraph, the problem remains unsolved for generating a synchronizable test sequence by an efficient algorithm which checks both the state transfer errors and the output errors. It would be the direction of the future study to continue this work.

REFERENCES

[1] A.V. Aho, A.T. Dahbura, D. Lee, and M.U. Uyar, "An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours", IEEE Trans. on Communications, Vol., 39, No. 11, pp. 1604-1615, 1991.

[2] G.v. Bochmann, and C.A. Sunshine, "A survey of formal methods", in Computer Networks and Protocols, P.E. Green (editor), Chapter 20, pp. 561-578, Plenum Press, New York, 1983.

[3] T. Bolognesi, "Introduction to the ISO specification language LOTOS", Computer Networks and ISDN Systems 14, pp. 25-29, 1987.

[4] J. A. Bondy and U. S. R. Murty, "Graph Theory with Application", New York: Elsevier North Holland, Inc., 1976.

[5] S.C. Boyd and H. Ural, "On the complexity of generating optimal test sequences", IEEE Trans. on Software Engineering, Vol., 17, No. 9, pp. 976-978, 1991.

[6] S.C. Boyd and H. Ural, "The synchronization problem in protocol conformance testing", Info. Proc. Letters 40, pp.131-136, 1991.

[7] S. Budkowski, and P. Dembinski, "An introduction to Estelle: a specification language for distributed systems", Computer Networks and ISDN Systems 14, pp. 3-23, 1987.

[8] R. G. Busacker and T.L. Saaty, "Finite Graphs and Networks", McGraw-Hill, New York, 1965.

[9] CCITT SG XI, Recommendation Z.100, 1987.

[10] W.Y.L. Chan, S. Vuong, and M.R. Ito, "An improved protocol test generation procedure based on UIOS", Proc. of ACM SIGCOMM'89, pp. 283-294, Austin, Texas, Sept. 19-22, 1989.

[11] M.S. Chen, Y. Choi, and A. Kershenbaum, "Approaches utilizing segment overlap to minimize test sequences", Protocol Specification, Testing and Verification, X, L. Lorgrippo, R.L. Probert, and H, Ural (Editors), Elservier Science Publishers B. V., North-Holland, pp. 67-84, 1990.

[12] W.H. Chen, C.S. Lu, E.R. Brozovsky, and J.T. Wang, "An optimization technique for protocol conformance testing using multiple UIO sequences", Info. Proc. Lett. 36, pp. 7-11, 1990.

[13] W.H. Chen, C.S. Lu, L. Chen, and J.T. Wang, "Synchronizable protocol test sequence generation via the duplex technique", IEEE INFOCOM'90, Vol. 2, pp. 561-563, 1990

[14] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to Algorithms", the MIT Press, McGraw-Hill, New York, 1990.

[15] T. S. Chow, "Testing software designs modeled by finite-state machines", IEEE Trans. on Software Engineering, Vol. SE-4, No. 3, pp. 178-187, 1978.

[16] N. Christofides, "Graph Theory: An Algorithmic Approach", Academic Press, New York, 1975.

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

[18] S. Even, "Graph Algoirthms", Computer Science Press, Maryland, 1979.

[19] H. Fleischer, "Eulerian graphs", in Selected Topics in Graph Theory 2, ed., L. W. Beineke and R. J. Wilson, Academic Press, London, pp. 17-53, 1983.

[20] A. Gill, "Introduction to the Theory of Finite-state Machines", McGraw-Hill, New York, 1962.

[21] G. Gonenc, "A method for the design of fault detection experiments", IEEE Trans., Comput., Vol. C-19, pp. 551-558, 1970.

[22] ISO, Lotos - a formal description technique based on the temporal ordering of observational behaviour, IS 8807, 1988.

[23] ISO TC9/SC21, Information Processing Interconnection - Basic Reference Model, ISO 7498, 1984

[24] ISO TC97/SC21, OSI conformance testing methodology and framework - Part 1-5, ISO 2nd DP9646-1 revised text, edited by D. Rayner, Vancouver, December 1987.

[25] ISO TC 97/SC 21 IS 9074, International standard Estelle: a FDT based on an extended state transition model, Sept., 1988.

[26] Z. Kohavi, "Switching and Finite Automata Theory", McGraw-Hill, New York, 1978.

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

[28] J. K. Lenstra, and A. H. G. Rinnooy Kan, "On general routing problems", Networks, Vol. 6, pp. 273-280, 1976.

[29] Y. Lu, and H. Ural, "On formal methods for test sequence generation", Technical Report, TR-90-09, Dept. of CSI, Univ. of Ottawa, February 1990.

[30] R.E. Miller, and S. Paul, "Generating minimal length test sequences for conformance testing of communication protocols", IEEE INFOCOM'91, Vol. 2, pp. 0970-0971, 1991.

[31] S. Naito, and M. Tsunoyama, "Fault detection for sequential machines by transition tours", Proc. of the 11th IEEE Fault Tolerant Comput. Symposium, pp. 238-243, 1981.

[32] D. Rayner, "OSI conformance testing", Computer Networks and ISDN Systems, Vol. 14, pp. 79-98, 1987.

[33] K. Sabnani, and A.T. Dahbura, "A protocol test generation procedure", Computer Networks and ISDN Systems, Vol. 15, pp.285-297, 1988.

[34] B. Sarikaya, and G.v. Bochmann, "Synchronization and specification issues in protocol testing", IEEE Trans. on Comm., Vol. Comm-32, pp. 389-395, 1984.

[35] B. Sarikaya, G.v. Bochmann, and E. Cerny, "A test design methodology for protocol testing", IEEE Trans. on Software Eng., Vol. SE-13, pp. 518-531, 1987.

[36] 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 (Editors), Elservier Science Publisher B. V., North-Holland, pp. 131-143, 1989.

[37] D. 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.

[38] R. Tarjan, "Depth-first search and linear graph algorithms", SIAM J. Comput., Vol.1, No. 2, pp. 147-160, 1972.

[39] R. E. Tarjan, "Data Structures and Network Algorithms", Society for Industrial and Applied Mathematics, 1983.

[40] H. Ural, "Test sequence selection based on static data flow analysis', Comput. Commun., Vol. 10, No. 5, pp. 234-242, 1987.

[41] H. Ural, and B. Yang, "A test sequence selection method for protocol testing", IEEE Trans. Comm., Vol. 39, No. 4, pp. 514-523, 1991.

[42] H. Ural, and Y. Lu, "An improved method for test sequence generation", Technical Report, TR-90-12, Dept. of CSI, Univ. of Ottawa, March 1990.

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

[44] B. Yang, and H. Ural, "Protocol conformance test generation using multiple UIO sequences with overlapping", ACM SIGCOMM'90, pp. 118-125, Philadephia, Pennsylvania, Sept. 24-27, 1990.