# 25

# Synchronizable Checking Sequences Based on UIO Sequences

*S. Guyot and H. Ural*
*Department of Computer Science, University of Ottawa*
*150 Louis Pasteur, Ottawa, Ontario, K1N 6N5, Canada*

**Abstract**

This study addresses the synchronization problem that arises during the application of a predetermined checking sequence in some protocol test architectures that utilize remote testers. The synchronization problem can usually be solved by adding an additional communication channel or an additional protocol for coordination between the remote testers. Such requirements can be eliminated by constructing a synchronizable checking sequence such that the corresponding sequence of transitions causes no synchronization problem. Based on UIO sequences, a synchronizable checking sequence construction method is proposed.

**Keywords**

Synchronization, test coordination, checking sequence construction

## 1 INTRODUCTION

Determining, under certain assumptions, whether any given "black box" implementation of a Finite State Machine (*FSM*) is functioning correctly is referred to as a *fault detection* (*checking*) *experiment*. Foundations of fault detection experiments can be found in sequential circuit testing literature (Gill, 1962), (Hennie, 1964). This experiment consists of applying an input sequence and comparing the resulting output sequence to the expected output sequence. The applied input sequence and the expected output sequence form a *checking sequence*.

Given a deterministic FSM $M$, the construction of a checking sequence from $M$ with a designated initial state that determines whether any given FSM $N$ is a correct implementation of $M$ is sought in practice by considering some basic assumptions. The assumptions commonly made in the literature (Sabnani, 1988), (Chan, 1989), (Dahbura, 1990), (Aho, 1991) are that 1) $M$ is *minimal*, *completely specified*, and represented by a *strongly connected* digraph, 2) $N$ has the same set of inputs as $M$, and 3) faults in $N$ do not increase the number of states in $N$. In addition, the construction of a checking sequence must deal with the "black box" nature of a given implementation $N$ of $M$ which allows only limited controllability and observability of $N$. The limited controllability refers to not being able to directly transfer $N$ to a designated state and the limited observability refers to not being able to directly recognize the current state of $N$. In order to overcome the restrictions imposed by the limited controllability and observability, some special input sequences must be utilized in the construction of a checking sequence such that the output sequences produced by $N$ in response to these input sequences provide sufficient information to deduce that every state transition of $M$ is implemented correctly by $N$.

A. Cavalli et al. (eds.), *Protocol Test Systems VIII*

In order to verify the state transition from state $a$ to $b$ under input $x$, 1) before the application of $x$, $N$ must be transferred to the state recognized as $a$, 2) the output produced by $N$ in response to the application of $x$ must be as specified in $M$, and 3) the state reached by $N$ after the application of $x$ must be recognized as $b$. Hence, a crucial part of testing the correct implementation of each transition is recognizing the starting and terminating states of the transition. The recognition of a state of an FSM $M$ can be achieved by a distinguishing sequence (Hennie, 1964), a characterization set (Hennie, 1964) or a unique input-output (UIO) sequence (Sabnani, 1988). It is known that UIO sequences may not exist for every state of every minimal FSM (Sabnani, 1988) and determining the existence of a UIO sequence for a state of an FSM is PSPACE-complete (Lee, 1994).

Nevertheless, based on UIO sequences, various methods have been proposed in the literature to test FSMs (Sabnani, 1988), (Chan, 1989), (Dahbura, 1990), (Aho, 1991). In some of these methods, the UIO sequences obtained from an FSM $M$ are used for verifying states of a given implementation $N$ of $M$ without confirming that they are also unique in $N$. Chan (1989) observed that when the uniqueness of UIO sequences does not hold in a faulty $N$, certain errors in $N$ can not be detected. In order to overcome this problem, they proposed the *UIOv* method which is the UIO method (Sabnani, 1988) with the addition of a verification procedure to ensure that the UIO sequence for each state of $M$ is also unique in $N$.

Fault detection experiments have been used to test the conformance of various protocol implementations to their specifications given as FSMs (Dahbura, 1990). It is observed that protocol testing can be carried out as a fault detection experiment in some specific test architectures. One such architecture is shown in Figure 1 where the lower interface and the upper interface of the protocol implementation $N$ may be controlled and observed indirectly by the lower tester ($L$) and directly by the upper tester ($U$), respectively.

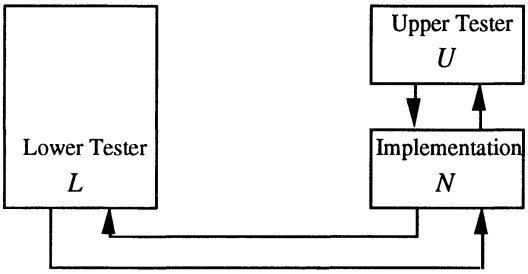


**Figure 1** A Distributed Test Architecture.

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

Synchronization between $U$ and $L$ can be achieved by any suitable test management protocol coordinating the actions of the testers (ISO TC97/SC21, 1987), by a ferry protocol with inter-process communication, or through manual coordination (by use of a telephone or terminal connection) (Rayner, 1987). 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 sequence of transitions that cause no synchronization problem (Ural, 1993), (Chen, 1995), (Tai, 1995).

In this paper, based on the work presented in (Chan, 1989), (Ural, 1993), (Perez, 1993), (Rezaki, 1994), a method for constructing synchronizable checking sequences is proposed. Related terminology is reviewed in section 2. In section 3, a method for synchronizable checking sequence construction using UIO sequences is presented and a proof for the resulting sequence to be a checking sequence is given. An illustrative example of the application of the proposed method is provided in section 4. In section 5, some minimization techniques are proposed. Concluding remarks are given in section 6.

# 2 PRELIMINARIES

## 2.1 FSM and its Graphical Representation

A *finite state machine*, *FSM*, is a quintuple $M = (S, X, Y, \delta, \lambda)$, where $S$ is a finite set of states, $X$ is a finite set of inputs, $Y$ is a finite set of outputs, $\delta$ is a state transition function that maps $S \times X$ to $S$, and $\lambda$ is an output function that maps $S \times X$ to $Y$. State $s_1 \in S$ is designated as the *initial state* of $M$, and $|S|$ is $n$. $- \in X$ and $- \in Y$ denote null input and null output, respectively. For a sequence of inputs $x_1x_2...\ x_k$, $x_i \in X$, $1 \le i \le k$ and a state $s \in S$, $\delta(s, x_1x_2...\ x_k) = \delta(\delta(s, x_1), x_2...\ x_k) = s_k$ and $\lambda(s, x_1x_2...\ x_k) = \lambda(s, x_1)\lambda(\delta(s, x_1), x_2...\ x_k) = y_1y_2...\ y_k$, where $y_i = \lambda(s_{i-1}, x_i)$, $s_i = \delta(s_{i-1}, x_i)$, $s_0 = s$, $y_i \in Y$, $s_i \in S$. In this case, it is said that $I = x_1x_2...\ x_k$ is an *input sequence*, $\lambda(s, I) = y_1y_2...\ y_k = O$ is an *output sequence*, and $I/O = (x_1x_2...\ x_k)/(y_1y_2...\ y_k) = (x_1/y_1)(x_2/y_2)...\ (x_k/y_k)$ is an *input/output sequence* (or IO-sequence, in short) starting at state $s$.

An FSM $M$ is said to be *minimal* if $\forall\ (s_i, s_j) \in S^2, i \neq j, \exists\ I \in X^k$, such that $\lambda(s_i, I) \neq \lambda(s_j, I)$. An FSM $M$ is said to be *completely specified* if $\forall x \in X, \forall s \in S, \exists\ (s, \delta(s, x); x/\lambda(s, x))$. An FSM $M$ can be represented by a directed graph (digraph) $D = (V,E)$ where a set of vertices $V = \{1, 2,..., n\}$ represents the set $S$ of states of $M$, and a set of directed edges $E = \{e_{ij} = (v_i, v_j; x/y);\ v_i, v_j \in V\}$ represents the set of transitions of $M$. Each $e_{ij} \in E$ represents a specified transition $t_{ij} = (s_i, s_j; x/y)$ of $M$ from state $s_i$ to state $s_j$ with *label* $x/y$, input $x \in X$ and output $y \in Y$. Vertices $v_i$ and $v_j$ which represent respectively state $s_i$ and $s_j$ are called the *head* and the *tail* of $e$, denoted by *head*($e$) and *tail*($e$). The label of an edge $e_{ij}$ (or transition $t_{ij}$) is denoted by *label*($e_{ij}$) (or *label*($t_{ij}$)).

A *path* $P = (n_1, n_2; x_1/y_1)(n_2, n_3; x_2/y_2)...\ (n_{k-1}, n_k; x_{k-1}/y_{k-1})$, $k > 1$, in a digraph $D = (V, E)$ is a finite sequence of adjacent (not necessarily distinct) edges in $D$, where $n_1$ and $n_k$, are called the *head* and the *tail* of $P$, denoted by *head*($P$) and *tail*($P$), respectively, and $(x_1/y_1)(x_2/y_2)...\ (x_{k-1}/y_{k-1})$, is called the *label* of $P$, denoted by *label*($P$). For convenience, a path $P = (n_1, n_2; x_1/y_1)(n_2, n_3; x_2/y_2)...\ (n_{k-1}, n_k; x_{k-1}/y_{k-1})$ will be represented by $(n_1, n_k; I/O)$ where $I/O$ is the IO-sequence $(x_1/y_1)(x_2/y_2)...\ (x_{k-1}/y_{k-1})$, $I = x_1x_2...\ x_{k-1}$ is the *input portion* of $I/O$, $O = y_1y_2...\ y_{k-1}$ is the *output portion* of $I/O$, respectively. A sequence $(i_1i_2...\ i_k)$ is a *subsequence* of $(x_1x_2...\ x_m)$ if there exists a $\Delta$, $0 \le \Delta \le m-k$, such that for all $j$, $1 \le j \le k$, $i_j = x_{j+\Delta}$. A sequence $(i_1i_2...\ i_k)$ is a *prefix* of $(x_1x_2...\ x_m)$ if for all $j$, $1 \le j \le k$, $i_j = x_j$.

A digraph $D = (V, E)$ is *strongly connected*, if for every pair of vertices $v_j$ and $v_k$, there exists a path from $v_j$ to $v_k$. An FSM $M$ has the *reset feature* if there is an input $r$ such that $\delta(s_i, r) = s_1$ (i.e. the initial state of $M$) and $\lambda(s_i, r) = -$, for every state $s_i$ of $M$. A *transfer sequence* $T$ of an FSM $M$ from state $s_i$ to state $s_j$ is the input portion of the label of a path from $s_i$ to $s_j$.

Let $\Phi(M)$ be the set of FSMs each of which has at most $n$ states and the same input set as $X$ of $M$. Let $N$ be in $\Phi(M)$. $M$ is *isomorphic* to $N$, denoted by $M \cong N$, if there is a one-to-one and onto function $f$ on the state sets of $M$ and $N$ such that for any transition $(i, j; x/y)$ of $M$, $(f(i), f(j); x/y)$ is a transition of $N$. A *checking sequence* of $M$ is an IO-sequence starting at a specific state of $M$ that distinguishes $M$ from any FSM of $\Phi(M)$ which is not isomorphic to $M$.

## 2.2 Synchronization Problem

Let each transition $t_{ij}$ of an FSM $M$ have one of the following labels, label($t_{ij}$) $\in$ $\{i^L/-, i^U/-, i^L/o^L, i^L/o^U, i^U/o^L, i^U/o^U, i^L/o^{U,L}, i^U/o^{U,L}\}$ where $i^L$ ($i^U$) is an input from $L(U)$, $o^L(o^U)$ is an output to $L(U)$ and $o^{U,L}$ is an output to $L$ and $U$. Then, considering two consecutive transitions of $M$, 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 an input to $M$ (Sarikaya, 1984). For example, a synchronization problem will occur if $t_1$ is followed by $t_2$ and label($t_1$) = $i^L/o^L$ whereas label($t_2$) $\in$ $\{i^U/-, i^U/o^L, i^U/o^U, i^U/o^{U,L}\}$.

Two consecutive transitions $t_{ij}$ and $t_{jk}$ of $M$ form a *synchronizable pair of transitions* if $t_{jk}$ can follow $t_{ij}$ without generating a synchronization problem. For example, a transition with label $i^L/o^U$ forms a synchronizable pair of transitions when followed by any other transition of $M$. 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 of an FSM is *synchronizable* if for every two consecutive transitions in the sequence, the second transition is an eligible successor of the first one. A checking sequence of an FSM is synchronizable if it is the label of a synchronizable transition sequence of the FSM. A *synchronizable input sequence* for state $s$ is the input portion of a synchronizable transition sequence that starts at $s$.

## 2.3 Order-specified Digraph and Synchronizable UIO Sequences

A digraph $D = (V, E)$ is called *order-specified* if for each edge $e_{ij} \in E$, a subset of the outgoing edges of vertex $j$ is specified as eligible successors of $e$. A path in an order-specified digraph $D$ is said to be *correctly-ordered* (CO) if for every pair of consecutive edges $e_{ij}$ and $e_{jk}$ in the path, $e_{jk}$ has been specified as an eligible successor of $e_{ij}$. The label of a CO path is called a *correctly-ordered IO-sequence*.

A *Unique input-output* (UIO) sequence for a state of an FSM is a sequence of input/output symbols that are not exhibited by any other state of the FSM (Sabnani, 1988). A *synchronizable unique input/output* (SUIO) sequence for a state of an FSM is a correctly-ordered UIO sequence for that state. Although an SUIO sequence itself does not cause any synchronization problem, a synchronization problem can still arise if the transition corresponding to the first edge of the SUIO sequence is not an eligible successor of the transition which precedes the SUIO sequence in a given transition sequence. For example, when an SUIO sequence for a state $s$ starts with an input sent by $U$, any incoming transition of state $s$ with label $i^L/o^L$ or $i^L/-$, will cause a synchronization problem if it precedes the SUIO sequence. To avoid a synchronization problem, for each state $s$, two SUIO sequences are needed, denoted by SUIO$^U(s)$ and SUIO$^L(s)$ where SUIO$^U(s)$ starts with an input

sent by $U$ and $\mathrm{SUIO}^L(s)$ starts with an input sent by $L$.

## 3 CONSTRUCTION OF SYNCHRONIZABLE CHECKING SEQUENCES

Let $M = (S, X, Y, \delta, \lambda)$ hereafter stand for a minimal and completely specified FSM which is represented by a strongly connected and order-specified digraph $D = (V, E)$. Let $|S|$ be $n$ and $s_1 \in S$ be the initial state of $M$. The construction of a synchronizable checking sequence of $M$ is based on the construction of a correctly-ordered digraph $D' = (V', E')$ such that all edges of $D$ are in one to one correspondence with edges in $D'$, and all paths in $D'$ are CO paths in $D$. Thus, finding a synchronizable checking sequence on $D$ will be reduced to finding a checking sequence on $D'$.

### 3.1 Construction of a Correctly-ordered Digraph $D'$

The correctly-ordered digraph $D' = (V', E')$ was introduced by Ural (1993), and later modified by Perez (1993). Before $D' = (V', E')$ is constructed, the following sets of edges are formed :
for each vertex $v$ in $D = (V, E)$,

$Leave^U[v] = \{e \in E : head(e) = v \text{ and } label(e) \in \{i^U/-, i^U/o^U, i^U/o^L, i^U/o^{U,L}\}\}$

$Leave^L[v] = \{e \in E : head(e) = v \text{ and } label(e) \in \{i^L/-, i^L/o^L, i^L/o^U, i^L/o^{U,L}\}\}$

$Arrive^U[v] = \{e \in E : tail(e) = v \text{ and } label(e) \in \{i^U/-, i^U/o^U\}\}$

$Arrive^L[v] = \{e \in E : tail(e) = v \text{ and } label(e) \in \{i^L/-, i^L/o^L\}\}$

$Arrive^{U,L}[v] = \{e \in E : tail(e) = v \text{ and } label(e) \in \{i^L/o^U, i^U/o^L, i^L/o^{U,L}, i^U/o^{U,L}\}\}$.

The edges in $Leave^U[v]$ are eligible successors of edges in $Arrive^U[v]$ and $Arrive^{U,L}[v]$, The 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 $Arrive^L[v]$.

During the construction of $D' = (V', E')$ from $D = (V, E)$, for each vertex $v \in V$, 1, 2 or 3 vertices are created in $D'$, depending on the labels of edges arriving and leaving $v$, so that if an edge arrives to a vertex in $V'$, it can take any of the edges starting at this vertex. Thus, a path in $D'$ becomes a CO path in $D$. Accordingly, the correctly-ordered digraph $D' = (V', E')$ is such that $V' = V^U \cup V^L \cup V^{U,L}$ and $E' = E_C \cup F$. The procedure of constructing the correctly-ordered digraph $D' = (V', E')$ from a strongly connected and order-specified digraph $D = (V, E)$ such that any edge in $D$ has an eligible successor is as follows:

(1) For each vertex $v$ in $V$,
If $Leave^U[v] \neq \emptyset$, then a vertex $v^U$ is created in $V^U$.
If $Leave^L[v] \neq \emptyset$, then a vertex $v^L$ is created in $V^L$.

(2) For each edge $(w, v; x/y) \in Arrive^U[v]$ (this implies that $(w, v; x/y) \in Leave^U[w]$), a directed edge from $w^U$ to $v^U$ is created in $E_C$. Similarly, for each edge $(w, v; x/y) \in Arrive^L[v]$ (this implies that $(w, v; x/y) \in Leave^L[w]$), a directed edge from $w^L$ to $v^L$ is created in $E_C$.

(3) For each edge $(w, v; x/y) \in Arrive^{U,L}[v]$, one of the following is performed:

a) In the case that vertex $v^U$ exists but vertex $v^L$ does not,
if $(w, v; x/y) \in Leave^U[w]$, then a directed edge from $w^U$ to $v^U$ is created in $E_C$,
else (i.e. $(w, v; x/y) \in Leave^L[w]$) a directed edge from $w^L$ to $v^U$ is created in $E_C$.

b) In the case that vertex $v^U$ exists but vertex $v^L$ does not,
if $(w, v; x/y) \in Leave^U[w]$, then a directed edge from $w^U$ to $v^L$ is created in $E_C$,
else (i.e. $(w, v; x/y) \in Leave^L[w]$) a directed edge from $w^L$ to $v^L$ is created in $E_C$.

c) In the case that both $v^U$ and $v^L$ exist, a vertex $v^{U,L}$ is created in $V^{U,L}$ (if it does not already exist) and two edges $(v^{U,L}, v^U)$ and $(v^{U,L}, v^L)$ with nil labels, denoted by -/- when needed, are constructed in $F$.
If $(w, v; x/y) \in Leave^U[w]$, then a directed edge $(w^U, v^{U,L}; x/y)$ is created in $E_C$,
else (i.e. $(w, v; x/y) \in Leave^L[w]$) a directed edge $(w^L, v^{U,L}; x/y)$ is created in $E_C$.

## 3.2 Proposed Method

The proposed method for the construction of a synchronizable checking sequence from $M$ is based on the following assumptions:

1) The implementation $N$ of $M$ implements the reset feature of $M$ correctly. A reset transition has label $r/-$ and can be followed by any input from any tester without causing any synchronization problem.
2) For each edge $e$ of $D$, there is a CO-path that starts at the initial vertex $v_1$ and contains $e$.
3) For each state $s$, represented by vertex $v$ in $D$, if $Arrive^U[v]$ (resp. $Arrive^L[v]$) $\neq \emptyset$, then $SUIO^U(s)$ (resp. $SUIO^L(s)$) exists and if $Arrive^{U,L}[v] \neq \emptyset$, then at least one of $SUIO^U(s)$ or $SUIO^L(s)$ exists (which implies that each edge has an eligible successor).
4) There is at most one state $s$ represented by vertex $v$ in $D$, such that $Arrive^{U,L}[v] = \emptyset$ and both $Leave^U[v]$ and $Leave^L[v]$ are not empty.
5) The input portion of each SUIO sequence is a synchronizable input sequence for all states in $M$.

The proposed method utilizes $D' = (V', E')$ constructed from the given $D = (V, E)$ and proceeds as follows :

**step1**: Construct the sets of input sequences corresponding to input portions of SUIO sequences that will be applied to each vertex in $D'$.
$I$ is defined as the set of input sequences corresponding to the input portions of the SUIO sequences of the states of $M$ that have to be used for transition verification. Let $I_{sU}$ and $I_{sL}$ denote the input portions of $SUIO^U(s)$ and $SUIO^L(s)$, respectively. Note that $I_{s\theta}$ will stand for either $I_{sU}$ or $I_{sL}$.

The construction of $I$ is as follows: For each edge $(w, v; x/y)$ in $D$ where $v$ represents $s \in S$, consider the corresponding edge $e$ in $E_C$:

if $tail(e) = v^U$, then $I = I \cup I_{sU}$ and define $I_{vU}$
if $tail(e) = v^L$, then $I = I \cup I_{sL}$ and define $I_{vL}$
if $tail(e) = v^{U,L}$, then mark $e$.

For each marked edge $e$, one of the following four cases will apply:

case 1: $I_{vU} \in I$, $I_{vL} \in I$, then $I_{vU,L}$ = shorter of $\{I_{vU}, I_{vL}\}$

case 2: $I_{vU} \in I$, $I_{vL} \notin I$, then $I_{vU,L} = I_{vU}$.

case 3: $I_{vU} \notin I$, $I_{vL} \in I$, then $I_{vU,L} = I_{vL}$.

case 4: $I_{vU} \notin I$, $I_{vL} \notin I$, then $I = I \cup$ shorter of $\{I_{vU}, I_{vL}\}$ and $I_{vU,L}$ = shorter of $\{I_{vU}, I_{vL}\}$.

Let $m$ denote $|I|$, $I = \{I_1, I_2, \ldots I_m\}$, and $I_i(k)$ denote the $k^{\text{th}}$ input of the input sequence $I_i$, $1 \le i \le m$. Since $D'$ can be viewed as representing an FSM which is not completely specified, only a specific subset of $I$ can be applied to each vertex $v^U$ (resp. $v^L$). This subset of $I$ is called $R_{vU}$ (resp. $R_{vL}$) and defined as follows:

$\forall\, v^\theta \in V^U \cup V^L, \forall\, i, 1 \le i \le m$, if $I_i(1)$ can be applied to $v^\theta$, then $I_i \in R_{v\theta}$.

Note that $|R_{v\theta}|$ is denoted by $m_{v\theta}$, $R_{vU} \cup R_{vL} = I$, $R_{vU} \cap R_{vL} = \varnothing$, $m_{vU} + m_{vL} = m$, the $i^{\text{th}}$ element (which is an input sequence) of $R_{v\theta}$ is denoted by $R_{v\theta}(i)$ and either $R_{vL}$ or $R_{vU}$ may be empty. When a reset input $r$ is applied, the next vertex is $v_{1U,L}$.

**step2:** Construct an input sequence $C$, called *cover* of $D'$, that contains:

a) a state recognition part which is the concatenation of
$rT_{iU}R_{iU}(1)rT_{iU}R_{iU}(2)r\ldots rT_{iU}R_{iU}(m_{iU})rT_{iL}R_{iL}(1)r\ldots rT_{iL}R_{iL}(m_{iL}), \forall i,\ 1 \le i \le n.$
where $T_{iU}$ and $T_{iL}$ are defined as follows:
- if vertex $v_{iU,L}$ exists, $T_{iU,L}$ is the shortest transfer sequence on $D'$ from the initial vertex $v_{1U,L}$ to vertex $v_{iU,L}$ and $T_{iU} = T_{iL} = T_{iU,L}$
- If $v_{iU,L}$ does not exist, $T_{iU}$ (resp. $T_{iL}$) is the shortest transfer sequence on $D'$ from the initial vertex $v_{1U,L}$ to vertex $v_{iU}$ (resp. $v_{iL}$).

b) a transition verification part which consists of the following *test segment*
for each edge $e$ in $E_C$ corresponding to the edge $(w, v; x/y)$ in $D$: $rT_{head(e)}xI_{tail(e)}$
where $T_{head(e)}$ is the same transfer sequence on $D'$ from vertex $v_{1U,L}$ to vertex $head(e)$ that was used in state recognition part.

**Proposition:** Cover $C$ of $D'$ exists.

**Proof:**
By assumption 2, for any edge $e$ of $E$, there is a CO-path starting at the initial vertex and containing $e$. Thus, by the construction of $D'$, for any $e' \in E_C$, there is a path on $D'$ starting at the initial vertex $v_{1U,L}$ and containing $e'$. Every vertex in $V^U \cup V^L$ is the head of some edge in $E_C$ and every vertex in $V^{U,L}$ is the tail of some edge in $E_C$. Therefore, for each vertex $v$ in $V'$, there is a transfer sequence on $D'$ from vertex $v_{1U,L}$ to vertex $v$. Moreover, by assumption 3 and by construction of $I$, every element of $I$ exists and, by assumption 5, every element of $R_{v\theta}$ is the input sequence of a path on $D'$ starting at $v^\theta$. Therefore, all the subsequences of $C$ of the type $rX$ are defined on $D'$, i.e. cover $C$ of $D'$ exists. **EOP.**

**Theorem**: Let $D' = (V', E')$ be a correctly-ordered digraph constructed from an order-specified digraph $D = (V, E)$ representing an FSM $M$ and let cover $C$ of $D'$ be the input portion of an IO-sequence $Q$ which is the label of a path $P'$ starting at any vertex $v$ in $D'$. Then $Q$ is a synchronizable checking sequence of $M$.

**Proof:**
This proof is composed of 3 parts.

First, it must be established that $Q$ is a synchronizable IO-sequence of $M$. Clearly, this is the case since, by construction, cover $C$ is an input sequence on $D'$. Therefore, $Q$ is a CO-sequence as it is the label of a path in $D'$. Hence, $Q$ is a synchronizable IO-sequence of $M$. Second, it must be shown that if $Q$ is also an IO sequence for an implementation $N = (S', X, Y, \delta', \lambda')$ of $\Phi(M)$ then $N$ has $n$ states and $I$ is also a set of SUIO sequences for $N$. In order to show that this is the case, note that, any state of $M$ is in one of the following three disjoint subsets of $S$:

**$S1$**= $\{s_i \in S \mid$ only one of $v_i^U$or $v_i^L \in V'\}$. Without loss of generality, say that $v_i^U \in V'$. Then, since $D'$ is strongly connected, $v_i^U$ is the tail of some edge in $E_C$ and, by construction of $I$, $I_{s_iU}$ is an element of $I$. Therefore, the state recognition for a state of $S1$ is achieved by $rT_{v_iU}I_1rT_{v_iU}I_2r...rT_{v_iU}I_m$

**$S2$** = $\{s_i \in S \mid v_i^{U,L} \in V'\}$. Then, by construction of $D'$, $v_i^{U,L}$ is the tail of some edge in $E_C$ and, by construction of $I$, at least one of $\{I_{s_iU}, I_{s_iL}\}$ is an element of $I$. Therefore, the state recognition for a state of $S2$ is achieved by $rT_{v_iU,L}I_1rT_{v_iU,L}I_2r...rT_{v_iU,L}I_m$.

**$S3$** = $\{s_i \in S \mid v_i^U \in V'$, $v_i^L \in V'$ and $v_i^{U,L} \notin V'\}$. Note that by the 3rd assumption, $|S3| \leq 1$. Then, since $D'$ is strongly connected, $v_i^U$ and $v_i^L$ are tails of some edges in $E_C$ and, by construction of $I$, $I_{s_iU}$ and $I_{s_iL}$ are elements of $I$. Therefore, the state recognition part for a state of $S3$ is achieved by $rT_{v_iU}R_{v_iU}(1)rT_{v_iU}R_{v_iU}(2)r...\ rT_{v_iU}R_{v_iU}(m_{v_iU})rT_{v_iL}R_{v_iL}(1)r\ ...\ rT_{v_iL}R_{v_iL}(m_{v_iL})$, where $T_{v_iU}$ and $T_{v_iL}$ are different.

Since $r$ is correctly implemented in $N$ and $N$ is deterministic, if $T$ is some transfer sequence, $N$ will always be in the same state after application of $rT$. Hence, one can rewrite the state recognition part of the input portion of the IO-sequence as in Table 1, where each column represents the state of $N$ reached by the transfer sequence given as the label of the column, each row represents an element of $I$ given as the label of the row and each cell contains the output produced by the state of $N$ reached by the transfer sequence given as the label of its column after the application of the element of $I$ given as the label of its row. Table 1 summarizes all realizable cases, with states $s_i$ and $s_j$ of $M$ in $S1$, states $s_k$, $s_p$ and $s_q$ of $M$ in $S2$ and state $s_z$ in $S3$ and $I = \{I_{s_iU}, I_{s_jL}, I_{s_kL}, I_{s_pU}, I_{s_qU}, I_{s_qL}, I_{s_zL}, I_{s_zU}\}$.

For any state $s$ of $M$ in $S1 \cup S2$ of $M$, there is a column in Table 1 to which all $m$ elements of $I$ are applied, i.e. every cell of the column has a value. Let us take the $(|S1|+|S2|)$x$m$ subtable containing only the states of $M$ in $S1 \cup S2$. The value of a cell will be denoted $o_{rc}$ with $r$ the row number and $c$ the column number. By construction, $I$ contains at least an SUIO for each state of $M$. Thus, $\forall c_0, \exists r_0$ such that $\forall c, c \neq c_0, o_{r_0c} \neq o_{r_0c_0}$.

Therefore, $N$ has $|S1|+|S2|$ distinct states. To each state $s$ of $M$ in $S1 \cup S2$, there corresponds a unique state $s'$ of $N$, such that $\forall I_i \in I$ , $\lambda'(s', I_i) = \lambda(s, I_i)$. As for the existence of a state in S3, there are two cases:

If $S3 = \varnothing$, then $N$ has exactly $n$ states and $\forall s' \in N, \forall I_i \in I, \lambda'(s', I_i) = \lambda(s, I_i)$

If $S3 \neq \varnothing$, say $S3 = \{s_z\}$, $|S1|+|S2| = n-1$, so $N$ has already $n$-1 determined states.
The last two rows of Table 1 show that the state of $N$ reached after the application of $rT_{v_zU}$ is different from the $n$-1 states of $N$ already determined and, identically, the state reached after the application of $rT_{v_zL}$ is different from the $n$-1 states of $N$ already determined. Since $N$ is in $\Phi(M)$, $N$ has at most $n$ states, thus, after the application of $rT_{v_zU}$ or $rT_{v_zL}$, $N$ is in a unique state $s'_z$ which is the same state as $s_z$. Hence $N$ has exactly $n$ states and the last two columns of Table 1 show that $\forall I_i \in I$ , $\lambda'(s'_z, I_i) = \lambda(s_z, I_i)$.

**Table 1** State Recognition Part

| | $rT_{v_iU}$ | $rT_{v_jL}$ | $rT_{v_kU,L}$ | $rT_{v_pU,L}$ | $rT_{v_qU,L}$ | $rT_{v_zU}$ | $rT_{v_zL}$ |
|---|---|---|---|---|---|---|---|
| $I_{s_iU}$ | $\lambda(s_i,I_{s_iU})$ | $\lambda(s_j,I_{s_iU})\neq \lambda(s_i,I_{s_jU})$ | $\lambda(s_k,I_{s_iU})\neq \lambda(s_i,I_{s_kU})$ | $\lambda(s_p,I_{s_iU})\neq \lambda(s_i,I_{s_pU})$ | $\lambda(s_q,I_{s_iU})\neq \lambda(s_i,I_{s_qU})$ | | $\lambda(s_z,I_{s_iU})\neq \lambda(s_i,I_{s_zU})$ |
| $I_{s_jL}$ | $\lambda(s_i,I_{s_jL})\neq\lambda(s_j,I_{s_jL})$ | $\lambda(s_j,I_{s_jL})$ | $\lambda(s_k,I_{s_jL})\neq \lambda(s_j,I_{s_jL})$ | $\lambda(s_p,I_{s_jL})\neq \lambda(s_j,I_{s_jL})$ | $\lambda(s_q,I_{s_jL})\neq \lambda(s_j,I_{s_jL})$ | | $\lambda(s_z,I_{s_jL})\neq \lambda(s_j,I_{s_jL})$ |
| $I_{s_kL}$ | $\lambda(s_i,I_{s_kL})\neq \lambda(s_k,I_{s_kL})$ | $\lambda(s_j,I_{s_kL})\neq \lambda(s_k,I_{s_kL})$ | $\lambda(s_k,I_{s_kL})$ | $\lambda(s_p,I_{s_kL})\neq \lambda(s_k,I_{s_kL})$ | $\lambda(s_q,I_{s_kL})\neq \lambda(s_k,I_{s_kL})$ | | $\lambda(s_z,I_{s_kL})\neq \lambda(s_k,I_{s_kL})$ |
| $I_{s_pU}$ | $\lambda(s_i,I_{s_pU})\neq \lambda(s_p,I_{s_pU})$ | $\lambda(s_j,I_{s_pU})\neq \lambda(s_p,I_{s_pU})$ | $\lambda(s_k,I_{s_pU})\neq \lambda(s_p,I_{s_pU})$ | $\lambda(s_p,I_{s_pU})$ | $\lambda(s_q,I_{s_pU})\neq \lambda(s_p,I_{s_pU})$ | $\lambda(s_z,I_{s_pU})\neq \lambda(s_p,I_{s_pU})$ | |
| $I_{s_qU}$ | $\lambda(s_i,I_{s_qU})\neq \lambda(s_q,I_{s_qU})$ | $\lambda(s_j,I_{s_qU})\neq \lambda(s_q,I_{s_qU})$ | $\lambda(s_k,I_{s_qU})\neq \lambda(s_q,I_{s_qU})$ | $\lambda(s_p,I_{s_qU})\neq \lambda(s_q,I_{s_qU})$ | $\lambda(s_q,I_{s_qU})$ | $\lambda(s_z,I_{s_qU})\neq \lambda(s_q,I_{s_qU})$ | |
| $I_{s_qL}$ | $\lambda(s_i,I_{s_qL})\neq \lambda(s_q,I_{s_qL})$ | $\lambda(s_j,I_{s_qL})\neq \lambda(s_q,I_{s_qL})$ | $\lambda(s_k,I_{s_qL})\neq \lambda(s_q,I_{s_qL})$ | $\lambda(s_p,I_{s_qL})\neq \lambda(s_q,I_{s_qL})$ | $\lambda(s_q,I_{s_qL})$ | | $\lambda(s_z,I_{s_qL})\neq \lambda(s_q,I_{s_qL})$ |
| $I_{s_zL}$ | $\lambda(s_i,I_{s_zL})\neq \lambda(s_z,I_{s_zL})$ | $\lambda(s_j,I_{s_zL})\neq \lambda(s_z,I_{s_zL})$ | $\lambda(s_k,I_{s_zL})\neq \lambda(s_z,I_{s_zL})$ | $\lambda(s_p,I_{s_zL})\neq \lambda(s_z,I_{s_zL})$ | $\lambda(s_q,I_{s_zL})\neq \lambda(s_z,I_{s_zL})$ | | $\lambda(s_z,I_{s_zL})$ |
| $I_{s_zU}$ | $\lambda(s_i,I_{s_zU})\neq \lambda(s_z,I_{s_zU})$ | $\lambda(s_j,I_{s_zU})\neq \lambda(s_z,I_{s_zU})$ | $\lambda(s_k,I_{s_zU})\neq \lambda(s_z,I_{s_zU})$ | $\lambda(s_p,I_{s_zU})\neq \lambda(s_z,I_{s_zU})$ | $\lambda(s_q,I_{s_zU})\neq \lambda(s_z,I_{s_zU})$ | $\lambda(s_z,I_{s_zU})$ | |

Therefore, the states of $M$ are in one to one correspondence with the states of $N$. If an element of $I$ is the input portion of an SUIO of a state $s$ of $M$, then it is also the input portion of an SUIO of state $s' = f(s)$ of $N$.

In order to complete the proof, the following must also be shown : suppose $Q$ is also an IO-sequence for an implementation $N$ of $\Phi(M)$. Let $(a, b; x/y)$ be an edge of $D$, then $(f(a), f(b); x/y)$ is a transition of $N$, and reciprocally, if $(a', b'; x/y)$ is a transition of $N$, then $(f^{-1}(a'), f^{-1}(b'); x/y)$ is an edge of $D$.

In order to show that this is the case, let $(a, b; x/y)$ be an edge of $D$, then there is a corresponding edge $e$ in $E_C$. The transition verification part is such that $(T_{head(e)}xI_{tail(e)})/\ (\lambda(v_1U,L, T_{head(e)})y\lambda(b, I_{tail(e)}))$ is a subsequence of $Q$ and, by construction, $I$ contains the input portion of the SUIO corresponding to the tail of each edge in $E_C$, thus $I_{tail(e)} \in I$. Then, since $Q$ is an IO-sequence of $N$, $I_{tail(e)}$ is the input portion of an SUIO of state $b' = f(b)$ of $N$. Hence, the state reached by $N$ after the application of $T_{head(e)}x$ is $b' = f(b)$.

Moreover, $I$ contains the input portion of an SUIO for each state of $M$ so $I_{a\theta} \in I$. Then, since $Q$ is an IO-sequence of $N$, $I_{a\theta}$ is the input portion of an SUIO of state $a' = f(a)$ of $N$. The state recognition part is such that $(T_{head(e)}I_{a\theta})\ /\ (\lambda(v_1U,L, T_{head(e)})\lambda(a, I_{a\theta}))$ is a subsequence of $Q$. Hence, the state reached by $N$ after the application of $T_{head(e)}$ is $a' = f(a)$.

Therefore, $(a', b'; x/y)$ is a transition of $N$. This holds for every transition of $M$. Since $M$ is completely specified, $N$ is also completely specified, so for any transition $(a', b'; x/y)$ of $N$, there is a corresponding transition $(f^{-1}(a'), f^{-1}(b'); x/y)$ in $M$. i.e. $N$ is isomorphic to $M$.

For any implementation $N$ of $\Phi(M)$, if $Q$ is an IO-sequence of $N$, then $N$ is isomorphic to $M$. Hence, $Q$ is a checking sequence of $M$. **EOP**

## 4 AN EXAMPLE

Consider the FSM $M$, represented by the digraph depicted in Figure 2, where input $a$ is applied by upper tester $U$ and input $b$ is applied by lower tester $L$. The output 0 is sent to $U$ and output 1 and 2 are sent to $L$.

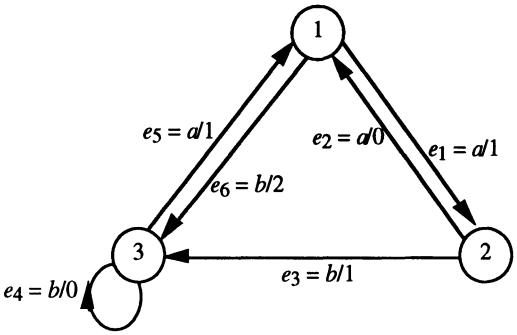


**Figure 2** Digraph $D = (V, E)$.

Edge Types and Order Specifications:
$e_1 = (1, 2)$, label $(e_1) = i^U/o^L$, Eligible Successor: $e_2$, $e_3$
$e_2 = (2, 1)$, label $(e_2) = i^U/o^U$, Eligible Successor: $e_1$
$e_3 = (2, 3)$, label $(e_3) = i^L/o^L$, Eligible Successor: $e_4$
$e_4 = (3, 3)$, label $(e_4) = i^L/o^U$, Eligible Successor: $e_4$, $e_5$
$e_5 = (3, 1)$, label $(e_5) = i^U/o^L$, Eligible Successor: $e_1$, $e_6$
$e_6 = (1, 3)$, label $(e_6) = i^L/o^L$, Eligible Successor: $e_4$

**Table 2** Arrive and Leave Sets for Each Vertex.

| *vertex* | $Leave^U$ | $Leave^L$ | $Arrive^U$ | $Arrive^L$ | $Arrive^{U,L}$ |
|---|---|---|---|---|---|
| 1 | $e_1$ | $e_6$ | $e_2$ | | $e_5$ |
| 2 | $e_2$ | $e_3$ | | | $e_1$ |
| 3 | $e_5$ | $e_4$ | | $e_3$, $e_6$ | $e_4$ |

The construction of the correctly-ordered digraph $D'$ proceeds as follows:
Arrive and Leave sets for all vertices of $M$ are determined as in Table 2.
For each vertex $v$ in $V$, $Leave^U[v]$ and $Leave^L[v]$ are not empty so vertices $1^U$, $1^L$, $2^U$, $2^L$, $3^U$, $3^L$ are created in $V'$. Then every edge of $E$ is considered:

- $e_1 \in Leave^U[1]$ and $Arrive^{U,L}[2]$, then construct $2^{U,L}$ in $V^{U,L}$, edges $(2^{U,L}, 2^U; -/-)$ and $(2^{U,L}, 2^L; -/-)$ in $F$ and edge $e'_1 = (1^U, 2^{U,L}; a/1)$ in $E_c$.
- $e_2 \in Leave^U[2]$ and $Arrive^U[1]$, then construct edge $e'_2 = (2^U, 1^U; a/0)$ in $E_c$.
- $e_3 \in Leave^L[2]$ and $Arrive^L[3]$, then construct edge $e'_3 = (2^L, 3^L; b/1)$ in $E_c$.
- $e_4 \in Leave^L[3]$ and $Arrive^{U,L}[3]$, then construct $3^{U,L}$ in $V^{U,L}$, edges $(3^{U,L}, 3^U; -/-)$ and $(3^{U,L}, 3^L; -/-)$ in $F$ and edge $e'_4 = (3^L, 3^{U,L}; b/0)$ in $E_c$.
- $e_5 \in Leave^U[3]$ and $Arrive^{U,L}[1]$, then construct $1^{U,L}$ in $V^{U,L}$, edges $(1^{U,L}, 1^U; -/-)$ and $(1^{U,L}, 1^L; -/-)$ in $F$ and edge $e'_5 = (3^U, 1^{U,L}; a/1)$ in $E_c$.
- $e_6 \in Leave^L[1]$ and $Arrive^L[3]$, then construct edge $e'_6 = (1^L, 3^L; b/2)$ in $E_c$.

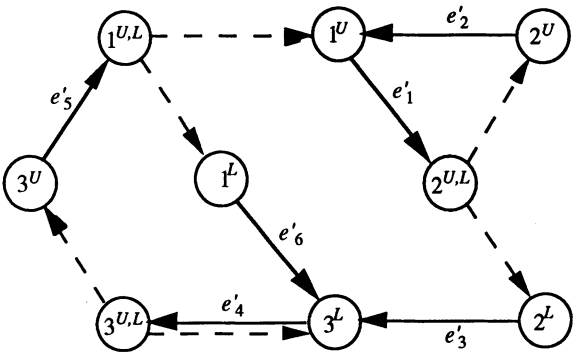


**Figure 3** The Correctly-ordered Digraph $D'$ of $D$ in Figure 2.Edges in $F$ are dashed.

SUIO sequences for each state are:
$SUIO^U(1) = a/1, a/0$ $SUIO^L(1) = b/2$
$SUIO^U(2) = a/0$ $SUIO^L(2) = b/1$
$SUIO^U(3) = a/1, a/1$ $SUIO^L(3) = b/0$

The construction of $I$ proceeds as follows:

For each edge in $E_c$,
$tail(e'_1) \notin V^U \cup V^L$, mark $e'_1$
$tail(e'_2) = 1^U$ so, $I = I \cup I_{1U}$. Hence, $I = \{aa\}$
$tail(e'_3) = 3^L$ so, $I = I \cup I_{3L}$. Hence, $I = I \cup \{b\} = \{aa, b\}$
$tail(e'_4) \notin V^U \cup V^L$, mark $e'_4$
$tail(e'_5) \notin V^U \cup V^L$, mark $e'_5$
$tail(e'_6) = 3^L$, $I = I \cup I_{3L}$. Hence, $I = I \cup \{b\} = \{aa, b\}$.

For each marked edge in $E_c$,
$tail(e'_1) = 2^{U,L}$ and $I_{2U} = a \notin I$, $I_{2L} = b \in I$. Hence, $I_{2U,L} = b$.
$tail(e'_4) = 3^{U,L}$ and $I_{3U} = aa \in I$, $I_{3L} = b \in I$. Hence, $I_{3U,L} = b$.
$tail(e'_5) = 1^{U,L}$ and $I_{1U} = aa \in I$, $I_{1L} = b \in I$. Hence, $I_{1U,L} = b$.
Finally, $I = \{aa, b\}$, $R_{1U} = R_{2U} = R_{3U} = aa$ and $R_{1L} = R_{2L} = R_{3L} = b$

The construction of a cover $C$ of $D'$ proceeds as follows:

1) State Recognition Part:
$T_{1U} = T_{1L} = T_{1U,L} =$ -
$T_{2U} = T_{2L} = T_{2U,L} = a$
$T_{3U} = T_{3L} = T_{3U,L} = bb$
$rT_{1U}\,aarT_{1L}brT_{2U}\,aarT_{2L}brT_{3U}\,aarT_{3L}b = raarbraaarabrbbaarbbb$

2) Transition Verification Part:
test segment($e'_1 = (1^U, 2^{U,L}; a/1)$) $= rT_{1U}\,a\,I_{2U,L} = rab$
which visits vertices $1^{U,L}1^U2^{U,L}2^L3^L$

test segment($e'_2 = (2^U, 1^U; a/0)$) $= rT_{2U}aI_{1U} = raaaa$
which visits vertices $1^{U,L}1^U2^{U,L}2^U1^U2^{U,L}2^U1^U$
test segment($e'_3 = (2^L, 3^L; b/1)$) $= rT_{2L}bI_{3L} = rabb$
which visits vertices $1^{U,L}1^U2^{U,L}2L3L3^{U,L}$
test segment($e'_4 = (3^L, 3^{U,L}; b/0)$) $= rT_{3L}bI_{3U,L} = rbbbb$
which visits vertices $1^{U,L}1L3L3^{U,L}3L3^{U,L}3L3L3^{U,L}$
test segment($e'_5 = (3^U, 1^{U,L}; a/1)$) $= rT_{3U}aI_{1U,L} = rbbab$
which visits vertices $1^{U,L}1L3L3^{U,L}3^U1^{U,L}1L3L$
test segment($e'_6 = (1^L, 3^L; b/2)$) $= rT_{1L}bI_{3L} = rbb$
which visits vertices $1^{U,L}1L3L3^{U,L}$

The input portion $C$ of $Q$ is *raarbraaarabrbbaarbbbrabraaaarabbrbbbbrbbabrbb.*
$Q$ is a synchronizable checking sequence of length 46.

## 5 MINIMIZATION TECHNIQUES

To minimize the sequence $Q$ produced by the proposed method, three complementary approaches can be used:

1) It is clear that the sequence derived in the example can substantially be reduced simply by eliminating the redundant subsequences. i.e. in cover $C$, eliminate $rX$ if there is an $rX'$ such that $X$ is a prefix of $X'$. For example,
$C$ = ***raa****rbraaa****a****rabrbbaarbbbrab****raaaa****rabbrbbbbrbbabrbb* where *raa* and *raaa* can be eliminated since they are prefixes of *raaaa*. This yields
$C$ = ***rb****rabrbbaa****rbbb****rabraaaaarabb****rbbbb****rbbabr****bb*** where *rb*, *rbb* and *rbbb* can be eliminated since they are prefixes of *rbbbb*. This yields
$C$ = ***rab****rbbaa****rab****raaaar****abb****rbbbbrbbab* where *rab* can be eliminated since it is a prefix of *rabb*.
Minimization using this approach yields $C$ = *rbbaaraaaarabbrbbbbrbbab* of length 24.

2) In the example, the same transfer sequences are used in the state recognition and in the transition verification parts. By choosing different transfer sequences, one can increase the possibility of overlapping the test segments or of using shorter transfer sequences to a given state prior to verifying its outgoing edges. For example, since test segment($e'_6$) = $rbI_{3L}$ = *rbb*, one can take transfer sequence $T_{3L}$ =*rb* to test $e'_4$ instead of *rbb*, yielding test segment($e'_4$) = *rbbb*.

3) In the case that both children of a vertex $v^{U,L}$ are in $I$, the method requires that $I_{vU,L}$ should be the shorter of $\{I_{vU}, I_{vL}\}$, it would be better to choose the one that contributes to the greater reduction in the sense of (1). For example, for $e'_4$, there would be two possible test segments : *rbbbb* and *rbbbaa* and for $e'_5$ there also would be two possible test segments : *rbbab* or *rbbaaa*. Choosing test segment($e'_5$) = *rbbaaa* implies that *rbbaa* can be deleted from $C$.

Combining these three approaches yields $C$ = *raaaarabbrbbbrbbaaa* which is of length 19.

## 6 CONCLUSIONS

A method for constructing a synchronizable checking sequence of a given FSM $M$ using UIOs has been proposed. The checking sequence constructed by this method has been shown to be easily reducible by eliminating redundancies and by making a wise choice of the transition sequences, and of the SUIO sequences to apply. The proposed method has been presented on the basis of some assumptions for the given FSM that are discussed hereafter. The first assumption states that

an implementation of the given FSM possesses a correctly implemented reset feature which dramatically reduces the length of the state recognition part. Without this assumption, the resulting synchronizable checking sequence of the FSM will be restrictively long. In addition, the application of a reset breaks the connection in most real protocols, and thus can be utilized to form test cases from the resulting synchronizable checking sequence. The last four assumptions are necessary for any method that will attempt to construct a synchronizable checking sequence of a given FSM using UIOs. The second assumption assures that every transition of the FSM is part of a synchronizable transition sequence starting at the initial state. The third assumption assures that for each transition of the FSM, there is an SUIO synchronizable with the transition so that one can derive a test segment in the transition verification part. The fourth assumption requires that there should be at most one state that receives inputs from both $U$ and $L$ and is reached only by transitions with label $\in \{i^L/-, i^U/-, i^L/o^L, i^U/o^U\}$. The fifth assumption requires that any SUIO should be a synchronizable sequence for all states of the FSM. The last two assumptions assure the uniqueness of the SUIOs in an implementation of the given FSM.

## Acknowledgments

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada under grants STR0149338 and OGP0000976.

## References

Aho, A.V., Dahbura, A.T., Lee, D. and Uyar, M.U. (1991) An optimization technique for protocol conformance test sequence generation based on UIO sequences and rural Chinese postman tours. *IEEE Trans. Comm.*, **39**, 1604-1615.

Chan, W.Y.L., Vuong, S.T. and Ito, M.R. (1989) An improved protocol test generation procedure based on UIOS. in: *Proc. SIGCOMM'89* , 283-294.

Chen, W.H. and Ural, H. (1995) Minimum-cost synchronizable test sequences based on multiple UIOs. *IEEE/ACM Trans. Networking*, **3**, 152-157.

Dahbura, A.T., Sabnani, K.K. and Uyar, M.U. (1990) Formal methods for generating protocol conformance test sequences. *Proc of IEEE,* **78**, 1317-1325.

Gill, A. (1962)*Introduction to the Theory of Finite-State Machines* . McGraw-Hill, New York, .

Hennie, F.C. (1964) Fault detecting experiments for sequential circuits. in: *Proc. Fifth Ann. Symp. Switching Circuit Theory and Logical Design* , 95-110.

ISO TC97/SC21, (1987) OSI conformance testing methodology and framework - Part 1-5, ISO 2nd and DP 9646-1 revised text (edited by D. Rayner), Vancouver, Canada

Lee, D. and Yannakakis, M. (1994) Testing finite state machines: state identification and verification. *IEEE Trans. Comput.,* **43**, 306-320.

Perez, M.E. (1993) Correctly-ordered Postman Tours and Synchronizable Test sequence generation for protocol conformance testing. *Master thesis*, Dept. of CSI, Univ. of Ottawa.

Rayner, D. (1987) OSI conformance testing. *Comput. Networks ISDN Systems.*, **14**, 79-98.

Rezaki, A., Ural, H. and White, G. (1994) Construction of checking sequences based on UIO sequences. *Proc. of ISCIS'94*, 315-326.

Sabnani, K.K. and Dahbura, A.T. (1988) A protocol test generation procedure. *Comput. Networks ISDN Systems.*, **15**, 285-297.

Sarikaya, B. and Bochmann, G.v. (1984) Synchronization and specification issues in protocol testing. *IEEE Trans. Comm.,* **32**, 389-395.

Tai, K.C. and Young, Y.C. (1995) Port-synchronizable test sequences for communications protocols. in this proceedings.

Ural, H. and Wang, Z. (1993) Synchronizable test sequence generation using UIO sequences. *Comput. Comm.*, **16**, 653-661.