Generating Checking Sequences for a Distributed Test Architecture

Hasan Ural and Craig Williams

School of Information Technology and Engineering
University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada
{ural,cwilliam}@site.uottawa.ca

Abstract. The objective of testing is to determine whether an implementation under test conforms to its specification. In distributed test architectures involving multiple testers, this objective can be complicated by the fact that testers may encounter problems relating to controllability and observability during the application of tests. The controllability problem manifests itself when a tester is required to send the current input and because it did not send the previous input nor did it receive the previous output it cannot determine when to send the input. The observability problem manifests itself when a tester is expecting an output in response to either the previous input or the current input and because it is not the sender of the current input, it cannot determine when to start and stop waiting for the output. Based on a distinguishing sequence, a checking sequence construction method is proposed to yield a sequence that is free from controllability and observability problems.

1 Introduction

Determining, under certain assumptions, whether a given "black box" implementation *N* of a Finite State Machine (*FSM*) *M* 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 [4, 7]. This experiment is based on an input sequence called a *checking sequence*, constructed from a given deterministic and minimal FSM *M* with a designated initial state, that determines whether a given FSM *N* is a correct or faulty implementation of *M*. The construction of a checking sequence must deal with the "black box" nature of the given implementation *N*, 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*. To overcome the restrictions imposed by the limited controllability and observability, 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*.

In order to verify the implementation of a transition from state *a* to *b* under input *x*, 1) *N* must be transferred to the state recognized as *a*, 2) the input *x* is applied and the output produced in response by *N* must be as specified in *M*, and 3) the state reached after the application of *x* must be recognized as state *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 (DS) [7], a characterization set [7] or a unique input-output (UIO) sequence [14]. It is known that a distinguishing sequence may not exist for every minimal FSM [11], and that determining the existence of a distinguishing sequence for an FSM is PSPACE-complete [12]. Nevertheless, based on distinguishing sequences, various methods have been proposed in the literature to test FSMs [5, 7, 18, among others].

Testing an implementation *N* of an FSM *M* can be carried out as a fault detection experiment in some specific test architectures. One such architecture is the distributed test architecture shown in Figure 1 [9] where the lower interface and the upper interface of the implementation *N* may be controlled and observed indirectly by the lower tester (*L*) and directly by the upper tester (*U*), respectively. A similar architecture is given in [10].

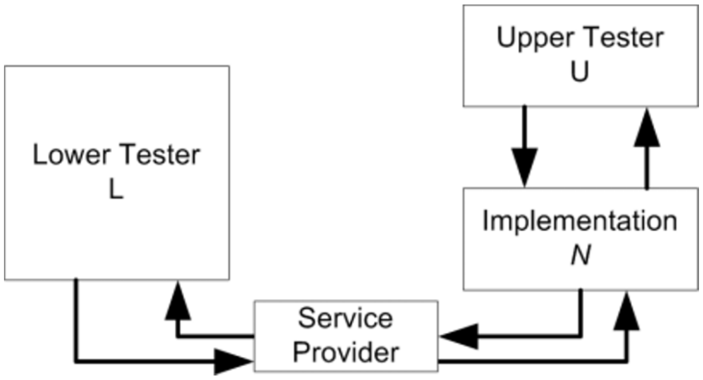

Fig. 1. A Distributed Test Architecture

In this architecture, *U* and *L* are two remote testers that are required to coordinate the application of a preset checking sequence. However, this requirement may lead to controllability and observability problems, in addition to those that stem from the black box nature of *N*. The *controllability* (*synchronization*) *problem* manifests itself 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. It is therefore important to construct a synchronizable checking sequence that causes no controllability problems during its application in the distributed test architecture. For some FSMs, a checking sequence can be constructed such that no two consecutive inputs will cause a controllability problem, and hence the coordination among testers is achieved indirectly through their interactions with *N* [6]. However, for some other FSMs, there may not exist a checking sequence in which the testers can coordinate solely via their interactions with *N* [6]. In this case it is necessary for testers to communicate directly by exchanging external coordination messages over a dedicated channel during the application of the checking sequence. An external coordination message exchange relating to controllability is denoted $<-C_{L(U)}, +C_{U(L)}>$, where "$-C_{L(U)}$" denotes the sending of an external coordination message to tester L(or U) from tester U(or L), and "$+C_{U(L)}$" denotes the receipt of an external coordination message from tester U(or L) by tester L(or U) [2].

During the application of even a synchronizable checking sequence in a distributed test architecture, the observability problem manifests itself when L (or U) is expected to receive an output from N in response to either the previous input or the current input and because L (or U) is not the one to send the current input, L (or U) is unable to determine when to start and stop waiting. Such observability problems hamper the detectability of output shift faults in N i.e., an output associated with the current input is generated by N in response to either the previous input or the next input. To ensure the detectability of output shift faults in N the checking sequence needs to be augmented either by additional input subsequences selected from FSM M or by external coordination message exchanges between testers such that during the application of the checking sequence testers can determine whether the output observed is received in response to the correct input as specified in M. An external coordination message exchange relating to observability is denoted $<-O_{L(U)}, +O_{U(L)}>$, where "$-O_{L(U)}$" denotes the sending of an external coordination message to tester L(or U) from tester U(or L), and "$+O_{U(L)}$" denotes the receipt of an external coordination message from tester U(or L) by tester L(or U) [2].

This paper proposes a method for constructing a checking sequence that does not pose controllability and observability problems during its application in a distributed test architecture. Earlier work on the controllability problem [1, 3, 15, 16, 17] and the observability problem [2, 13, 19, 20] consider the construction of a test sequence rather than a checking sequence, except [6, 8] which use UIO sequences for the construction of a checking sequence. It is well known that the complete fault coverage of a checking sequence cannot always be achieved by a test sequence where transition verification is not necessarily based on state verification.

The rest of the paper is organized as follows: Related terminology is reviewed in Section 2. In Section 3, the proposed method is presented along with an illustrative example. Section 4 concludes with a summary and future research directions.

2 Preliminaries

2.1 FSM and Its Graphical Representation

For ease of presentation and readability, the proposed method will be presented using a 2-port FSM. The generalization of the method to n-port ($n \geq 2$) FSM is simply a matter of adapting a different notation as the one given in [19]. A *2-port Finite State Machine* (2p-FSM) $\boldsymbol{M} = (\boldsymbol{S}, \Sigma, \Gamma, \delta, \lambda, \boldsymbol{s}_1)$ where

- S is a finite set of states of M,
- $s_1 \in S$ is the initial state of M,
- $\Sigma = \Sigma_U \cup \Sigma_L$, where $\Sigma_{U(L)}$ is the input alphabet of port U (L), and $\Sigma_U \cap \Sigma_L = \varnothing$. Let $I = \Sigma_U \cup \Sigma_L \cup \{-\}$, where - means *null* input,
- $\Gamma = \Gamma_U \cup \Gamma_L$, where $\Gamma_{U(L)}$ is the output alphabet of port U (L), and $\Gamma_U \cap \Gamma_L = \varnothing$. Let $O = \{<a_U, a_L> \mid \exists\, a_{U(L)} \in \Gamma_{U(L)} \cup \{-\}\}$, where - means *null* output,
- δ is the transition function that maps $S \times I$ to S, i.e., δ: $S \times I \rightarrow S$, and
- λ is the output function that maps $S \times I$ to O, i.e., λ: $S \times I \rightarrow O$.

Henceforth, a 2p-FSM M or N will be called simply FSM M or N, respectively.

An FSM M is *deterministic* if, for each input $x \in I$, there is at most one transition defined at each state of M. An FSM M is said to be *minimal* if none of its states are equivalent (i.e., $\forall\ s_i, s_j \in S$, $s_i \neq s_j$, $\exists$ an input sequence $X \in I^*$ such that $\lambda(s_i, X) \neq \lambda(s_j,X)$). An FSM M is said to be *completely specified* if, for each input $x \in I$, there is a transition defined at each state of M.

An FSM M can be represented by a directed graph $G = (V, E)$ where a set of vertices V represents the set S of states of M, and a set of directed edges E represents all specified transitions of M. A *transition* of an FSM M is a triple $t_{jk} = (s_j, s_k; x/y)$, where $s_j, s_k \in S$, $x \in I$, and $y \in O$ such that $\delta(s_j, x) = s_k$, $\lambda(s_j, x) = y$, and x/y is known as an *input/output pair*. Each edge $e_{jk} = (v_j, v_k; x/y) \in E$ represents a state transition from state s_j to state s_k with input x and output y where the input/output pair x/y is the *label* of e_{jk}, denoted by *label*(e_{jk}), v_j is called the *head* of e_{jk}, denoted by *head*(e_{jk}), and v_k is called the *tail* of e_{jk}, denoted by *tail*(e_{jk}).

A *path* $P = (v_1, v_2; x_1/y_1)(v_2, v_3; x_2/y_2)\ldots(v_{k-1}, v_k; x_{k-1}/y_{k-1})$, $k>1$, in $G = (V, E)$ is a finite sequence of adjacent (but not necessarily distinct) edges in G, where v_1 and v_k are *head*(P) and *tail*(P), and $x_1/y_1, x_2/y_2, \ldots, x_{k-1}/y_{k-1}$ is the *label* of P, denoted *label*(P). A path P is represented by $(v_1, v_k; X/Y)$ where *label*(P) = X/Y is the *input/output sequence* $(x_1/y_1)(x_2/y_2)\ldots(x_{k-1}/y_{k-1})$, input sequence $X = (x_1x_2\ldots x_{k-1})$ is the *input portion* of X/Y, and output sequence $Y = (y_1y_2\ldots y_{k-1})$ is the *output portion* of X/Y. The *cost* or *length* of each edge of G is equal to the number of input/output pairs in its label. The cost of a path (or length of a path) P in G is the sum of the costs (or lengths) of edges included in P and is denoted *cost*(P). The first transition $(v_1, v_2; x_1/y_1)$ of path P is denoted *first*(P) and the last transition *last*(P). The *concatenation* of a path A and a path B is denoted $A@B$.

A sequence $(i_1 i_2 \ldots i_k)$ is a *subsequence* of $(x_1 x_2 \ldots x_m)$ if there exists a Δ, $0 \leq \Delta \leq m-k$, such that for all j, $1 \leq j \leq k$, $i_j = x_{j+\Delta}$. A sequence $(i_1 i_2 \ldots i_k)$ is a *prefix* of $(x_1 x_2 \ldots x_m)$ if $\forall\ j$, $1 \leq j \leq k$, $i_j = x_j$. An FSM M has a *reset* function if there exists an input $r \in I$ which takes M from any state s_i to the initial state s_1 with a single transition $(s_i, s_1; r/<\text{-},\text{-}>)$.

A digraph $G = (V, E)$ is *strongly connected* if, for any pair of vertices v_j and v_k, there exists a path from v_j to v_k. It is *weakly connected* if its underlying undirected graph is connected. A *tour* of G is a path in G that starts and ends at the same vertex of G. An *Euler tour* of G is a tour that contains every edge of E exactly once. A *postman tour* (PT) of G is a tour that contains every edge in E at least once. A *rural postman tour* (RPT) of G over a set $E_C \subseteq E$ is a tour traversing every edge in E_C at least once. A *Chinese postman tour* (CPT) is a minimum-cost PT. A *rural Chinese postman tour* (RCPT) of G over a set $E_C \subseteq E$ is a minimum-cost RPT over E_C. A *rural postman path* (RPP) from v_i to v_j over $E_C \subseteq E$ is a path from v_i to v_j that includes every edge in E_C. A *rural Chinese postman path* (RCPP) from v_i to v_j over $E_C \subseteq E$ is a minimum-cost RPP.

Given a vertex $v \in V$, *in-degree*(v), is defined as $|\{(u, v; x/y): (u, v; x/y) \in E\}|$ and *out-degree*(v), is defined as $|\{(v, w; x/y): (v, w; x/y) \in E\}|$.

Given an FSM M, an input sequence X is a *distinguishing sequence* (D) if the output sequence Y produced by M in response to X is different for each state. $DS(s_i)$ denotes the transition sequence induced by the application of D at state s_i. A *test seg-*

ment for a transition $t_{ij} = (s_i, s_j; x/y)$ is the transition sequence induced by the application of xD at state s_i.

Given an FSM M, let $\Phi(M)$ be the set of FSMs each of which has at most $|S|$ states and the same input and output sets as M. Let N be an FSM of $\Phi(M)$. N is *isomorphic* to M if there is a one-to-one and onto function f on the state sets of M and N such that for any state transition $(s_i, s_j; x/y)$ of M, $(f(s_i), f(s_j); x/y)$ is a transition of N. A *checking sequence* of M is an input sequence starting at a specific state of M that distinguishes M from any N of $\Phi(M)$ that is not isomorphic to M. In the context of testing, this means that in response to this input sequence, any faulty implementation N will produce an output sequence different than the expected output, thereby indicating the presence of a fault(s).

2.2 Controllability (Synchronization) Problem

Given an FSM M and a global input/output sequence $\omega = x_1/y_1\, x_2/y_2 \ldots x_m/y_m$ of M, where $x_i \in I$ and $y_i \in O$, $1 \le i \le m$, a *controllability (synchronization) problem* occurs when, in the labels x_j/y_j and x_{j+1}/y_{j+1} of any two consecutive transitions, there exists a tester k that sends x_{j+1} that is neither the one sending x_j nor one of those receiving an output belonging to y_j, $1 \le j \le m$-1.

Given an FSM M and a global input/output sequence $\omega = x_1/y_1\, x_2/y_2 \ldots x_m/y_m$ of M, where $x_i \in I$ and $y_i \in O$, $1 \le i \le m$, any two consecutive transitions t_{ij} and t_{jk} whose labels are x_j/y_j and x_{j+1}/y_{j+1} form a *synchronizable pair* of transitions if t_{jk} can follow t_{ij} without causing a synchronization problem. For a transition $t_{ij} = (v_i, v_j; x_j/y_j)$, each transition $t_{jk} = (v_j, v_k; x_k/y_k)$ that forms a synchronizable pair of transitions with t_{ij} is called an *synchronizable successor* of t_{ij}. Any (sub)sequence of transitions in which every pair of transitions is synchronizable is called a *synchronizable transition (sub)sequence*. A global input/output sequence is said to be *synchronizable* if it is the label of a synchronizable transition sequence.

2.3 Observability Problem

Given an FSM M and a global input/output sequence $\omega = x_1/y_1\ x_2/y_2 \ldots x_m/y_m$ of M, where $x_i \in I$ and $y_i \in O$, $1 \le i \le m$, a *1-shift output fault* in an implementation N of M exists when, in the labels x_j/y_j and x_{j+1}/y_{j+1} of any two consecutive transitions, there exists one $a_{L(U)} \in \Gamma_{L(U)}$ in y_j of M which occurs in y_{j+1} in N (and not in y_j in N) or there exists one $a_{L(U)} \in \Gamma_{L(U)}$ in y_{j+1} of M which occurs in y_j in N (and not in y_{j+1} in N), $1 \le j \le m$-1. An instance of the observability problem manifests itself as an *undetectable 1-shift output fault* if there is a 1-shift output fault related to $a_{L(U)} \in \Gamma_{L(U)}$ in any two consecutive transitions whose labels are x_j/y_j and x_{j+1}/y_{j+1}, such that tester $L(U)$ satisfies the condition ($a_{L(U)}$ is in y_j XOR $a_{L(U)}$ is in y_{j+1}) AND $x_{j+1} \notin \Sigma_{L(U)}$. In this case, we say that tester $L(U)$ is *involved* in the shift, and would not be able to detect it.

3 The Proposed Method

Let $M = (S, \Sigma, \Gamma, \delta, \lambda, s_1)$ hereafter stand for a minimal, (in)completely specified FSM which is represented by a strongly connected digraph $G = (V, E)$ and has a distinguishing sequence D. Let $|S|$ be n and $s_1 \in S$ be the initial state of M. It is assumed that any implementation N of M correctly implements a reset function which takes M from any state s_i to the initial state s_1 with a single transition $(s_i, s_1; r/<\text{-},\text{-}>)$. The construction of a synchronizable global checking sequence of M is based on the construction of two sets which represent all potential controllability and observability problems for M. A state cover and transition cover are generated, and an auxiliary graph G'' representing these sequences is constructed. A rural Chinese postman path on G'' yields a checking sequence which can be applied in a distributed test architecture without encountering controllability or observability problems.

3.1 Identifying Controllability and Observability Problems

In the first phase of the proposed method, the set of all controllability problems, T_C, and the set of all observability problems, T_O, are generated from the digraph $G = (V, E)$ of the given FSM M. The set T_C is constructed as follows:

```
for each vertex $v_j \in V$ do
    for each edge $e_{ij}$ (say $t_{ij}$) = $(v_i, v_j; x_j/y_j)$ entering vertex $v_j$ do
        for each edge $e_{jk}$ (say $t_{jk}$)= $(v_j, v_k; x_{j+1}/y_{j+1})$ leaving vertex $v_j$ do
            if $x_j \in \Sigma_U$ AND $x_{j+1} \in \Sigma_L$ AND $a_L$ = - in $y_j$
                then add $(t_{ij}, t_{jk}; <-C_L, +C_U>)$ to $T_C$
            else if $x_j \in \Sigma_L$ AND $x_{j+1} \in \Sigma_U$ AND $a_U$ = - in $y_j$
                then add $(t_{ij}, t_{jk}; <-C_U, +C_L>)$ to $T_C$
```

Each transition pair added to T_C forms a controllability problem as the sender of x_{j+1} is not the sender of x_j and does not receive an output in y_j. Given a path P on G representing a sequence of transitions, the sequence can be made synchronizable as follows: For each pair of consecutive transitions t_{mn} t_{no} in P, if $(t_{mn}, t_{no}; <-C_{U(L)}, +C_{L(U)}>) \in T_C$ then insert the external coordination message exchange $<-C_{U(L)}, +C_{L(U)}>$ relating to controllability between t_{mn} and t_{no} in the label of P.

Consider the FSM $M1$ shown in Figure 2. Applying the above procedure results in a set consisting of 1 non-synchronizable transition pair, i.e., $T_C = \{(t3, t10, <-C_U, +C_L>)\}$.

The set T_O of all triples corresponding to transition pairs with a potential undetectable 1-shift output fault is generated next. The set T_O is constructed from $G = (V, E)$ as follows:

```
for each vertex $v_j \in V$ do
    for each edge $e_{ij}$ (say $t_{ij}$) = $(v_i, v_j; x_j/y_j)$ entering vertex $v_j$ do
        for each edge $e_{jk}$ (say $t_{jk}$)= $(v_j, v_k; x_{j+1}/y_{j+1})$ leaving vertex $v_j$ do
            if for output $a_{L(U)} \in \Gamma_{L(U)}$, $a_{L(U)}$ is in $y_j$ XOR $a_{L(U)}$ is in $y_{j+1}$ AND $x_{j+1} \notin \Sigma_{L(U)}$
            AND $(t_{ij}, t_{jk}; <-C_{U(L)}, +C_{L(U)}>) \notin T_C$
                then add $(t_{ij}, t_{jk}, <-O_{L(U)}, +O_{U(L)}>)$ to $T_O$,
```

where $U(L)$ is the tester sending the input x_{j+1} in t_{jk} and $L(U)$ is the tester involved in the shift.

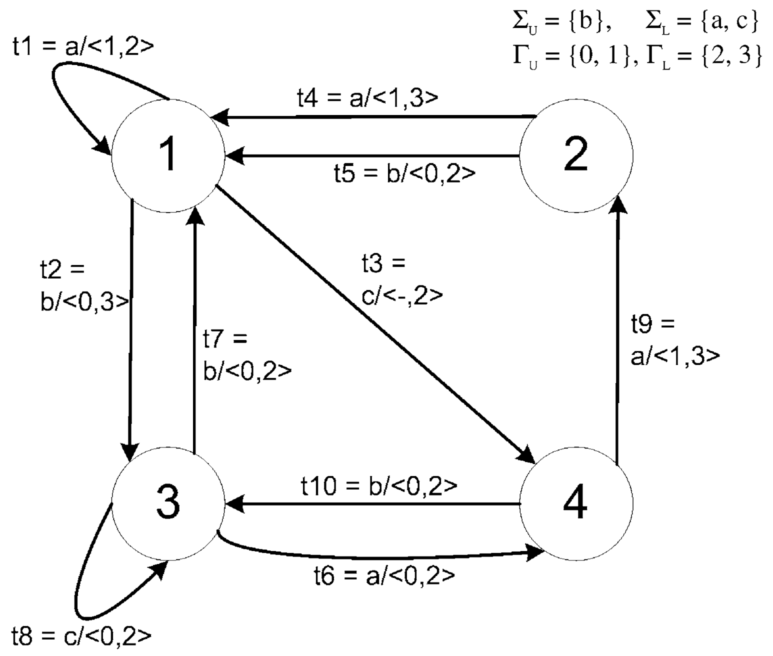

Fig. 2. Digraph $G = (V, E)$ of $2p$-FSM $M1$

The set T_0 identifies *only the necessary subset* of all potential undetectable 1-shift output faults in G as defined in Section 2.3. Specifically, the *if-statement* in the algorithm limits T_0 to only those transition pairs that form a synchronizable pair of transitions in G. Given the input and output alphabets shown in Figure 2, consider a pair of consecutive transitions $t_{ij} = (s_i, s_j; a/<\text{-},2>)$ and $t_{jk} = (s_j, s_k; b/<0,\text{-}>)$. This transition pair forms a potential undetectable forward shift fault of the output '2', i.e. L cannot determine whether '2' is output by a correctly implemented t_{ij}, or by faulty implementations of both t_{ij} and t_{jk}, i.e., $t_{ij} = (s_i, s_j; a/<\text{-},\text{-}>)$ and $t_{jk} = (s_j, s_k; b/<0,2>)$. However, note that any instance of t_{ij} followed by t_{jk} would not form a synchronizable pair of transitions and hence would require the insertion of an external coordination message exchange $<\text{-}C_U,+C_L>$ relating to controllability. If we justifiably assume that L waits to receive the '2' from t_{ij} before sending the external coordination message $-C_U$ relating to controllability to U, the observability problem is resolved; i.e. if L waits and does not receive '2' before it sends $-C_U$, we conclude that the implementation of t_{ij} is faulty. A similar argument and intuitive treatment does not apply in any case of backward shifts if the output <-,-> is not allowed for any transition, other than the reset transitions.

Applying the above procedure to FSM $M1$ produces a set of 5 potential undetectable 1-shift output faults, i.e.: $T_0 = \{(t1, t3, <\text{-}O_U, +O_L>), (t3, t9, <\text{-}O_U, +O_L>), (t4, t3, <\text{-}O_U, +O_L>), (\text{t5}, \text{t3}, <\text{-O}_U, +\text{O}_L>), (\text{t7}, \text{t3}, <\text{-O}_U, +\text{O}_L>)\}$.

3.2 Construction of Test Segments

The second phase of the proposed method generates a test segment $test(t_{ij}) = t_{ij}@DS(s_j)$ for each transition t_{ij}, where $DS(s_i)$ is the transition sequence induced by D on G at v_i. Observability and controllability problems within $DS(s_i)$ and between t_{ij} and $first(DS(s_i))$ are resolved by inserting the corresponding external coordination message from T_O or T_C, respectively. Formally, this phase consists of two steps:

Step 1. For each state s_i, find the transition sequence $DS(s_i)$ induced by D on G at v_i. For each pair of consecutive transitions $t_{mn}\ t_{no}$ in $DS(s_i)$:

- If $(t_{mn}, t_{no}, <-O_{U(L)}, +O_{L(U)}>) \in T_O$ then insert the external coordination message exchange $<-O_{U(L)}, +O_{L(U)}>$ between t_{mn} and t_{no} in $label(DS(s_i))$ and remove $(t_{mn}, t_{no}, <-O_{U(L)}, +O_{L(U)}>)$ from T_O.
- If $(t_{mn}, t_{no}, <-C_{U(L)}, +C_{L(U)}>) \in T_C$ then insert the external coordination message exchange $<-C_{U(L)}, +C_{L(U)}>$ between t_{mn} and t_{no} in $label(DS(s_i))$.

Step 2. For each transition $t_{ij} = (v_i, v_j; x_j/y_j)$:

- Construct $test(t_{ij}) = t_{ij}@DS(s_j)$.
- If there is a triple $(t_{mn}, t_{no}, <-O_{U(L)}, +O_{L(U)}>) \in T_O$ where $t_{mn} = t_{ij}$ and $t_{no} = first(DS(s_j))$ then insert the external coordination message exchange $<-O_{U(L)}, +O_{L(U)}>$ between t_{ij} and $DS(s_j)$ in label($test(t_{ij})$) and remove $(t_{mn}, t_{no}, <-O_{U(L)}, +O_{L(U)}>)$ from T_O.
- If there is a triple $(t_{mn}, t_{no}, <-C_{U(L)}, +C_{L(U)}>) \in T_C$ where $t_{mn} = t_{ij}$ and $t_{no} = first(DS(s_j))$ then insert the external coordination message exchange $<-C_{U(L)}, +C_{L(U)}>$ between t_{ij} and $DS(s_j)$ in $label(test(t_{ij}))$.

The distinguishing sequence D for FSM $M1$is ab. The transition sequences induced by this D at each state are shown in Table 1, and the test segments generated in step 2 are shown in Table 2. Note that Step 2 removes triple $(t3, t9, <-O_U, +O_L>)$ from T_O as each potential 1-shift output fault needs only be handled once. Removing triples ensures that the remaining phases of the proposed method do not unnecessarily avoid transition pairs whose potential 1-shift output fault is already handled in some $test(t_{ij})$.

Table 1. $label(DS(s_i))$ for 2p-FSM $M1$

State s_i	$label(DS(s_i))$
s_1	$t1\ t2$
s_2	$t4\ t2$
s_3	$t6\ t10$
s_4	$t9\ t5$

3.3 Selection of Preambles

This phase of the proposed method generates a preamble for each state s_i, which is a transition sequence that transfers M from the initial state s_1 to state s_i. In choosing a preamble for state s_i, denoted $preamble(s_i)$, three goals must be considered:

Table 2. *label*(*test*(t_{ij})) for 2*p*-FSM *M*1

Transition t_{ij}	*label*(*test*(t_{ij}))
*t*1	*t*1 *t*1 *t*2
*t*2	*t*2 *t*6 *t*10
*t*3	*t*3 $<\text{-}O_U, +O_L>$ *t*9 *t*5
*t*4	*t*4 *t*1 *t*2
*t*5	*t*5 *t*1 *t*2
*t*6	*t*6 *t*9 *t*5
*t*7	*t*7 *t*1 *t*2
*t*8	*t*8 *t*6 *t*10
*t*9	*t*9 *t*4 *t*2
*t*10	*t*10 *t*6 *t*10

1) Minimize observability and controllability problems in *preamble*(s_i)
2) Minimizing observability and controllability problems between the last transition of *preamble*(s_i) and transitions starting at s_i.
3) Minimize the length of *preamble*(s_i)

These goals may conflict; the preamble for s_i that requires the fewest external coordination message exchanges may end with a transition which causes significant problems when followed by the transitions starting at s_i. As our goal is to minimize the number of external coordination message exchanges introduced by the use of *preamble*(s_i), goals 1 and 2 are given precedence and both these goals are considered in choosing *preamble*(s_i). This is accomplished by first calculating a cost for a transition t_{ij} based on the number of controllability and observability problems that will be introduced if a transition sequence ending with transition t_{ij} is chosen as *preamble*(s_j). Using these costs, preambles for each state $s_j \neq s_1$ are found by first constructing a graph $G' = (V', E')$. Edges in E' are assigned a cost based on the controllability and observability problems caused by transition pairs represented by adjacent edges. For each state s_j, a graph G_j is created from G'. An RCPT over a selected edge in G_j selects the preamble for state s_j that causes the fewest controllability and observability problems in the resulting state cover and transition cover sequences. Formally, this phase proceeds as follows:

Step 1. The sum of the external coordination message exchange costs for each transition t_{ij} (that may be the last transition in a preamble for state s_j) followed by any of its adjacent transitions is calculated as follows:

```
for each vertex $v_j \in V$, $v_j \neq v_1$, do
    for each edge $e_{ij}$ (say $t_{ij}$) = $(v_i, v_j; x/y_j)$ entering vertex $v_j$ where $v_i \neq v_j$ do
        let sum_cost($t_{ij}$) = 0
        for each edge $e_{jk}$ (say $t_{jk}$)= $(v_j, v_k; x_{j+1}/y_{j+1})$ leaving vertex $v_j$ do
            if ($t_{ij}, t_{jk}, <\text{-}C_{U(L)}, +C_{L(U)}>$) $\in T_C$ then
                if $t_{jk}$ = first(DS($s_j$))
```

```
                    then $sum\_cost(t_{ij}) = sum\_cost(t_{ij}) + 2w$ (as the pair $t_{ij}\,t_{jk}$ will
                    occur twice, once in verifying $s_j$ and once to verify $t_{jk}$)
                    else $sum\_cost(t_{ij}) = sum\_cost(t_{ij}) + w$
            if $(t_{ij}, t_{jk}, <\text{-}O_{U(L)}, +O_{L(U)}>) \in T_O$
                then $sum\_cost(t_{ij}) = sum\_cost(t_{ij}) + w$
```

Step 2. Construct the graph $G' = (V', E')$ from $G = (V, E)$ by the following steps:

```
    create a vertex $v_1$ in $V'$
    for each edge $e_{1k}$ (say $t_{1k}$) = $(v_1, v_k; x_k/y_k)$ leaving vertex $v_1 \in V$ where $v_k \neq v_1$ do
        create a vertex labelled "$v_1$-$t_{1k}$-$v_k$" in $V'$
        add an edge from $v_1$ to "$v_1$-$t_{1k}$-$v_k$" labelled $t_{1k}$, i.e., $(v_1, v_1\text{-}t_{1k}\text{-}v_k; t_{1k})$
        let $cost(t_{1k}) = 1$
    for each vertex $v_j \in V$, $v_j \neq v_1$
        for each edge $e_{ij}$ (say $t_{ij}$) = $(v_i, v_j; x_j/y_j)$ entering vertex $v_j \in V$, $head(e_{ij}) \neq v_j$:
            for each edge $e_{jk}$ (say $t_{jk}$) = $(v_j, v_k; x_k/y_k)$ leaving vertex $v_j \in V$,
                $tail(e_{jk}) \neq v_j$, $tail(e_{jk}) \neq v_1$:
                create a vertex labelled "$v_i$-$t_{ij}$-$v_j$" in $V'$ if it doesn't exist already
                create a vertex labelled "$v_j$-$t_{jk}$-$v_k$" in $V'$ if it doesn't exist already
                add an edge $e'_{jk} = (v_i\text{-}t_{ij}\text{-}v_j, v_j\text{-}t_{jk}\text{-}v_k; t_{jk})$ in $E'$
                if $\exists$ a triple $(t_{ij}, t_{jk}, <\text{-}O_{U(L)}, +O_{L(U)}>) \in T_O$ then $cost(e'_{jk}) = 1 + w$
                if $\exists$ a triple $(t_{ij}, t_{jk}, <\text{-}C_{U(L)}, +C_{L(U)}>) \in T_C$
                    then $cost(e'_{jk}) = 1 + w*(m + 1)$
                    else $cost(e'_{jk}) = 1$
```

Step 3. For each state $s_j \neq s_1$, create a graph $G_j = (V_j, E_j)$ from $G' = (V', E')$ as follows:

```
        Initially, $G_j = G'$
        create a vertex $v'$ in $V_j$ and add an edge $Z = (v', v_1; \text{"Z"})$ to $E_j$
        for each vertex labelled "$v_i$-$t_{ij}$-$v_j$" in $V'$
            add a dashed edge $e'_{ij} = (v_i\text{-}t_{ij}\text{-}v_j, v'; \text{-})$ to $E_j$
            let $cost(e'_{ij}) = sum\_cost(t_{ij})$ (as calculated in step 1)
```

Step 4. For each state $s_j \neq s_1$: Find an RCPT P of G_j, starting at v_1, over the single edge Z. Let $preamble(s_j) = label(P')$ where P' is the subpath of P from v_1 to v_i-t_{ij}-v_j.

For each pair of consecutive transitions $t_{mn}\,t_{no}$ in $preamble(s_j)$:

- if $(t_{mn}, t_{no}, <\text{-}C_{U(L)}, +C_{L(U)}>) \in T_C$ then insert the external coordination message exchange $<\text{-}C_{U(L)}, +C_{L(U)}>$ between t_{mn} and t_{no} in $label(preamble(s_j))$.
- if $(t_{mn}, t_{no}, <\text{-}O_{U(L)}, +O_{L(U)}>) \in T_O$ then insert the external coordination message exchange $<\text{-}O_{U(L)}, +O_{L(U)}>$ between t_{mn} and t_{no} in $label(preamble(s_j))$.

In Step 2, the cost of each edge leaving v_1 is 1. The cost of every remaining edge e'_{jk} in E' depends on whether the transition pair $(v_i, v_j; x_j/y_j)$ $(v_j, v_k; x_k/y_k)$ appears in T_O or T_C. The variable w represents a high cost to be associated with external coordination message exchanges. The cost of $w*(m + 1)$ for a controllability problem is based on the following: The purpose of G' is to aid in choosing preambles which minimize the number of external coordination message exchanges required in the resulting sequences forming the state cover and transition cover. Each preamble will be used once to form a state cover sequence for state s_i, and m times to verify each of the m transitions starting at s_i. Therefore if a transition pair in a preamble contains a controllability problem, the resulting number of external coordination message exchanges

Table 3. *sum_cost*(t_{ij}) for transition t_{ij}

Transition t_{ij}	*sum_cost*(t_{ij})
*t*2	0
*t*6	0
*t*10	0
*t*3	*w*
*t*9	0

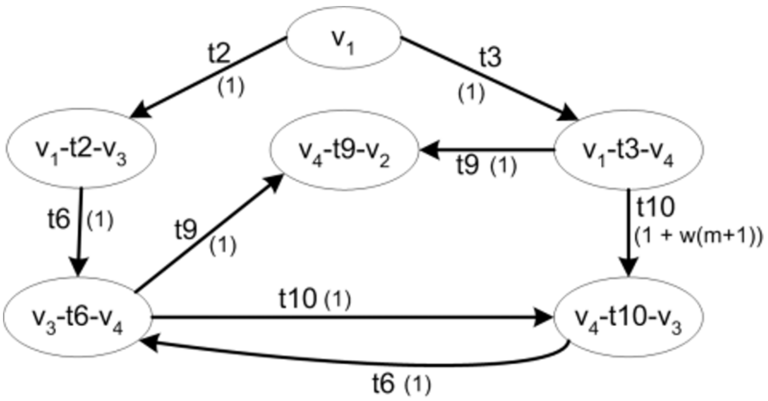

Fig. 3. Digraph $G' = (V', E')$ of 2*p*-FSM *M*1

introduced is $(m + 1)$, so the cost assigned in E' is $w*(m + 1)$. When constructing the graph G_j, the outdegree of node v_j in G is then substituted for m. In contrast, observability problems result in a cost of w as each potential 1-shift output fault need only be checked once. If the pair $(v_i, v_j; x_j/y_j)$ $(v_j, v_k; x_k/y_k)$ does not appear in a triple in T_O or T_C, the cost of edge e'_{jk} is 1.

For FSM *M*1, Step 1 calculates the costs shown in Table 3. Note that the transitions not included are those entering state s_1, as s_1 does not require a preamble, and those that start and end at the same state.

Applying Step 2 to FSM *M*1 generates the graph $G' = (V', E')$ shown in Figure 3, with the cost of each edge shown in parentheses. Based on graphs G_2, G_3, and G_4, preambles selected for states s_2, s_3, s_4, of FSM *M*1 are *t*3 *t*9, *t*2, and *t*2 *t*6, respectively. As an example, the Digraph $G_3 = (V_3, E_3)$ created for s_3 is shown in Figure 4.

3.4 Generating the Checking Sequence

The final phase of the proposed method first generates sequences that form the state cover and transition cover, using the preambles found in the previous phase. Following prefix elimination, an RCPP on an auxiliary graph G'' yields a synchronizable ordering of the remaining sequences. The input portion of the transition sequence

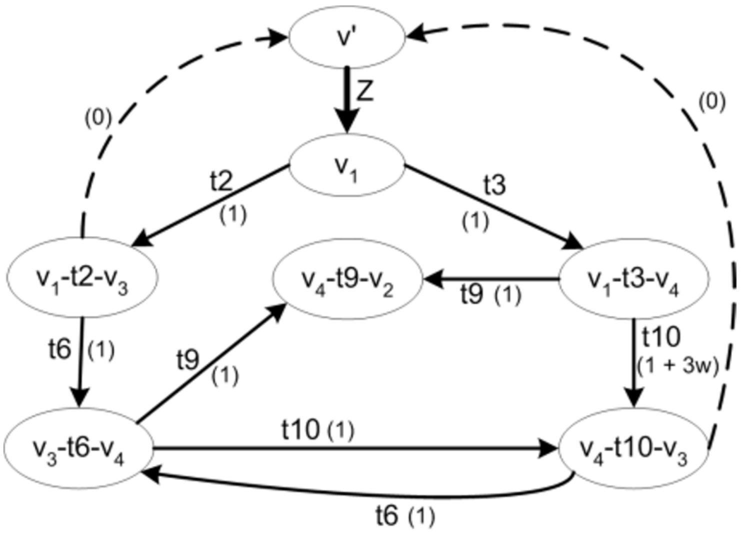

Fig. 4. Digraph $G_3 = (V_3, E_3)$ of $2p$-FSM $M1$

represented by the label of this path is a synchronizable checking sequence with no potential undetectable 1-shift output faults for FSM M. This phase proceeds as follows:

Step 1. For each state s_j:

- Let $state\_cover(s_j) = preamble(s_j)@DS(s_j)$
- If $\exists$ a triple $(t_{mn}, t_{no}, <-C_{U(L)}, +C_{L(U)}>) \in T_C$ where $t_{mn} = last(preamble(s_j))$ and $t_{no} = first(DS(s_j))$ then insert the external coordination message exchange $<-C_{U(L)}, +C_{L(U)}>$ between t_{mn} and t_{no} in $label(state\_cover(s_j))$.
- If $\exists$ a triple $(t_{mn}, t_{no}, <-O_{U(L)}, +O_{L(U)}>) \in T_O$ where $t_{mn} = last(preamble(s_j))$ and $t_{no} = first(DS(s_j))$ then insert the external coordination message exchange $<-O_{U(L)}, +O_{L(U)}>$ between t_{mn} and t_{no} in $label(state\_cover(s_j))$.

Step 2. For each transition $t_{jk} = (s_j, s_k; x/y)$:

- Let $trans\_cover(t_{jk}) = preamble(s_j)@test(t_{jk})$
- If $\exists$ a triple $(t_{mn}, t_{no}, <-C_{U(L)}, +C_{L(U)}>) \in T_C$ where $t_{mn} = last(preamble(s_j))$ and $t_{no} = t_{jk}$ then insert the external coordination message exchange $<-C_{U(L)}, +C_{L(U)}>$ between t_{mn} and t_{no} in $label(trans\_cover(t_{jk}))$.
- If $\exists$ a triple $(t_{mn}, t_{no}, <-O_{U(L)}, +O_{L(U)}>) \in T_O$ where $t_{mn} = last(preamble(s_j))$ and $t_{no} = t_{jk}$ then insert the external coordination message exchange $<-O_{U(L)}, +O_{L(U)}>$ between t_{mn} and t_{no} in $label(trans\_cover(t_{jk}))$.

Step 3. Let C be the set of all transition sequences in the state and transition covers. For every transition sequence $c_p \in C$, if c_p is a prefix of some c_q, $c_q \in C$, $c_p \neq c_q$, then for any external coordination message exchange $<-O_{U(L)}, +O_{L(U)}>$ relating to observability between transitions t_{mn} and t_{no} in label(c_p), insert this observability message between t_{mn} and t_{no} in label(c_q). Then eliminate c_p.

Step 4. For the m sequences $c_1, \ldots, c_m$ remaining in C, let $l_i = label(c_i)$, $1 \leq i \leq m$.
For every subsequence $label(t_{mn})$ $<\text{-}O_{U(L)}, +O_{L(U)}>$ $label(t_{no})$ in any l_i, remove $<\text{-}O_{U(L)}, +O_{L(U)}>$ from any subsequent occurrence of this subsequence in any l_j.

In the following steps, h is the tester sending the *last* input of D, and h' is the other tester.

Step 5. Create four vertices in V'' labelled 1^U, 1^L, T^B, and T^h.

Step 6. For each sequence $c_i \in C$, a solid edge is added to E'' as follows:
- $(1^U, T^h; l_i)$, if sender of x in $first(c_i)$ is U, and $last(c_i)$ sends output only to h,
- $(1^U, T^B; l_i)$, if sender of x in $first(c_i)$ is U, and $last(c_i)$ sends output to h',
- $(1^L, T^h; l_i)$, if sender of x in $first(c_i)$ is L, and $last(c_i)$ sends output only to h,
- $(1^L, T^B; l_i)$, if sender of x in $first(c_i)$ is L, and $last(c_i)$ sends output to h'.

Step 7. Add the following dashed edges to E'' representing reset transitions:
- $(T^B, 1^U$; "rfU/-") and $(T^B, 1^L$; "rfL/-")
- $(T^h, 1^h$; "rf$h$/-")

Step 8. Add a dashed edge $(1^h, 1^{h'}; <\text{-}C_{h'}, +C_h>)$ to E''.
After this step is complete, the resulting digraph will be known as $G'' = (V'', E'')$.

Step 9. Beginning at $1^{h'}$, find a rural Chinese postman path (RCPP) over the solid edges in E''. The input portion of the label of this path represents a checking sequence for FSM M.

Step 10. Eliminate any external coordination message exchanges in the resulting checking sequence that relate to potential 1-shift output faults that can be rendered detectable by some subsequence in the checking sequence, as in [13].

The first two steps of this phase generate the state and transition covers, respectively. In the prefix elimination in Step 3, a transition sequence c_p is eliminated if it is a prefix (without considering external coordination message exchanges) of some other sequence c_q. However, if $label(c_p)$ contains external coordination messages relating to observability, these messages are first copied into $label(c_q)$ to ensure that all potential undetectable output shift faults remain detectable. Redundant external coordination message exchanges relating to observability are removed in Step 4.

In Step 5, vertices 1^L and 1^U are created as all sequences in C begin at the initial state with input from U or L. Step 6 adds solid edges representing sequences $c_1, \ldots, c_m$. Edges that terminate at T^B can be followed by a reset input from either tester. If the last transition of c_i sends output only to h, the edge terminates at T^h and may only be followed by a reset input from h. In Step 7, dashed edges representing these reset inputs from U and L are added to E'' as 'rfU/-' and 'rfL/-' respectively. The dashed edge $(1^h, 1^{h'}; <\text{-}C_{h'}, +C_h>)$ added in Step 8 represents an external coordination message exchange relating to controllability that may be used during the construction of the rural Chinese postman path over the set of solid edges in E''.

Applying steps 1 and 2 to the example FSM $M1$ yield the state and transition covers shown in Table 4 and Table 5, respectively.

Table 4. State cover for 2*p*-FSM *M*1

State s_i	*label*(*state_cover*(s_i))
s_1	*t*1 *t*2
s_2	*t*3 *t*9 *t*4 *t*2
s_3	*t*2 *t*6 *t*10
s_4	*t*2 *t*6 *t*9 *t*5

Table 5. Transition cover for 2*p*-FSM *M*1

Transition t_{ij}	*label*(*trans_cover*(t_{ij}))
*t*1	*t*1 *t*1 *t*2
*t*2	*t*2 *t*6 *t*10
*t*3	*t*3 <$-O_U$, $+O_L$> *t*9 *t*5
*t*4	*t*3 *t*9 *t*4 *t*1 *t*2
*t*5	*t*3 *t*9 *t*5 *t*1 *t*2
*t*6	*t*2 *t*6 *t*9 *t*5
*t*7	*t*2 *t*7 *t*1 *t*2
*t*8	*t*2 *t*8 *t*6 *t*10
*t*9	*t*2 *t*6 *t*9 *t*4 *t*2
*t*10	*t*2 *t*6 *t*10 *t*6 *t*10

Table 6. Sequences eliminated in Step 3 for 2*p*-FSM *M*1

Sequence Eliminated	Prefix of
state_cover(s_3)	*trans_cover*(*t*10)
state_cover(s_4)	*trans_cover*(*t*6)
trans_cover(*t*2)	*trans_cover*(*t*10)
trans_cover(*t*3)	*trans_cover*(*t*5)

In Step 3, 4 sequences are eliminated as shown in Table 6. The set of 10 remaining sequences is shown in Table 7; note that the observability message in the label of the eliminated sequence *trans_cover*(*t*3) has been copied into l_5 in Step 4.

Figure 5 shows the digraph G″ for *M*1 obtained by Step 5 to Step 8. A rural Chinese postman path over the solid edges obtained in Step 9 yields a synchronizable checking sequence, with no potential undetectable 1-shift output faults, represented on G″ by the input portion of the path represented by the label sequence:

rfL/- l_1 rfL/- l_2 rfL/- l_3 rfL/- l_4 rfL/- l_5 rfU/- l_6 rfU/- l_7 rfU/- l_8 rfU/- l_9 rfU/- l_{10}

The input portion of the path represented by this label sequence on G″ corresponds to a checking sequence composed of 10 reset inputs, 41 non-reset inputs, 1 external coordination message exchange relating to observability, and no external coordination message exchanges relating to controllability.

Table 7. Checking sequence subsequences for 2*p*-FSM *M*1

Label	Corresponding Sequence	Verifies
l_1	*t*1 *t*2	s_1
l_2	*t*3 *t*9 *t*4 *t*2	s_2
l_3	*t*1 *t*1 *t*2	*t*1
l_4	*t*3 *t*9 *t*4 *t*1 *t*2	*t*4
l_5	*t*3 <-O_U, +O_L> *t*9 *t*5 *t*1 *t*2	*t*3, *t*5
l_6	*t*2 *t*6 *t*9 *t*5	s_4, *t*6
l_7	*t*2 *t*7 *t*1 *t*2	*t*7
l_8	*t*2 *t*8 *t*6 *t*10	*t*8
l_9	*t*2 *t*6 *t*9 *t*4 *t*2	*t*9
l_{10}	*t*2 *t*6 *t*10 *t*6 *t*10	s_3, *t*2, *t*10

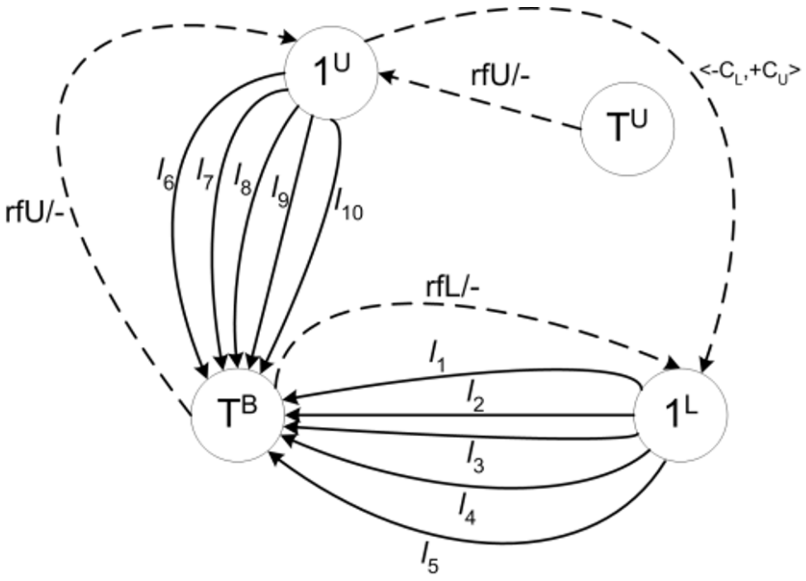

Fig. 5. Digraph *G″* = (*V″*, *E″*) for 2*p*-FSM *M*1

In Step 10, the subsequence *t*6 *t*9 *t*5 in the checking sequence above is found to be sufficient to detect the potential backward shift of ‘1’ in *t*3 *t*9, thus rendering the external coordination message exchange <-O_U, +O_L> between *t*3 and *t*9 unnecessary. As a result, the checking sequence generated by our method for FSM *M*1 requires no external coordination message exchanges relating to controllability and observability.

4 Concluding Remarks

A method for constructing a checking sequence of a given 2*p*-FSM *M* using a distinguishing sequence has been proposed. The resulting checking sequence does not pose controllability and observability problems during its application in a distributed test architecture. The method can easily be generalized to *np*-FSMs, $n \geq 2$, by adapting a different notation as the one given in [19].

One alternative approach is to first generate a checking sequence using the D-method [18] and then identify controllability and observability problems and insert external coordination message exchanges relating to controllability and observability. In the D-method, the shortest path from s_1 to s_j is chosen as *preamble*(s_j). However, this may introduce controllability and/or observability problems that are avoided by our proposed method. For example, applying the D-method to the example FSM *M*1 yields a checking sequence composed of 10 reset inputs, 38 non-reset inputs, 1 external coordination message exchange relating to observability, and 1 external coordination message exchange relating to controllability. As a result of the controllability problem, this checking sequence requires a test architecture which supports direct communication among testers, while the checking sequence generated by our method for this same FSM does not. In general, for a given FSM and a given distinguishing sequence, if there exists a set of checking sequences that do not require any external coordination message exchanges relating to controllability or observability, then one such checking sequence will be selected by the proposed method.

Our method would tie the D-method in terms of the number of non-reset inputs for FSM *M*1 if triples in T_O were not removed during the construction of test segments in Section 3.2. In that case, the preamble *t*2 *t*6 *t*9 would be chosen for s_2 instead of *t*3 *t*9; while this is a longer preamble, prefix elimination results in a checking sequence of only 38 non-reset inputs and no external coordination message exchanges relating to controllability and observability. Unfortunately, considering the effect of prefix elimination during the selection of preambles would be computationally expensive. The same remark applies to the D-method as there may be numerous shortest paths to some states. Another computationally expensive issue stems from the possible existence of more than one distinguishing sequence for a given FSM. It is therefore recognized that our heuristic approach may not yield the shortest synchronizable checking sequence for a given FSM.

A given implementation *N* may not implement the reset function correctly. A method for generating checking sequences for a distributed test architecture that does not assume the correct implementation of the reset function is part of our current research.

Acknowledgments

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada under grant OGP00000976 and Post Graduate Studies Scholarship Program.

References

1. S. Boyd and H. Ural, “The synchronization problem in protocol testing and its complexity,” *Information Processing Letters*, vol. 40, pp. 131-136, 1991.
2. L. Cacciari and O. Rafiq, “Controllability and observability in distributed testing,” *Information and Software Technology*, vol. 41, pp. 767-780, 1999.
3. W. Chen and H. Ural, “Synchronizable checking sequences based on multiple UIO sequences,” *IEEE/ACM Transactions on Networking*, vol 3, pp. 152-157, 1995.
4. A. Gill, *Intro. to the Theory of Finite-State Machines*, New York: McGraw-Hill, 1962.
5. G. Gonenc, “A method for the design of fault detection experiments”, *IEEE Trans. on Computers*, vol. 19, pp. 551-558, 1970.
6. S. Guyot and H. Ural, “Synchronizable checking sequences based on UIO sequences,” Proc. *IFIP IWPTS’95*, Evry, France, 395-407, Sept. 1995.
7. F.C. Hennie, "Fault detecting experiments for sequential circuits", *Proc. Fifth Ann. Symp. Switching Circuit Theory and Logical Design*, pp. 95-110, Princeton, N.J., 1964.
8. R.M. Hierons and H. Ural, “UIO sequence based checking sequences for distributed test architectures”, Accepted for publication in *JIST*.
9. ISO/IEC Information technology – Opens Systems Interconnection – Conformance testing methodology and framework, 9646-1, Part 1: General Concepts, 1995.
10. ISO/IEC Open Distributed Processing, Reference Model, 10748, Parts 1-4, 1995.
11. Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill, Inc.: New York, N.Y.
12. D. Lee and M. Yannakakis, “Testing finite state machines: State identification and verification,” *IEEE Transactions on Computers*, vol. 43, pp. 306-320, 1994.
13. G. Luo, R. Dssouli, G. v. Bochmann, P. Venkataram and A. Ghedamsi, “Test generation with respect to distributed interfaces,” *Computer Standards and Interfaces*, vol. 16, pp. 119-132, 1994.
14. K.K. Sabnani and A.T. Dahbura, “A protocol test generation procedure,” *Computer Networks*, vol. 15, pp. 285-297, 1988.
15. B. Sarikaya and G. v. Bochmann, “Synchronization and specification issues in protocol testing,” *IEEE Transactions on Communications*, vol. 32, pp. 389-395, Apr. 1984.
16. K.C. Tai and Y.C. Young, “Synchronizable test sequences of finite state machines,” *Computer Networks*, vol. 13, pp. 1111-1134, 1998.
17. H. Ural and Z. Wang, “Synchronizable test sequence generation using UIO sequences,” *Computer Communications*, vol.16, pp. 653-661, 1993.
18. H.Ural, X. Wu and F. Zhang, “On minimizing the lengths of checking sequences,” *IEEE Transactions on Computers*, vol. 46, pp. 93-99, 1997.
19. D. Whittier, “Solutions to Controllability and Observability Problems in Distributed Testing,” Master’s thesis, University of Ottawa, Canada, 2001.
20. Y.C. Young and K.C. Tai, “Observation inaccuracy in conformance testing with multiple testers,” Proc. *IEEE WASET*, 80-85, 1998.