Resolving Observability Problems in Distributed Test Architectures

J. Chen[1], R.M. Hierons[2], and H. Ural[3]

[1] School of Computer Science, University of Windsor,
Windsor, Ontario, Canada N9B 3P4
xjchen@uwindsor.ca

[2] Department of Information Systems and Computing, Brunel University,
Uxbridge, Middlesex, UB8 3PH United Kingdom
rob.hierons@brunel.ac.uk

[3] School of Information Technology and Engineering, University of Ottawa,
Ottawa, Ontario, Canada K1N 6N5
ural@site.ottawa.ca

Abstract. The introduction of multiple remote testers to apply a test or checking sequence in a test architecture brings out the possibility of controllability and observability problems. These problems often require the use of external coordination message exchanges among testers. In this paper, we consider constructing a test or checking sequence from the specification of the system under test such that it will be free from these problems and will not require the use of external coordination messages. We give an algorithm that can check whether it is possible to construct subsequences from a given specification that eliminate the need for using external coordination message exchanges, and when it is possible actually produces such subsequences.

Keywords: Finite state machine, testing, test architecture, observability, controllability.

1 Introduction

In a distributed test architecture, a tester is placed at each port of the system under test (SUT) N to apply an input sequence constructed from the specification M of N. When N is a state based system whose externally observable behaviour is specified as a finite state machine (FSM) M, the input sequence applied to N is called a test sequence [13,14] or a checking sequence [6,8,10]. The application of a test/checking sequence in the distributed test architecture introduces the possibility of controllability and observability problems. These problems occur if a tester cannot determine either when to apply a particular input to N, or whether a particular output from N has been generated in response to a specific input, respectively [12].

It is nesessary to construct a test or checking sequence that causes no controllability or observability problems during its application in a distributed test architecture (see, for example, [1,5,7,9,11,15–17]). For some specifications, there exists such an input sequence in which the coordination among testers is achieved indirectly via their interactions with N [14,12]. However, for some other specifications, there may not exist an input sequence in which the testers can coordinate solely via their interactions with N [1,15]. In this case it is necessary for testers to communicate directly by exchanging external coordination messages among themselves over a dedicated channel during the application of the input sequence [2].

It is argued that both controllability and observability problems may be overcome through the use of external coordination messages among remote testers [2]. However, there is often a cost associated with the use of such messages which is composed of the cost of setting up the infrastructure required to allow the exchange of such messages and the cost of delays introduced by exchanging these messages. It is thus desirable to construct a test or checking sequence from the specification of the system under test such that it will not cause controllability and observability problems and will not require the use of external coordination message exchanges.

In [4] we have given a necessary and sufficient condition so that each transition *involved in a potentially undetectable output shift fault* can be *independently verified at port p*. By *verified at port p*, we mean we are able to conclude that the output of this transition at port p is correct according to the correct output sequence of a certain transition path. By *indepedently*, we mean that the above conclusion on the output at port p of each transition does not rely on the correctness of any other transitions. Independence here can be helpful for fault diagnoses: in the case that the system under test contains only undetectable output shift faults, we will be able to identify them. In [3] we have given a necessary and sufficient condition so that each transition involved in a potentially undetectable output shift fault *and has a non-empty output at port p* can be *independently verified at port p*. Based on this we can conclude that each transition involved in a potentially undetectable output shift fault can be *verified at port p*. In this way, we have a weaker condition than that of [4] but we will no more be able to diagnose the undetectable output shift faults: in the case that the system under test contains only undetectable output shift faults, we can only identify those incorrect *non-empty outputs* at port p. In this paper, we do not consider the fault diagnosis problem and we show that in this context, we can have more specifications than those satisfying the conditions in [4] or [3] with which we can construct a subsequence for each transition involved in a potentially undetectable output shift fault so that we can conclude that the outputs at port p of these transitions are correct according to the correct output sequences of the constructed subsequences. We present an algorithm that identifies whether a given specification falls in this category and when it does so constructs the subsequences.

The rest of the paper is organized as follows. Section 2 introduces the preliminary terminology. Section 3 gives a formal definition of the problem and identifies the condition that the specification of the system under test is checked against. Section 4 presents an algorithm for constructing subsequences that eliminate the need for using external coordination messages, proves the correctness of the algorithm, and gives its computational complexity. Section 5 discusses the related work. Section 6 gives the concluding remarks.

2 Preliminaries

An *n-port Finite State Machine* M (simply called an FSM M) is defined as $M = (S, I, O, \delta, \lambda, s_0)$ where S is a finite set of states of M; $s_0 \in S$ is the initial state of M; $I = \bigcup_{i=1}^{n} I_i$, where I_i is the input alphabet of port i, and $I_i \cap I_j = \emptyset$ for $i, j \in [1, n]$, $i \neq j$; $O = \prod_{i=1}^{n} (O_i \cup \{-\})$, where O_i is the output alphabet of port i, and $-$ means null output; δ is the transition function that maps $S \times I$ to S; and λ is the output function that maps $S \times I$ to O. Each $y \in O$ is a *vector of outputs*, i.e., $y =< o_1, o_2, ..., o_n >$ where $o_i \in O_i \cup \{-\}$ for $i \in [1, n]$. We use $*$ to denote any possible output, including $-$, at a port and $+$ to denote non-empty output. We also use $*$ to denote any possible input or any possible vector of outputs. In the following, $p \in [1, n]$ is a port. A *transition* of an FSM M is a triple $t = (s_1, s_2; x/y)$, where $s_1, s_2 \in S$, $x \in I$, and $y \in O$ such that $\delta(s_1, x) = s_2$, $\lambda(s_1, x) = y$. s_1 and s_2 are called the *starting state* and the *ending state* of t respectively. The *input/output pair* x/y is called the *label* of t and t will also be denoted as $s_1 \xrightarrow{x/y} s_2$. p will denote a port and we use $y\mid_p$ or $t\mid_p$ to denote the output at p in output vector y or in transition t respectively. We use T to denote the set of all transitions in M.

A *path* $\rho = t_1\ t_2\ \ldots\ t_k$ $(k \geq 0)$ is a finite sequence of transitions such that for $k \geq 2$, the ending state of t_i is the starting state of t_{i+1} for all $i \in [1, k-1]$. When the ending state of the last transition of path ρ_1 is the starting state of the first transition of path ρ_2, we use $\rho_1 @ \rho_2$ to denote the *concatenation* of ρ_1 and ρ_2. The *label* of a path $(s_1, s_2, x_1/y_1)$ $(s_2, s_3, x_2/y_2)$ $\ldots$ $(s_k, s_{k+1}, x_k/y_k)$ $(k \geq 1)$ is the sequence of input/output pairs $x_1/y_1\ x_2/y_2\ \ldots\ x_k/y_k$ which is an *input/output sequence*. The *input portion* of a path $(s_1, s_2, x_1/y_1)$ $(s_2, s_3, x_2/y_2)$ $\ldots$ $(s_k, s_{k+1}, x_k/y_k)$ $(k \geq 1)$ is the input sequence $x_1 x_2 \ldots x_k$. We say t is *contained in* ρ if t is a transition along path ρ.

When ρ is non-empty, we use *first*(ρ) and *last*(ρ) to denote the first and last transitions of path ρ respectively and *pre*(ρ) to denote the path obtained from ρ by removing its last transition.

We will use 2-port FSMs to show some examples. In a 2-port FSM, ports U and L stand for the upper interface and the lower interface of the FSM. An output vector $y = \langle o_1, o_2 \rangle$ on the label of a transition of the 2-port FSM is a pair of outputs with $o_1 \in O_1$ at U and $o_2 \in O_2$ at L.

Given an FSM M and an input/output sequence $x_1/y_1\ x_2/y_2\ \ldots\ x_k/y_k$ of M a *controllability* (also called *synchronization*) *problem* occurs when, in the labels x_i/y_i and x_{i+1}/y_{i+1} of two consecutive transitions, there exists $p \in [1, n]$ such that $x_{i+1} \in I_p$, $x_i \notin I_p$, $y_i\mid_p = -$ $(i \in [1, k-1])$. If this controllability problem occurs then the tester at p does not know when to send x_{i+1} and the test/checking sequence cannot be applied. Consecutive transitions t_i and t_{i+1} form a *synchro-*

nizable pair of transitions if t_{i+1} can follow t_i without causing a synchronization problem. Any path in which every pair of transitions is synchronizable is called a *synchronizable path*. An input/output sequence is *synchronizable* if it is the label of a synchronizable path.

We assume that for every pair of transitions (t, t') there is a synchronizable path that starts with t and ends with t'. If this condition does not hold, then the FSM is called *intrinsically non-synchronizable* [1].

A *same-port-output-cycle* in an FSM is a path $(s_1, s_2, x_1/y_1)$ $(s_2, s_3, x_2/y_2)$ $\ldots$ $(s_k, s_{k+1}, x_k/y_k)$ $(k \geq 2)$ such that $s_1 = s_{k+1}$, $s_i \neq s_{i+1}$ for $i \in [1, k]$, and there exists a port p with $y_i \mid_p \neq -$ and $x_i \notin I_p$ for all $i \in [1, k]$. An *isolated-port-cycle* in an FSM is a path $(s_1, s_2, x_1/y_1)$ $(s_2, s_3, x_2/y_2)$ $\ldots$ $(s_k, s_{k+1}, x_k/y_k)$ $(k \geq 2)$ such that $s_1 = s_{k+1}$, $s_i \neq s_{i+1}$ for $i \in [1, k]$, and there exists a port p with $y_i \mid_p = -$ and $x_i \notin I_p$ for all $i \in [1, k]$.

A transition t is involved in a potentially undetectable output shift fault at p if and only if there exists a transition t' and a transition path ρ such that at least one of the following holds.

1. $t\rho t'$ is a synchronizable path, no transition in $\rho t'$ contains input at p, the ouputs at p in all transitions contained in ρ are empty, and $t \mid_p = - \Leftrightarrow t' \mid_p \neq -$. In this case an undetectable output shift fault can occur between t and t' in $t\rho t'$. If $t \mid_p = -$ we call it a *backward* output shift fault and if $t \mid_p \neq -$ we call it a *forward* output shift fault.
2. $t'\rho t$ is a synchronizable path, no transition in ρt contains input at p, the ouputs at p in all transitions contained in ρ are empty, and $t \mid_p = - \Leftrightarrow t' \mid_p \neq -$. In this case an undetectable output shift fault can occur between t and t' in $t'\rho t$. If $t \mid_p = -$ we call it a *forward* output shift fault and if $t \mid_p \neq -$ we call it a *backward* output shift fault.

When ρ is empty, we also say that t is involved in a potentially undetectable *1-shift* output fault.

The observability problem occurs when we have potentially undetectable output shift faults in the specification of the FSM.

We will use $\mathcal{T}_p$ to denote the set of transitions that are involved in potentially undetectable output shift faults at p. Let $\mathcal{T}'_p = \mathcal{T}_p \cap \{t \mid t|_p \neq -\}$. $\mathcal{T}'_p$ denotes the set of transitions that are involved in potentially undetectable output shift fault at p and whose output at p are non-empty.

A *relation* R between elements of a set A and elements of a set B is a subset of $A \times B$. If (a, b) is an element of relation R then a is related to b under R and we also write aRb. The set of elements related to $a \in A$ under R is denoted $R(a)$ and thus $R(a) = \{b \in B | (a, b) \in R\}$.

Given a set A, a relation R between A and A is a *partial order* if it satisfies the following conditions.

1. For all $a \in A$, aRa.
2. If aRa' and $a'Ra$ then $a = a'$.
3. If a_1Ra_2 and a_2Ra_3 then a_1Ra_3.

3 Verifiability of Outputs

To verify the output of transition t at port p, we search for a path ρ containing t such that

- ρ is synchronizable;
- we are able to determine the output sequence of ρ at p from applying the input portion of ρ from the starting state of ρ;
- from the correct output sequence of ρ at p we can determine that the output of t at p is correct.

We require that *first*(ρ) and *last*(ρ) have input at p in order to identify a certain output sequence: no matter how ρ is concatenated with other subsequences, we can always determine the output sequence produced at p in response to the first $|pre(\rho)|$ inputs of ρ since this output sequence is immediately preceded and followed by input at p.

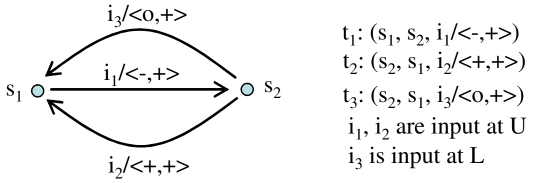
To determine the correct output of (t, p) from the correct output sequence of ρ at p, we require that

- If the output of (t, p) is nonempty, then all the outputs at p in $pre(\rho)$ are either also nonempty or already known to be correct.
- If the output of (t, p) is empty, then all the outputs at p in $pre(\rho)$ are either also empty or already known to be correct.

Example 1. In the given specification in Figure 1, there is an undetectable output shift fault in t_1t_3 at port U, because the input of t_3 is not at U while there is a potential output shift of o from t_3 to t_1. We are interested in constructing a path to verify that the output of transition t_1 and that of t_3 at this port are correct.

$\rho_1 = t_1t_2$ is such a synchronizable path for t_1: it has input at U in t_1 (*first*(ρ)) and input at U in t_2 (*last*(ρ)), and according to the output at U between these two inputs when ρ_1 is applied as a subsequence, we are able to verify that the output of t_1 at U is correct.

If we know that the output of t_1 at U is correct, then $\rho_2 = t_1t_3t_1$ is also a desirable synchronizable path for t_2: it has input at U in t_1 (for both *first*(ρ) and *last*(ρ)), and according to the output at U between these two inputs when ρ_2 is applied as a subsequence, we are able to verify that the output of t_2 at U is correct since we already know that the output of t_1 at U is correct.



i_3/<o,+>
i_1/<-,+>
s_1
s_2
i_2/<+,+>
t_1: (s_1, s_2, i_1/<-,+>)
t_2: (s_2, s_1, i_2/<+,+>)
t_3: (s_2, s_1, i_3/<o,+>)
i_1, i_2 are input at U
i_3 is input at L

Fig. 1. An example where $\mathcal{T}_p$ is verifiable at U

Formally, we introduce the following concept.

Definition 1. *Let t be a transition, and v a set of transitions in M. ρ is* an absolute verifying path upon v for (t,p) *if*

- *ρ is a synchronizable path;*
- *t is contained in pre(ρ);*
- *first(ρ) and last(ρ) and only these two transitions in ρ have input at p;*
- *$t \notin v$ and for all t' contained in pre(ρ), either $t' \in v$ or $t' |_p = - \Leftrightarrow t |_p = -$.*

Note that given t and ρ we will typically consider a minimal *set v that satisfies the above conditions: if $t' |_p = - \Leftrightarrow t |_p = -$ then $t' \notin v$.*

Example 2. In Example 1,

- $t_1 t_2$ is an absolute verifying path upon $\emptyset$ for (t_1, U).
- $t_1 t_3 t_1$ is an absolute verifying path upon $\{t_1\}$ for (t_3, U).

Directly from this definition, we have:

Proposition 1. *If ρ is an absolute verifying path upon v for (t,p) and v is a minimal such set, then ρ is an absolute verifying path upon v for (t',p) for any t' contained in* pre(ρ) *such that $t' |_p = - \Leftrightarrow t |_p = -$.*

Proposition 2. *Let v be a set of transitions in M, ρ an absolute verifying path upon v for (t,p). If for every transition t' in v, the output at p of t' in the SUT is correct, then the correct output sequence at p in response to the first* $|\text{pre}(\rho)|$ *inputs of ρ implies the correct output of (t,p).*

Proof. Suppose $t |_p \neq -$ (The proof for the case when $t |_p = -$ is analogous).

Suppose that m inputs from $pre(\rho)$ lead to non-empty output at p in M. Thus, if we observe the correct output sequence in response to the first $|pre(\rho)|$ inputs of ρ then we must observe m outputs at p in response to these inputs.

Since $t |_p \neq -$, and ρ is an absolute verifying path upon v for (t,p), we know by definition that for all t' in ρ' such that $t' |_p = -$, the output of t' at p is correct (and so is $-$) in the SUT. So, we know that the corresponding $|pre(\rho)| - m$ inputs in $pre(\rho)$ lead to empty output at p. Thus we can map the observed outputs at p, in response to the input portion of $pre(\rho)$, to the inputs that caused them and so if the correct output sequence is observed then the output of p at t must be correct.

To verify the output of (t,p), we try to find a path ρ that is an absolute verifying path upon v for (t,p) for some set v such that the output at p for every transition in v is verified. So in general, we search for an acyclic digraph of transitions such that each transition in this digraph has an absolute verifying path upon a set of transitions that appear as its successors in the digraph. Such an acyclic graph can be represented as a partial order in the following way.

Definition 2. *Suppose that $\mathcal{U}$ is a set of transitions of M, $\mathcal{R}$ is a relation from $\mathcal{U}$ to $\mathcal{U}$, and $\mathcal{P}$ is a function from $\mathcal{U}$ to synchronizable paths of M. Let p be any port in M. The set $\mathcal{U}$ of transitions is verifiable at p under $\mathcal{R}$ and $\mathcal{P}$ if the following hold.*

(a) For all $t \in \mathcal{U}$, $\mathcal{P}(t)$ is an absolute verifying path upon $\mathcal{R}(t)$ for (t, p);
(b) $\mathcal{R} \cup \{(t,t)|t \in \mathcal{U}\}$ is a partial order.

Where such $\mathcal{R}$ and $\mathcal{P}$ exist we also say that $\mathcal{U}$ is verifiable at p.

Suppose that $\mathcal{U}$ is verifiable at p under $\mathcal{R}$ and $\mathcal{P}$ and we observe correct output sequence corresponding to the first $|pre(\mathcal{P}(t))|$ output of $\mathcal{P}(t)$ for each $t \in \mathcal{U}$. Then according to Proposition 2, we know that the output of t at p is correct for each $t \in \mathcal{U}$. So our goal is to find a set $\mathcal{U}$ that is verifiable at p such that $\mathcal{T}_p \subseteq \mathcal{U}$.

Example 3. In Example 1, for port U, we have $\mathcal{T}_U = \{t_1, t_3\}$. $\mathcal{T}_U$ is verifiable at U because

– t_1t_2 is an absolute verifying path upon $\emptyset$ for (t_1, U).
– $t_1t_3t_1$ is an absolute verifying path upon $\{t_1\}$ for (t_3, U).

So let $\mathcal{P}(t_1) = t_1t_2$, $\mathcal{P}(t_3) = t_1t_3t_1$, $\mathcal{R}(t_1) = \emptyset$, $\mathcal{R}(t_3) = \{t_1\}$ (i.e. $\mathcal{R} = \{(t_3, t_1)\}$), then $\mathcal{T}_p = \{t_1, t_3\}$ is verifiable at U under $\mathcal{P}$ and $\mathcal{R}$.

Proposition 3. *If ρ is an absolute verifying path upon v for (t, p) and v is a minimal such set then $v \subseteq \mathcal{T}_p$.*

Proof. Let $\rho = t_1 \ldots t_k$ (for $k \geq 2$) where $t = t_i$ for some $i \in [1, k-1]$. Suppose $t_i \mid_p \neq -$ (the case for $t_i \mid_p = -$ is analogous). Consider an arbitrary transition $t' \in v$: it is sufficient to prove that $t' \in \mathcal{T}_p$.

By the minimality of v we have t' is contained in $pre(\rho)$ and so $t' = t_j$ for some $j \in [1, k-1]$. Since ρ is an absolute verifying path upon v for (t_i, p), $t_i \notin v$ and so $j \neq i$. Suppose $i < j$ (the case for $i > j$ is analogous).

Since $t_j \in v$, by the minimality of v we have that $t_j \mid_p = -$. Now as $i < j$, $t_i \mid_p \neq -$, $t_j \mid_p = -$, there exists some maximal l with $i \leq l < j$ such that $t_l \mid_p \neq -$. Let $\rho' = t_l \ldots t_j$. By Definition 1, no transition in ρ' has input at p. By considering ρ' we see that $t_j \in \mathcal{T}_p$.

This result allows us to consider only transitions in $\mathcal{T}_p$ for $\mathcal{U}$.

Proposition 4. *Suppose M is an FSM that is not intrinsically non-synchronizable, p is a port of M and $\mathcal{U}$ is a set of transitions verifiable at port p. If $\mathcal{T}'_p \subseteq \mathcal{U}$ or $\mathcal{T}_p - \mathcal{T}'_p \subseteq \mathcal{U}$, then $\mathcal{T}_p$ is verifiable at p.*

Proof. Suppose $\mathcal{U}$ is verifiable under $\mathcal{R}$ and $\mathcal{P}$ and that $\mathcal{R}$ is a minimal such relation (i.e. $\mathcal{U}$ is not verifiable using a relation that contains fewer pairs).

First, consider the case that $\mathcal{T}'_p \subseteq \mathcal{U}$. According to Theorem 2 in [3], there exists an absolute verifying path upon $\mathcal{T}'_p$ for (t, p) for every $t \notin \mathcal{T}'_p$. Since $\mathcal{T}'_p \subseteq \mathcal{U}$, there exists $\rho'_{p,t}$, the absolute verifying path upon $\mathcal{T}'_p$ for (t, p), for $t \in \mathcal{T}_p - \mathcal{U}$. Now define relation $\mathcal{R}'$ and function $\mathcal{P}'$ in the following way.

1. $\mathcal{R}' = \mathcal{R} \cup \{(t, t') | t \in \mathcal{T}_p - \mathcal{U} \wedge t' \in \mathcal{T}'_p\}$
2. $\mathcal{P}' = \mathcal{P} \cup \{(t, \rho'_{p,t}) | t \in \mathcal{T}_p - \mathcal{U}\}$

It is easy to check that $\mathcal{T}_p$ is verifiable at p under $\mathcal{R}'$ and $\mathcal{P}'$ as required.

Now consider the case that $\mathcal{T} - \mathcal{T}'_p \subseteq \mathcal{U}$. Similar to Theorem 2 in [3], we can prove that there exists an absolute verifying path upon $\mathcal{T}_p - \mathcal{T}'_p$ for (t, p) for every $t \notin \mathcal{T} - \mathcal{T}'_p$. The proof is then similar to that for the case where $\mathcal{T}'_p \subseteq \mathcal{U}$.

4 Algorithm

To calculate $\mathcal{T}'_p$ and $\mathcal{T}_p - \mathcal{T}'_p$, we can first determine all transitions involved in potentially undetectable *1-shift* output fault. This can be done by comparing every two transitions $s_1 \xrightarrow{x_1/y_1} s_2$ and $s_2 \xrightarrow{x_2/y_2} s_3$ where x_2 is not at p. If y_1 has nonempty output at p while y_2 does not, or vice versa, then t_1 and t_2 are involved in potentially undetectable 1-shift output fault and we can put them into $\mathcal{T}'_p$ and $\mathcal{T}_p - \mathcal{T}'_p$ respectively. In particular, for the purpose of the next step of the calculation, we can mark those transitions put into $\mathcal{T}_p - \mathcal{T}'_p$ as *backward* or *forward* to indicate whether it is involved in a potentially undetectable backward or forward output shift. This step takes $\mathcal{O}(v^2)$ time where v is the number of transitions in the given specification. At the end of this step, the set $\mathcal{T}'_p$ calculated is what we want. Then we can calculate all of the other transitions in $\mathcal{T}_p - \mathcal{T}'_p$ that have empty output at p and are involved in potentially undetectable output fault. We can keep adding transitions $s_1 \xrightarrow{x_1/y_1} s_2$ into $\mathcal{T}_p - \mathcal{T}'_p$ if the output of y_1 at p is empty and one of the following holds:

– There exists $s_2 \xrightarrow{x_2/y_2} s_3$ in $\mathcal{T}_p - \mathcal{T}'_p$ marked as *backward* and x_2 is not at p. In this case, the added transition is also marked as *backward.*
– There exists $s_3 \xrightarrow{x_2/y_2} s_1$ in $\mathcal{T}_p - \mathcal{T}'_p$ marked as *forward* and x_1 is not at p. In this case, the added transition is also marked as *forward.*

This step also takes $\mathcal{O}(v^2)$ time.

Next, we consider an algorithm:

– to check if $\mathcal{T}_p$ is verifiable at p. According to Proposition 4, this amounts to check if there exists $\mathcal{U}$ such that $\mathcal{U}$ is verifiable at p and $\mathcal{T}'_p \subseteq \mathcal{U}$ or $\mathcal{T}_p - \mathcal{T}'_p \subseteq \mathcal{U}$;
– when $\mathcal{T}_p$ is verifiable at p, construct absolute verifying paths for each transition in $\mathcal{T}_p$.

Figure 2 gives such an algorithm. Here, $\mathcal{U}$ is a set of transitions that is verifiable at p. It is initially set to empty. We search for transitions to be added into $\mathcal{U}$ and try to make $\mathcal{U} \supseteq \mathcal{T}_p$. According to Proposition 3, we only need to consider transitions in $\mathcal{T}_p$ to be added into $\mathcal{U}$, so in fact, we seek a set $\mathcal{U}$ such that $\mathcal{U} = \mathcal{T}_p$.

If we succeed, we have an absolute verifying path $\rho_{p,t}$ kept in $\mathcal{P}(t)$ for each $t \in \mathcal{U}$. Of course, if we do not need the absolute verifying paths but just want to

```
1: input: $M$ and a port $p$ of $M$
2: output: answer if $\mathcal{T}_p$ is verifiable at $p$, and if so, provide $\rho_{p,t}$ for each transition $t$
   in $\mathcal{T}_p$
3: $\mathcal{U} := \emptyset$
4: for all $t \in \mathcal{T}_p$ do
5:    $\mathcal{P}(t) := null$
6: end for
7: if $\mathcal{T}_p = \emptyset$ then
8:    $success := true$
9:    goto line 27
10: end if
11: $success := false$
12: $checkset := \mathcal{T}_p$
13: $checkset' := \emptyset$
14: while $checkset \neq \emptyset \wedge checkset' \neq checkset$ do
15:    $checkset' := checkset$
16:    if we can find an absolute verifying path $\rho_{p,t}$ upon $\mathcal{U}$ for $(t, p)$ for some $t \in$
       $checkset$ then
17:       for $t'$ contained in $pre(\rho_{p,t})$ such that $(t' \notin \mathcal{U})$ and $(t'|_p = - \Leftrightarrow t|_p = -)$ do
18:          add $t'$ to $\mathcal{U}$
19:          $\mathcal{P}(t') := \rho_{p,t}$
20:       end for
21:       $checkset := \mathcal{T}_p - \mathcal{U}$
22:       if $checkset := \emptyset$ then
23:          $success := true$
24:       end if
25:    end if
26: end while
27: if $success$ then
28:    output("success", $\mathcal{P}$)
29: else
30:    output("no such set of sequences exists.")
31: end if
```

Fig. 2. Algorithm 1: generating a set of paths

check whether $\mathcal{T}_p$ is verifiable at p, the algorithm can be easily modified so that it stops whenever $\mathcal{T}_p \subseteq \mathcal{U}$ or $\mathcal{T}'_p \subseteq \mathcal{U}$ (Proposition 4).

If $\mathcal{T}_p$ is empty, then we do not need to do anything (lines 7-10). If $\mathcal{T}_p \neq \emptyset$, then we start to check if there exists a transition $t \in \mathcal{T}_p$ that has an absolute verifying path (upon $\emptyset$) for (t, p). We use *checkset* to denote the current set of transitions that we need to search for absolute verifying paths and initially *checkset* $= \mathcal{T}_p$. Thus if *checkset* becomes $\emptyset$ then we terminate the loop and the algorithm has found a sufficient set of paths. At the end of an iteration the set *checkset*$'$ denotes the value of *checkset* before the iteration of the while loop and thus if there is no progress (*checkset*$'$ $=$ *checkset* at this point) the algorithm terminates with failure.

Whenever we find an absolute verifying path $\rho_{p,t}$ upon $\mathcal{U}$, we can add t' to $\mathcal{U}$ for all t' contained in $pre(\rho)$ and $t' \,|_p = - \Leftrightarrow t|_p = -$. This is based on Proposition 1. At the same time, we update *checkset.*

To find an absolute verifying path ρ upon $\mathcal{U}$ for (t, p), we can construct $G[t,\mathcal{U}]$ which is obtained from G by removing all edges except those corresponding to a transition t' in one of the following cases:

- t' has input at p;
- $t' \mid_p = -$ iff $t \mid_p = -$;
- $t' \in \mathcal{U}$.

We then search for a synchronizable path in $G[t,\mathcal{U}]$ that contains t, starts with input at p, and ends with input at p. We can search for such a path similar to standard algorithms (e.g. find all vertices reachable from all ending vertex of edges representing t and all vertices that get us to the starting vertex of edges representing t). Note that we do not need to consider cycles in $G[t,\mathcal{U}]$: if there exists an absolute verifying path with a cycle then there is such a path that has no cycles.

The following two results show that Algorithm 1 is correct.

Theorem 1. *Suppose that Algorithm 1 outputs "success" and $\mathcal{P}$. Then there exists a relation $\mathcal{R}$ such that $\mathcal{T}_p$ is verifiable at p under $\mathcal{R}$ and $\mathcal{P}$.*

Proof. Define a relation $\mathcal{R}$ in the following way. Given a transition $t \in \mathcal{T}_p$ consider the iteration in which t is added to $\mathcal{U}$ and let $\mathcal{U}_t$ denote the value of $\mathcal{U}$ at the beginning of this iteration. Then, since we could add t to $\mathcal{U}$ on this iteration, there is an absolute verifying path upon $\mathcal{U}_t$ for (t, p). Thus, we let $\mathcal{R}$ be the relation such that for all $t \in \mathcal{T}_p$, $\mathcal{R}(t) = \mathcal{U}_t$. Clearly $\mathcal{T}_p$ is verifiable at p under $\mathcal{R}$ and $\mathcal{P}$ as required.

Theorem 2. *Suppose that Algorithm 1 does not output "success". Then $\mathcal{T}_p$ is not verifiable at p.*

Proof. Proof by contradiction: suppose that there exists $\mathcal{R}$ and $\mathcal{P}$ such that $\mathcal{T}_p$ is verifiable at p under $\mathcal{R}$ and $\mathcal{P}$ and that Algorithm 1 terminates with a set $\mathcal{U}$ such that $\mathcal{T}_p \nsubseteq \mathcal{U}$.

Define a function *depth* from $\mathcal{T}_p$ to the integers in the following way. The base case is $depth(t) = 1$ if $\mathcal{R}(t) = \{t\}$. The recursive case is if $\mathcal{R}(t) \neq \{t\}$ then $depth(t) = 1 + max_{t' \in \mathcal{R}(t) \setminus \{t\}} depth(t')$. Let t denote an element of $\mathcal{T}_p \setminus \mathcal{U}$ that minimises $depth(t)$. But, every element of $\mathcal{R}(t)$ is in $\mathcal{U}$ and thus there exists an absolute verifying path upon $\mathcal{R}(t)$ for (p, t). This contradicts the algorithm terminating with set $\mathcal{U}$ such that $\mathcal{T}_p \nsubseteq \mathcal{U}$ as required.

Now we turn to the complexity of the algorithm.

Let $m = |\mathcal{T}_p|$ be the number of transitions involved in output shift faults at p. For each while-loop (line 14-26), we construct an absolute verifying path upon $\mathcal{U}$ for one of the transitions in *checkset*, and we can remove at least one transition from *checkset*. As initially $|checkset| = m$, the while-loop will be executed at most m times.

Within each while-loop in lines 14-26, we need to check if we can find an absolute verifying path $\rho_{p,t}$ upon $\mathcal{U}$ for (t, p) for some $t \in$ *checkset*. This can be realized by trying to construct $\rho_{p,t}$ for each $t \in$ *checkset* until such a $\rho_{p,t}$ is found. This takes at most $|$*checkset*$|$ times of effort for each attempt.

For each attempt to construct an absolute verifying path upon $\mathcal{U}$ for a given transition t, it takes $\mathcal{O}(wv)$ times to construct a path where w is the number of the states in M and v is the number of transitions in M.

For the for-loop in lines 17-20, we can keep a set α of all transitions t' contained in $pre(\rho_{p,t})$ such that $t' \notin \mathcal{U}$ and $t'|_p = - \Leftrightarrow t|_p = -$ during the construction of $\rho_{p,t}$. This does not affect our estimated time $\mathcal{O}(wv)$. After we have found such an $\rho_{p,t}$ successfully, we can move all transitions in α from *checkset* to $\mathcal{U}$. For each such move, there will be one less while-loop executed, and thus the time for the operation of the for-loop in lines 17-20 can be ignored.

In summary, the time complexity of Algorithm 1 is $\mathcal{O}(m^2wv)$.

5 Relationship with Previous Work

To make sure that each transition involved in a potentially undetectable output shift fault can be *independently* verified at port p, we need to have $\rho_1@t@\rho_2$ as an absolute verifying path upon $\emptyset$ for (t, p) for all transition t involved in a potentially undetectable output shift fault. If $\rho_1@t@\rho_2$ is an absolute verifying path upon $\emptyset$ for (t, p), then $\rho_1@t$ and $t@\rho_2$ correspond to the absolute leading path and absolute trailing path respectively defined in [4], where we have presented a necessary and sufficient condition to guarantee the existence of absolute leading path and absolute trailing path for (t, p) for each t involved in a potentially undetectable output shift fault:

Given an FSM with no same-port-output-cycles or isolated-port-cycles, for any transition t involved in a potentially undetectable 1-shift output faults, there is an absolute leading path and an absolute trailing path for (t, p) *if and only if for any pair of transitions* $s_1 \xrightarrow{*/*} s$ *and* $s \xrightarrow{*/*} t_1$ *in the FSM,*

a *if there exists a potential undetectable forward shift of an output at port p, then there exists at least one transition to s with a null output at port p, and at least one transition from s with either an input or a non-empty output at port p.*

b *if there exists a potential undetectable backward shift of an output at port p, then there exists at least one transition to s with a non-empty output at port p, and at least one transition from s with either an input or a null output at port p.*

This result is presented in terms of 1-shift output faults while it holds also for general output shift faults.

Apparently, when the above condition holds, there exists an absolute verifying path upon $\emptyset$ for (t, p) for every $t \in \mathcal{T}_p$, and thus $\mathcal{T}_p$ is verifiable. In other words, we presented in [4] a condition to guarantee that for each $t \in \mathcal{T}_p$, there exists an absolute verifying path upon $\emptyset$ for (t, p), and this condition is sufficient for $\mathcal{T}_p$ to be verifiable.

In [3], we have given a weaker condition than the one in [4]:

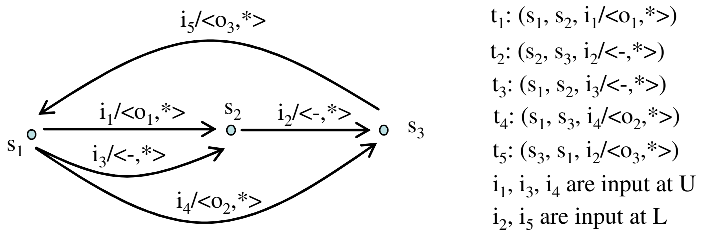


Fig. 3. Example to show the relationship with previous work

Theorem 3. *Let M be a given FSM which is not intrinsically non-synchronizable and has no same-port-output-cycles. Let p be any port of M.*

(i) *(t_0, p) has an absolute leading path for every $t_0 \in \mathcal{T}'_p$, if and only if $\forall t = s_1 \xrightarrow{x/y} s_2 \in \mathcal{T}'_p$, $x \notin I_p$ implies $\exists s_3 \xrightarrow{x'/y'} s_1 \in T$ synchonizable with t such that $y'|_p \neq -$;*
(ii) *(t_0, p) has an absolute trailing path for every $t_0 \in \mathcal{T}'_p$, if and only if $\forall t = s_1 \xrightarrow{x/y} s_2 \in \mathcal{T}'_p$, $\exists s_2 \xrightarrow{x'/y'} s_4 \in T$ synchonizable with t such that $x' \in I_p \vee y' |_p \neq -$.*

The above theorem gives a condition and declares that under this condition, it is guaranteed the existence of absolute leading path and absolute trailing path for (t, p) only for all those transitions involved in potentially undetectable output shift *and have non-empty output at p*. So it guarantees that for each transition t of this category, (t, p) has an absolute verifying path upon $\emptyset$.

Then it is proved there that for other transitions t' involved in potentially undetectable output shift but with empty output at p, there is an absolute verifying path upon $\mathcal{T}'_p$ for (t', p):

Theorem 4. *Given any FSM M that is not intrinsically non-synchronizable and port p, every $t \notin \mathcal{T}'_p$ has an absolute verifying path upon $\mathcal{T}'_p$.*

According to these two theorems, the condition in Theorem 3 is sufficient for $\mathcal{T}_p$ to be verifiable.

On the other hand, the conditions in [4,3] are not necessary for $\mathcal{T}_p$ to be verifiable.

Example 4. In Example 1 we have shown that $\mathcal{T}_p$ is verifiable at U. However, the conditions in [4,3] do not hold. This is because for (t_3, U), t_3 does not have input at U and there is no transition ending at s_2 with non-empty output at U.

The following shows another example where $\mathcal{T}_p$ is verifiable at U while the conditions in [4,3] do not hold.

Example 5. In Figure 3, there are undetectable output shift faults at port U in t_1t_2 and in t_2t_5. $\mathcal{T}_U = \{t_1, t_2, t_5\}$. $\mathcal{T}'_U = \{t_1, t_5\}$.

The conditions in [4,3] do not hold because for (t_1, U), there is no transition starting from s_2 that has either input at U or non-empty output at U.

However, $\mathcal{T}_U$ is verifiable at U:

- $t_4t_5t_1$ is an absolute verifying path upon $\emptyset$ for (t_5, U).
- $t_3t_2t_5t_1$ is an absolute verifying path upon $\{t_5\}$ for (t_2, U).
- $t_1t_2t_5t_1$ is an absolute verifying path upon $\{t_2, t_5\}$ for (t_1, U).

6 Conclusion

This paper has presented a sound procedure to check for the possibility of constructing a test/checking sequence that will not cause controllability and observability problems and will not require external coordination message exchanges among remote testers during its application in a distributed test architecture. This is realized by constructing a path that can help checking the output of a transition t at a certain port p, for each transition t involved in a potentially undetectable output shift fault. The effectiveness of this path on checking the output of transition t at port p must not be affected by controllability and observability problems. The correct output of transition t at port p is actually derived from the correct output sequence when applying the input portion of this path during the test. It remains as an interesting problem to produce an *efficient* test or checking sequence from an FSM, that is guaranteed to determine the correctness of the SUT for the considered fault model.

Acknowledgements

This work was supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada under grant RGPIN 976 and 209774, Leverhulme Trust grant number F/00275/D, Testing State Based Systems, and Engineering and Physical Sciences Research Council grant number GR/R43150, Formal Methods and Testing (FORTEST).

References

1. S. Boyd and H. Ural. The synchronization problem in protocol testing and its complexity. Information Processing Letters, 40:131136, 1991.
2. L. Cacciari and O. Rafiq. Controllability and observability in distributed testing. Information and Software Technology, 41:767780, 1999.
3. J. Chen, R. M. Hierons, and H. Ural. Overcoming observability problems in distributed test architectures. Submitted for publication.
4. J. Chen, R. M. Hierons, and H. Ural. Conditions for resolving observability problems in distributed testing. In 24rd IFIP International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2004), volume 3235 of LNCS, pages 229242. Springer-Verlag, 2004.
5. W. Chen and H. Ural. Synchronizable checking sequences based on multiple UIO sequences. IEEE/ACM Transactions on Networking, 3:152157, 1995.

6. A. Gill. Introduction to the Theory of Finite-State Machines. New York: McGraw-Hill, 1962.
7. S. Guyot and H. Ural. Synchronizable checking sequences based on UIO sequences. In Proc. of IFIP IWPTS95, pages 395407, Evry, France, September 1995.
8. F.C. Hennie. Fault detecting experiments for sequential circuits. In Proc. of Fifth Ann. Symp. Switching Circuit Theory and Logical Design, pages 95110, Princeton, N.J., 1964.
9. R. M. Hierons. Testing a distributed system: generating minimal synchronised test sequences that detect output-shifting faults. Information and Software Technology, 43(9):551560, 2001.
10. D. Lee and M. Yannakakis. Principles and methods of testing finitestate machines a survey. Proceedings of the IEEE, 84(8):10891123, 1996.
11. G. Luo, R. Dssouli, and G. v. Bochmann. Generating synchronizable test sequences based on finite state machine with distributed ports. In The 6th IFIP Workshop on Protocol Test Systems, pages 139153. Elsevier (North-Holland), 1993.
12. G. Luo, R. Dssouli, G. v. Bochmann, P. Venkataram, and A. Ghedamsi. Test generation with respect to distributed interfaces. Computer Standards and Interfaces, 16:119132, 1994.
13. K.K. Sabnani and A.T. Dahbura. A protocol test generation procedure. Computer Networks, 15:285297, 1988.
14. B. Sarikaya and G. v. Bochmann. Synchronization and specification issues in protocol testing. IEEE Transactions on Communications, 32:389395, April 1984.
15. K.C. Tai and Y.C. Young. Synchronizable test sequences of finite state machines. Computer Networks, 13:11111134, 1998.
16. H. Ural and Z. Wang. Synchronizable test sequence generation using UIO sequences. Computer Communications, 16:653661, 1993.
17. Y.C. Young and K.C. Tai. Observation inaccuracy in conformance testing with multiple testers. In Proc. of IEEE WASET, pages 8085, 1998.