Detecting Observability Problems in Distributed Testing

Jessica Chen[1] and Hasan Ural[2]

[1] School of Computer Science, University of Windsor
Windsor, Ontario, Canada N9B 3P4
xjchen@uwindsor.ca
[2] School of Information Technology and Engineering, University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
ural@site.ottawa.ca

Abstract. Application of a test or checking sequence in a distributed test architecture often requires the use of external coordination message exchanges among multiple remote testers for eluding potential controllability and observability problems. Recent literature reports on conditions on a given finite state machine (FSM) under which controllability and observability problems can be overcome without using external coordination messages. However, these conditions do not guarantee that any test/checking sequence constructed from such FSMs are free from controllability and observability problems. For a given test or checking sequence, this paper investigates whether it is possible to eliminate the need for external coordination messages and proposes algorithms to identify or construct subsequences either within the given sequence or as an extension to the given sequence, respectively.

Keywords: finite state machine, testing, distributed test architecture, observability, controllability.

1 Introduction

In a *distributed test architecture*, there is one tester at each interface/*port* of the system under test (SUT) N. These testers participate in applying a given test sequence [1, 15, 16] or checking sequence [7, 9, 11, 19] which is a sequence of input/output pairs, constructed from the specification M of the SUT N. The use of multiple remote testers in a distributed architecture brings out the possibility of controllability and observability problems during the application of a test or checking sequence. A controllability problem arises when a tester is required to send the current input and because it did not send the previous input and did not receive the previous output it cannot determine when to send the input. An observability problem arises when a tester is expecting an output in response to either a 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.

These problems and their solutions have been studied in the context where M is a Finite State Machine (FSM) and N is a state-based system whose externally observable behavior can also be represented by an FSM. Much of the previous work has been focused on automatically generating test or checking sequences from FSMs that causes no controllability or observability problems during its application in a distributed test architecture (see, for example, [2, 6, 8, 10, 13, 17, 18, 20]). For some FSMs, there have been test/checking sequences in which the coordination among testers can be achieved indirectly via their interactions with N [14, 16]. For some others, it may be necessary for testers to communicate directly by exchanging external coordination messages among themselves over a dedicated channel for overcoming the controllability and observability problems encountered during the application of the test/checking sequence [2, 3, 17]. Using external coordination messages introduces delays and the necessity to set up a dedicated communications channel among testers. Thus, the emphasis of the recent work is to minimize the use of external coordination message exchanges among testers [3, 10] or to identify conditions on a given FSM M under which controllability and observability problems can be overcome without using external coordination messages [4, 5].

Such conditions lead to the algorithms for identifying paths within a given FSM M that provide evidence for the possibility of eliminating the controllability and observability problems [4, 5]. [4] gives conditions on M so that each transition involved in an observability problem can be independently verified at port p. By *verified at port p*, it is meant that one can conclude that the output of this transition at port p is correct if one observes the correct output sequence on a certain path within M. By *independently*, it is meant that the above conclusion regarding the output at port p for a transition does not rely on the correctness of any other transitions. Since the notion of independence may not be required in some cases, the above condition on M can be weakened in these cases. [5] gives an algorithm that determines whether M satisfies this weaker condition and when it does so, identifies paths within M that check the output of the transitions.

In this paper, we assume that the given FSM M satisfies the condition in [5]. Then, we pose the following problem and solve it in a restricted setting: Given an FSM M and a synchronizable test or checking sequence τ_0 starting at the initial state of M, extend τ_0 with minimal number of subsequences to form a synchronizable test or checking sequence τ^* such that the detectability of the observability problems in τ_0 is guaranteed without using external coordination messages exchanged among remote testers.

The rest of the paper is organized as follows. Section 2 introduces the preliminary terminology. Section 3 gives a formal definition of the general problem and defines a restricted version of this problem. Section 4 presents our solution. Section 5 concludes the paper with our final remarks.

2 An n-port FSM and Directed Graphs

An *n-port Finite State Machine* M (called henceforth an FSM M) is defined as $M = (S, I, O, \delta, \lambda, s_0)$ where S is a finite set of states; $s_0 \in S$ is the initial state; $I = \bigcup_{i=1}^{n} I_i$, where I_i is the set of input symbols 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 set of output symbols 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., $y = \langle o_1, o_2, ..., o_n \rangle$ where $o_i \in O_i \cup \{-\}$ for $i \in [1, n]$. 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. $p \in [1, n]$ 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 $\mathcal{T}$ 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 $i \in [1, k-1]$. We say t is *contained in* (or simply *in*) ρ if t is a transition along path ρ. 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*.

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.

Given an FSM M and a path $t_1 t_2 \ldots t_k$ $(k > 1)$ of M with label $x_1/y_1\ x_2/y_2\ \ldots\ x_k/y_k$, 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 *synchronizable pair* of transitions if t_{i+1} can follow t_i without causing a synchronization problem. Any path in which every pair of consecutive 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 holds, then the FSM is called *intrinsically synchronizable*.

Suppose that we are given an FSM M and a synchronizable path $\rho = t_1 t_2 \ldots t_k$ of M with label $x_1/y_1 x_2/y_2 \ldots x_k/y_k$. An *output shift fault* in an implementation N of M exists if one of the following holds for some $1 \leq i < j \leq k$:

a) For some $p \in [1, n]$ and $o \in O_p$, $y_i\mid_p = o$ in M and for all $i < l \leq j$, $y_l\mid_p = -$ in M whereas for all $i \leq l < j$, N produces output $-$ at p in response to

x_l after $x_1 \ldots x_{l-1}$, and N produces output o at p in response to x_j after $x_1 \ldots x_{j-1}$.

b) For some $p \in [1, n]$ and $o \in O_p$, $y_j \mid_p = o$ in M and for all $i \leq l < j$, $y_l \mid_p = -$ in M whereas for all $i < l \leq j$, N produces output $-$ at p in response to x_l after $x_1 \ldots x_{l-1}$, and N produces output o at p in response to x_i after $x_1 \ldots x_{i-1}$.

In a) the output o shifts from being produced in response to x_i to being produced in response to x_j and the shift is from t_i to t_j (i.e., a *forward* shift). In b) the output o shifts from being produced in response to x_j to being produced in response to x_i and the shift is from t_j to t_i (i.e., a *backward* shift).

An instance of the observability problem manifests itself as a *potentially undetectable output shift fault* if there is an output shift fault related to $o \in O_p$ in two transitions t_i and t_j in ρ with labels x_i/y_i and x_j/y_j, such that $x_{i+1} \ldots x_j \notin I_p$. The tester at p will not be able to detect the faults since it will observe the expected sequence of interactions in response to $x_i \ldots x_j$. Both t_i and t_j are said to be *involved* in the potentially undetectable output shift fault. When $j = i + 1$, we also call it potentially undetectable *1-shift* output fault.

In the following, τ_0 is a given test/checking sequence, which is the label of path $\rho_0 = t_1 t_2 \ldots t_m$. We will use $\mathcal{T}_{\rho_0,p}$ to denote the set of transitions of M that can be involved in potentially undetectable output shift faults in ρ_0. Thus $t \in \mathcal{T}_{\rho_0,p}$ if there exists a transition t' and a synchronizable path $t\rho t'$ or $t'\rho t$ such that both t and t' are involved in a potentially undetectable output shift fault when we apply τ_0 to N.

Let t be a transition, and $\mathcal{U}$ a set of transitions in M. ρ is *an absolute verifying path upon* $\mathcal{U}$ *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;
- $t \notin \mathcal{U}$ and for all t' contained in pre(ρ), either $t' \in \mathcal{U}$ or $t' \mid_p = - \Leftrightarrow t \mid_p = -$ [5].

Note that given t and ρ we will typically consider a *minimal* set $\mathcal{U}$ that satisfies the above conditions: if $t' \mid_p = - \Leftrightarrow t \mid_p = -$ then $t' \notin \mathcal{U}$.

Suppose that $\mathcal{U}$ is a set of transitions of M, $\mathcal{R} \subseteq \mathcal{U} \times \mathcal{U}$ is a relation, 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 [5].

(a) For all $t \in \mathcal{U}$, $\mathcal{P}(t)$ is an absolute verifying path upon $\{t' \mid (t, t') \in \mathcal{R}\}$ 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.

Let $\mathcal{T}_p$ be the set of all transitions involved in some potentially undetectable output shift faults in M at port p. In this paper, we assume that $\mathcal{T}_p$ is verifiable at p for all $p \in [1, n]$.

A *directed graph (digraph)* G is defined by a tuple (V, E) in which V is a set of vertices and E is a set of directed edges between the vertices. An edge e from vertex v_i to vertex v_j is represented by (v_i, v_j). A *walk* is a sequence of pairwise adjacent edges in G. A digraph is *strongly connected* if for any ordered pair of vertices (v_i, v_j) there is a walk from v_i to v_j.

3 The problem definition

Given a deterministic, minimal, and completely specified FSM M which is intrinsically synchronizable, and a synchronizable test/checking sequence τ_0 starting at the initial state of M, we consider the problem of constructing a synchronizable test/checking sequence τ^* that can be applied to resolve observability problems in τ_0 without using external coordination message exchanges by identifying the subsequences within τ_0 or to be appended to τ_0.

Clearly, for each $t \in \mathcal{T}_{\rho_0,p}$, we should verify its output at port p. As we discussed in [5], to verify the output of transition t at port p, we can construct an *absolute verifying path upon a set* $\mathcal{U}$ of transitions whose outputs at p are verified. Such a path ρ has the following properties:

- it is synchronizable;
- we are able to determine the output sequence of ρ at p by applying the label 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.

This is because (i) 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 in the label of ρ since this output sequence is immediately preceded and followed by input at p; (ii) the condition *for all* t' *contained in* $pre(\rho)$*, either* $t' \in \mathcal{U}$ *or* $t' \mid_p = - \Leftrightarrow t \mid_p = -$ allows us to determine the correct output of (t, p) from the correct output sequence of ρ at p (Proposition 2 in [5]).

Thus, to verify the outputs of the transitions in $\mathcal{T}_{\rho_0,p}$ at port p, we search for an acyclic digraph of transitions such that all transitions in $\mathcal{T}_{\rho_0,p}$ are present, and each transition has an absolute verifying path upon a set of transitions that appear as its successors in the digraph. In other words, we search for $\mathcal{R}$ and $\mathcal{P}$ such that set $\mathcal{T}_{\rho_0,p}$ of transitions is verifiable at p under $\mathcal{R}$ and $\mathcal{P}$.

It is possible that ρ_0 contains some absolute verifying paths for transitions in $\mathcal{T}_{\rho_0,p}$. Let Q_p be the set of all those paths in $codomain(\mathcal{P})$ but not as subsequences in ρ_0. τ^* will be the label of a path ρ^* which contains both ρ_0 and all paths in Q_p.

Clearly, for efficiency reasons,

- We should maximize the images of $\mathcal{P}$ in ρ_0. That is, whenever possible, we should define $\mathcal{P}(t)$ as a subsequence in ρ_0 for any $t \in \mathcal{T}$.
- No path in Q_p should appear as a subsequence of another path in Q_p. This is always true as the absolute verifying paths have input at port p only in its first and last transitions.

– There is no redundant path in Q_p. An absolute verifying path ρ is redundant in Q_p if we can modify $\mathcal{P}$ (and $\mathcal{R}$ correspondingly) by changing the mapping of all transitions whose image is ρ under $\mathcal{P}$ to some other paths in Q_p while keeping the property that $\mathcal{T}_{\rho_0,p}$ is verifiable at p under the modified definitions of $\mathcal{P}$ and $\mathcal{R}$. Figure 1(a) shows a case where $\{t_1, t_2, t_3\}$ is verifiable at p under $\mathcal{P}$ and $\mathcal{R}$ where $\mathcal{P}(t_i) = \rho_i$ for $i = 1, 2, 3$. Suppose that ρ_2 is also an absolute verifying path upon $\{t_3\}$ for (t_1, p), then Figure 1(b) shows an alternative way to verify $\{t_1, t_2, t_3\}$ which requires less paths in Q_p to be considered in constructing τ^*: $\mathcal{P}(t_1) = \mathcal{P}(t_2) = \rho_2$, $\mathcal{P}(t_3) = \rho_3$.

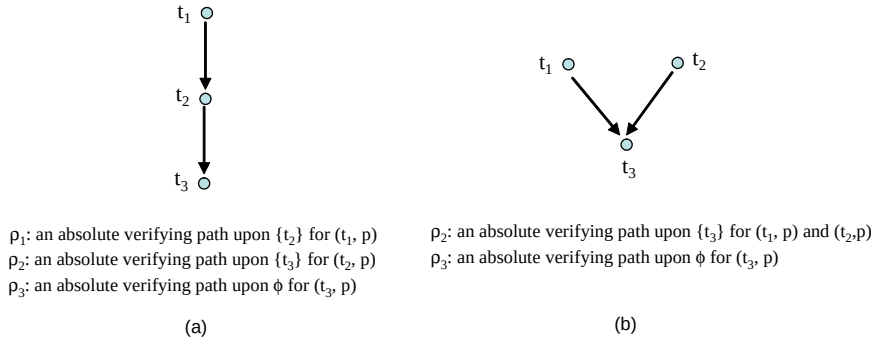

Fig. 1. An example of reducing paths in Q_p

4 Our proposed solution

Now we present our solution to construct Q_p and τ^*.

4.1 Identifying transitions involved in observability problems

Recall that $\tau_0 = x_1/y_1\ x_2/y_2\ \ldots\ x_m/y_m$ is a test/checking sequence of M which is the label of a path $\rho_0 = t_1 t_2 \ldots t_m$. First we need to calculate $\mathcal{T}_{\rho_0,p}$, the set of transitions involved in potentially undetectable output shift faults at port p in ρ_0, for all $p \in [1, n]$. Figure 2 shows an algorithm for this purpose. It scans τ_0 and uses *emptyPointer* and *nonEmptyPointer* as auxiliary variables. We do not consider the case when $|\tau_0| = 0$ which is meaningless. Suppose we are currently considering $x_i/y_i \in \tau_0$.

emptyPointer is the minimal index of the transitions in τ_0 such that

– $\forall k \in [emptyPointer + 1, i - 1].\ x_k \notin I_p$ and
– $\forall k \in [emptyPointer, i - 1].\ y_k\ |_p = -$

nonEmptyPointer is the index of the transitions in τ_0 such that

- $y_k \mid_p \neq -$ for $k = nonEmptyPointer$ and
- $\forall k \in [nonEmptyPointer + 1, i - 1]$. $x_k \notin I_p \wedge y_k \mid_p = -$

If neither *emptyPointer* nor *nonEmptyPointer* is *null*, then for all $k \in [nonEmptyPointer, i-1]$, t_k is involved in a potentially undetectable forward output shift fault. Furthermore, in the case $x_i \notin I_p$ and $y_i \mid_p = -$, t_i is also involved in a potentially undetectable forward output shift fault.

If *emptyPointer* is not *null*, no matter whether *nonEmptyPointer* is *null* or not, t_k is involved in a potentially undetectable backward output shift fault for all $k \in [emptyPointer, i]$ when $x_i \notin I_p$ and $y_i \mid_p \neq -$.

Note that some transitions at the end of ρ_0 that are not involved in any potentially undetectable output shift fault in ρ_0 may be involved in such faults in the constructed ρ^*. All these transitions are also added into $\mathcal{T}_{\rho_0,p}$ in lines 36-52 which specifically handle the case when $i = m$.

The execution of Algorithm 1 can be done in $\mathcal{O}(|\tau_0|)$ time.

4.2 Identifying verifiable transitions

By definition, the transitions in $\mathcal{T} - \mathcal{T}_{\rho_0,p}$ all have correct output at p. On the other hand, not all transitions in $\mathcal{T}_{\rho_0,p}$ need to be verified for its output at p with additional subsequences. This is based on the following two observations:

- A transition in $\mathcal{T}_{\rho_0,p}$ may appear in a different place in ρ_0 where it is not involved in any potentially undetectable output shift faults at p in ρ_0, and thus its output at p is verified in ρ_0.
- Given a transition $t \in \mathcal{T}_{\rho_0,p}$, there may exist an absolute verifying path upon $\mathcal{T} - \mathcal{T}_{\rho_0,p}$ for (t, p) in ρ_0.

In general, before constructing additional subsequences to be appended to τ_0, we would like to find $\mathcal{R}_0$, $\mathcal{P}_0$ and $\mathcal{U}_0 \subset \mathcal{T}_{\rho_0,p}$ such that

- $\mathcal{U}_0$ is verifiable at p under $\mathcal{R}_0$ and $\mathcal{P}_0$ *in* ρ_0, in the sense that $\mathcal{U}_0$ is verifiable at p under $\mathcal{R}_0$ and $\mathcal{P}_0$, and the paths in *codomain*$(\mathcal{P}_0)$ are all in ρ_0;
- $\mathcal{U}_0$ is maximized, in the sense that for any $\mathcal{R}'_0$, $\mathcal{P}'_0$ and $\mathcal{U}'_0$ such that $\mathcal{U}'_0$ is verifiable at p under $\mathcal{R}'_0$ and $\mathcal{P}'_0$ in ρ_0, $\mathcal{U}'_0 \subseteq \mathcal{U}_0$.

The following proposition follows directly from the definition.

Proposition 1. *Let ρ be a synchronizable path with input at p only in* first(ρ) *and* last(ρ)*, and $t \in$* pre(ρ)*. Let $D_{t,\rho}$ be the set of transitions in* pre(ρ) *such that for any $t' \in D_{t,\rho}$, $t' \mid_p = - \Leftrightarrow t \mid_p \neq -$. Then ρ is an absolute verifying path upon $D_{t,\rho}$ for (t, p).*

Let ρ be a subsequence in ρ_0 with input at p both at the beginning and at the end. Based on the above proposition, if the set of all those transitions in ρ with empty output at p is verifiable, then the set of all transitions in ρ

1: **input**: an FSM M, a port p, a test/checking sequence $\tau_0 = x_1/y_1\ x_2/y_2\ \ldots x_m/y_m$ of M
2: **output**: $\mathcal{T}_{\rho_0,p}$
3: $nonEmptyPointer := null$
4: $emptyPointer := null$
5: $i := 1$
6: **while** $i < m$ **do**
7: **if** $x_i \notin I_p$ **then**
8: **if** $y_i\ |_p \neq -$ **then**
9: **if** $emptyPointer \neq null \wedge nonEmptyPointer \neq null$ **then**
10: add $t_{nonEmptyPointer}, \ldots, t_i$ to $\mathcal{T}_{\rho_0,p}$
11: **end if**
12: **if** $emptyPointer \neq null \wedge nonEmptyPointer = null$ **then**
13: add $t_{emptyPointer}, \ldots, t_i$ to $\mathcal{T}_{\rho_0,p}$
14: **end if**
15: $nonEmptyPointer := i$
16: $emptyPointer := null$
17: **else**
18: **if** $nonEmptyPointer = i - 1$ **then**
19: $emptyPointer = i$
20: **end if**
21: **end if**
22: **else**
23: **if** $emptyPointer \neq null \wedge nonEmptyPointer \neq null$ **then**
24: add $t_{nonEmptyPointer}, \ldots, t_{i-1}$ to $\mathcal{T}_{\rho_0,p}$
25: **end if**
26: **if** $y_i\ |_p \neq -$ **then**
27: $nonEmptyPointer := i$
28: $emptyPointer := null$
29: **else**
30: $nonEmptyPointer := null$
31: $emptyPointer := i$
32: **end if**
33: **end if**
34: $i := i + 1$
35: **end while**
36: **if** $emptyPointer \neq null \wedge nonEmptyPointer \neq null$ **then**
37: add $t_{nonEmptyPointer}, \ldots, t_m$ to $\mathcal{T}_{\rho_0,p}$
38: **end if**
39: **if** $emptyPointer \neq null \wedge nonEmptyPointer = null$ **then**
40: **if** $x_m \notin I_p$ **then**
41: add $t_{emptyPointer}, \ldots, t_m$ to $\mathcal{T}_{\rho_0,p}$
42: **else**
43: add t_m to $\mathcal{T}_{\rho_0,p}$
44: **end if**
45: **end if**
46: **if** $emptyPointer = null \wedge nonEmptyPointer \neq null$ **then**
47: **if** $x_m \notin I_p$ and $y_m\ |_p = -$ **then**
48: add t_{m-1}, t_m to $\mathcal{T}_{\rho_0,p}$
49: **else**
50: add t_m to $\mathcal{T}_{\rho_0,p}$
51: **end if**
52: **end if**

Fig. 2. Algorithm 1: Construction of $\mathcal{T}_{\rho_0,p}$

1: **input**: an FSM M, a port p, a test/checking sequence $\tau_0 = x_1/y_1x_2/y_2 \dots x_m/y_m$ of M, and $\mathcal{T}_{\rho_0,p}$
2: **output**: a set $\mathcal{U}_0$ of transitions that is verifiable at p in ρ_0, and a set Θ of counter-pairs of p
3: $\Theta := \emptyset$
4: Let $r \leq m$, s.t. $x_r \in I_p$ and $\forall k$, $1 \leq k < r$, $x_k \notin I_p$
5: **while** $\exists j.\ r < j \leq m$ s.t. $x_j \in I_p$ and $\forall k,\ r < k < j,\ x_k \notin I_p$ **do**
6: let j be such that $r < j \leq m$, $x_j \in I_p$ and $\forall k,\ r < k < j,\ x_k \notin I_p$
7: **if** $\exists r \leq k < j$ s.t. $t_k \in \mathcal{T}_{\rho_0,p}$ **then**
8: $L_1 := \emptyset$
9: $L_2 := \emptyset$
10: **for** $k,\ r \leq k < j$ **do**
11: **if** $y_k \mid_p = -$ **then**
12: add t_k to L_1
13: **else**
14: add t_k to L_2
15: **end if**
16: **end for**
17: add (L_1,L_2) to Θ
18: **end if**
19: $r = j$
20: **end while**
21: $(\mathcal{U}', \Theta') := counterPairsUpdate(\mathcal{T} - \mathcal{T}_{\rho_0,p}, \Theta)$
22: return $\mathcal{U}'$ and Θ'

Fig. 3. Algorithm 2: Construction of U_0 and Θ

```
1: input: 𝒰 and Θ
2: output: updated 𝒰 and Θ
3: change := true
4: while change = true do
5:    for each (L1, L2) ∈ Θ do
6:       L1 := L1 − 𝒰
7:       L2 := L2 − 𝒰
8:    end for
9:    change := false
10:   for  each (L1, L2) ∈ Θ do
11:      if L1 = ∅ then
12:         add all transitions in L2 to 𝒰
13:         remove (L1, L2) from Θ
14:         change := true
15:      end if
16:      if L2 = ∅ then
17:         add all transitions in L1 to 𝒰
18:         remove (L1, L2) from Θ
19:         change := true
20:      end if
21:   end for
22: end while
```

Fig. 4. procedure of *counterPairsUpdate*

is verifiable using ρ as an absolute verifying path. Analogously, if the set of all those transitions in ρ with non-empty output at p is verifiable, then the set of all transitions in ρ is verifiable.

Thus, we can derive from ρ_0 a set of so-called *counter-pairs* (L_1, L_2) of sets of transitions. Each counter-pair (L_1, L_2) corresponds to a *potential candidate* of absolute verifying path in ρ_0 that can be used in defining $\mathcal{P}$. It is obtained in this way: for any subsequence ρ of ρ_0 with input at p both at the beginning and at the end (and no other input at p in it), there is a counter-pair (L_1, L_2) where L_1 contains all transitions in $pre(\rho)$ with empty output at p, and L_2 contains all transitions in $pre(\rho)$ with non-empty output at p. Such counter-pairs hold the following property: for any set A of transitions in $\mathcal{T}$, the outputs of all transitions in L_1 are verifiable upon A implies the outputs of all transitions in L_2 are verifiable upon $A \cup L_1$; and the outputs of all transitions in L_2 are verifiable upon A implies the outputs of all transitions in L_1 are verifiable upon $A \cup L_2$. Consequently, for any $t \in L_1$, the path corresponding to (L_1, L_2) can be used as an absolute verifying path upon $\mathcal{U}$ for (t, p) if $L_2 \subseteq \mathcal{U}$. Conversely, for any $t \in L_2$, the path corresponding to (L_1, L_2) can be used as an absolute verifying path upon $\mathcal{U}$ for (t, p) if $L_1 \subseteq \mathcal{U}$.

Figure 3 gives an algorithm to calculate set $\mathcal{U}_0$ of transitions whose outputs at p are verifiable in ρ_0. Set Θ contains those counter-pairs that correspond to potential candidates of absolute verifying paths. Given a set $\mathcal{U}_0$ of transitions that is verifiable at p under $\mathcal{R}_0$ and $\mathcal{P}_0$ in ρ_0, we can check if any potential

candidate of absolute verifying path can be used to extend $\mathcal{U}_0$. This operation is performed in Figure 4. Counter-pairs whose corresponding paths will no more be used during the construction of $\mathcal{R}_0$ and $\mathcal{P}_0$ are removed from Θ.

Note that if there is no input in τ_0 that will be given at port p, then we are not able to construct an absolute verifying path for any output at p. Since we assume that $\mathcal{T}_p$ is verifiable, this implies that $\mathcal{T}_{\rho_0,p} = \emptyset$, and thus there is no need for the subsequences to be appended to ρ_0 for port p. Hence we consider there is at least one input at p in τ_0.

At the end of Algorithm 2, we have that (i) $\mathcal{U}_0$ is verifiable at p under $\mathcal{R}_0$ and $\mathcal{P}_0$ in ρ_0, and it is maximized; (ii) all potential absolute verifying paths in ρ_0 for further use have their correspondence in Θ.

We know that $\Sigma_{(L_1,L_2)\in\Theta}(|L_1| + |L_2|) \leq |\tau_0|$, and $|\mathcal{U}| \leq |\mathcal{T}|$. So in Figure 4, the first for-loop will be executed maximally $|\tau_0| \times |\mathcal{T}|$ times, and the second for-loop will be executed maximally $|\tau_0|$ times. The while-loop each time removes at least one counter-pair from Θ. So in total it takes $\mathcal{O}(|\tau_0| \times |\mathcal{T}| \times |\Theta|)$ time to perform *counterPairsUpdate*. Consequently, it takes $\mathcal{O}(|\tau_0| \times |\mathcal{T}| \times |\Theta|)$ time to run Algorithm 2.

4.3 Identifying subsequences to be added to τ_0

Given an initial set $\mathcal{U}_0$ of transitions that is verifiable at p in ρ_0, and a set Θ of counter-pairs corresponding to some potential absolute verifying paths, we define $\mathcal{P}$ and $\mathcal{R}$ such that $\mathcal{T}_{\rho_0,p}$ is verifiable at p under $\mathcal{R}$ and $\mathcal{P}$; the images of $\mathcal{P}$ in ρ_0 is maximized; there is no redundant path in $\mathcal{U}$. This leads to the construction of Q_p that we want.

Figure 5 gives an algorithm to construct Q_p. Here *checkset* is used to keep the transitions that we may need to construct additional subsequences to verify their output at p. Since we assume that $\mathcal{T}_p$ is verifiable at port p, $\mathcal{T}_{\rho_0,p} - \mathcal{U}$ is also verifiable. So for each iteration of the outer while-loop, we can surely find an absolute verifying path upon $\mathcal{U}$ for some $t \in$ *checkset* before *checkset* becomes empty.

Whenever we find an absolute verifying path upon $\mathcal{U}$ for some $t \in$ *checkset*, we add to $\mathcal{U}$ all transitions in $pre(\rho)$ such that they have empty output at p if and only if t has empty output at p. This is because if ρ is an absolute verifying path upon $\mathcal{U}$ for (t, p), then ρ is an absolute verifying path upon $\mathcal{U}$ for (t', p) for all $t' \in pre(\rho)$ such that $t' \mid_p = - \Leftrightarrow t \mid_p = -$ (Proposition 1 in [5]). This also guarantees that when we search for an absolute verifying path upon $\mathcal{U}$ for (t, p), we do not need to check whether previously constructed subsequences in Q_p can be re-used. Consequently, there is no redundant path in $\mathcal{U}$.

Whenever an additional sequence is constructed and added to Q_p, $\mathcal{U}$ is updated. Correspondingly, we call procedure *counterPairsUpdate* to check if based on the updated $\mathcal{U}$ any potential absolute verifying path in ρ_0 can be used. As the initial value of $\mathcal{U}$ is from Algorithm 2, this guarantees that for any $\rho \in Q_p$, ρ is not a subsequence of ρ_0. Thus, the images of $\mathcal{P}$ in ρ_0 is maximized.

¿From [5], we know that if ρ is an absolute verifying path upon $\mathcal{U}$ for (t, p), then when we apply the label of ρ from a state in N similar to the starting state

```
1: input: p, T_{ρ0,p}, Θ and U_0
2: output: Q_p
3: U = U_0
4: while T_{ρ0,p} − U ≠ ∅ do
5:    checkset := T_{ρ0,p} − U
6:    found := false
7:    while found = false do
8:       let t ∈ checkset
9:       if there exists an absolute verifying path upon U for (t, p) then
10:         let ρ be a minimal-length absolute verifying path upon U for (t, p)
11:         add ρ to Q_p
12:         for each transition t' ∈ pre(ρ) s.t. t' |_p= − ⇔ t |_p= −, add t' to U
13:         (U, Θ) := counterPairsUpdate(U, Θ)
14:         found := true
15:      else
16:         checkset := checkset − {t}
17:      end if
18:   end while
19: end while
```

Fig. 5. Algorithm 3: Construction of Q_p

of ρ, then we can verify that the output of t at p is correct. So, when we have $\mathcal{T}_{\rho_0,p} - \mathcal{U} = \emptyset$ at the end of the algorithm, we know that if we apply τ_0 from the initial state of N and apply the label of ρ from a state similar to the starting state of ρ for all $\rho \in Q_p$, then we can verify that there is no undetectable output shift faults occurred in applying τ_0 to N.

To find a minimal-length absolute verifying path upon $\mathcal{U}$ for (t, p), similar as in [5], 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;
- $t' \mid_p = -$ if and only if $t \mid_p = -$;
- $t' \in \mathcal{U}$

We then use breadth-first search to construct minimal-length synchronizable path in $G[t,\mathcal{U}]$ that starts with input at p and ends with input at p. Note that there may exist more than one such path with minimal-length.

Note also that while more transitions are added to $\mathcal{U}$, there may exist shorter path for a transition whose image under $\mathcal{P}$ was previously added to Q_p.

Now we turn to the complexity of the algorithm. For each outer while-loop, $\mathcal{U}$ is augmented by at least one transition. So the outer while-loop will be executed at most v times where v is the number of transitions to be verified. For the inner while-loop, we need to check if we can find an absolute verifying path upon $\mathcal{U}$ for some $t \in$ *checkset* where $|checkset| \leq v$. This can be realized by trying to construct an absolute verifying path upon $\mathcal{U}$ for each t in *checkset* until such a path 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 $\mathcal{O}(w \times |\mathcal{T}|)$ times where w is the number of states in M. In summary, the time complexity of Algorithm 3 is $\mathcal{O}(v^2 \times w \times |\mathcal{T}|)$.

4.4 Adding subsequences to τ_0

Finally, given ρ_0 and Q_p for each p, we need to construct a minimal-length test/checking sequence τ^* so that (i) it is synchronizable; (ii) it starts with τ_0 and it contains all the input/output sequences of the paths in Q_p for each $p \in [1, n]$. Figure 6 gives such an algorithm. It generates a synchronizable path ρ^* and its label τ^*.

1: **input**: M, τ_0, and Q_p for each $p \in [1, n]$
2: **output**: test/checking sequence τ^*
3: Let $Q = \cup_{p \in [1,n]} Q_p \cup \{\rho_0\}$
4: Let graph G contain one vertex v_ρ for each path ρ in Q
5: **for** each ordered pair $(\rho_1, \rho_2) \in Q$ such that $\rho_1 \neq \rho_2$ **do**
6: find a shortest path ρ' in M such that $last(\rho_1)\ \rho'\ first(\rho_2)$ is a synchronizable path.
7: In G, add an edge $e = (v_{\rho_1}, v_{\rho_2})$, with $|\rho'|$ as its weight
8: let $f_1(e) = \rho_1$, $f_2(e) = \rho_1\rho'$, $f_3(e) = \rho_1\rho'\rho_2$
9: **end for**
10: Find a walk $r = e_1 e_2 \dots e_k$ in G that visits all vertices at least once with minimal cost, and that $f_1(e_1) = \rho_0$
11: Let $\rho^* = f_2(e_1) f_2(e_2) \dots f_2(e_{k-1}) f_3(e_k)$
12: Let τ^* be the label of ρ^*

Fig. 6. Algorithm 4: Addition of elements of Q_p to ρ_0 to form ρ^*

As we assume that M is intrinsically synchronizable, G is a strongly-connected digraph. This guarantees the existence of r. In general, the time complexity of Algorithm 4 is equivalent to that of finding a travelling saleman tour in a digraph. Efficient heuristics exist for the solution of Travelling Saleman Problem, cf. [12].

Note that ρ^* may introduce new observability problems. However, since each path in Q_p has input at p in its first and last transitions, a new observability problem cannot happen between a transition in a *connecting path*, i.e. a path used to connect paths in Q_p, and a transition in an absolute verifying path in Q_p. It can only happen (i) within a connecting path; (ii) within an absolute verifying path; or (iii) between a transition in ρ_0 and a transition in a connecting path. The new observability problems occurred in cases (i) and (ii) do not affect the ability of τ^* to verify that there is no undetectable output shift faults when τ_0 is applied to N. The new observability problems in case (iii) are resolved because we have included into $\mathcal{T}_{\rho_0,p}$ all transitions that may possibly get involved in some potentially undetectable output shift fault between a transition in ρ_0 and a transition in a path concatenated to the end of ρ_0 (cf. Algorithm 1).

5 Conclusions and Final Remarks

We have presented a method for eliminating the use of external coordination message exchanges for resolving observability problems in a given test/checking sequence constructed from an FSM satisfying conditions given in [5]. There are various optimization problems remaining to be solved. First, the existence of multiple minimal-length absolute verifying paths can be used to optimize the total length of ρ^*. Second, in our solution, the order of generating the subsequences will have an effect on the final set of additional subsequences. It will be interesting to find approaches for eliminating this effect. Third, our solution only considers the subproblem of constructing the subsequences for each port p individually. It remains as an interesting problem to consider the global optimization problem among all ports. Fourth, it will be quite interesting to incorporate some of the algorithms proposed here into a checking sequence construction method to construct a checking sequence in which there are no external coordination message exchanges. It is anticipated that the complexity of the last two optimization problems will be very high.

Acknowledgements

This work is supported by Natural Sciences and Engineering Research Council (NSERC) of Canada under grant RGPIN 976 and 209774.

References

1. A. V. Aho, A. T. Dahbura, D. Lee, and M. U. Uyar. An optimization technique for protocol conformance test generation based on UIO sequences and Rural Chinese Postman Tours. In *Protocol Specification, Testing, and Verification VIII*, pages 75–86, Atlantic City, 1988. Elsevier (North-Holland).
2. S. Boyd and H. Ural. The synchronization problem in protocol testing and its complexity. *Information Processing Letters*, 40:131–136, 1991.
3. L. Cacciari and O. Rafiq. Controllability and observability in distributed testing. *Information and Software Technology*, 41:767–780, 1999.
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 229–242. Springer-Verlag, 2004.
5. J. Chen, R. M. Hierons, and H. Ural. Resolving observability problems in distributed test architecture. In *25rd IFIP International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2005)*, volume 3731 of *LNCS*, pages 219–232. Springer-Verlag, 2005.
6. W. Chen and H. Ural. Synchronizable checking sequences based on multiple UIO sequences. *IEEE/ACM Transactions on Networking*, 3:152–157, 1995.
7. A. Gill. *Introduction to the Theory of Finite-State Machines*. New York: McGraw-Hill, 1962.
8. S. Guyot and H. Ural. Synchronizable checking sequences based on UIO sequences. In *Proc. of IFIP IWPTS'95*, pages 395–407, Evry, France, September 1995.

9. F.C. Hennie. Fault detecting experiments for sequential circuits. In *Proc. of Fifth Ann. Symp. Switching Circuit Theory and Logical Design*, pages 95–110, Princeton, N.J., 1964.
10. R. M. Hierons. Testing a distributed system: generating minimal synchronised test sequences that detect output-shifting faults. *Information and Software Technology*, 43(9):551–560, 2001.
11. D. Lee and M. Yannakakis. Principles and methods of testing finite–state machines – a survey. *Proceedings of the IEEE*, 84(8):1089–1123, 1996.
12. S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, March-April 1973.
13. 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 139–153. Elsevier (North-Holland), 1993.
14. G. Luo, R. Dssouli, G. v. Bochmann, P. Venkataram, and A. Ghedamsi. Test generation with respect to distributed interfaces. *Computer Standards and Interfaces*, 16:119–132, 1994.
15. K.K. Sabnani and A.T. Dahbura. A protocol test generation procedure. *Computer Networks*, 15:285–297, 1988.
16. B. Sarikaya and G. v. Bochmann. Synchronization and specification issues in protocol testing. *IEEE Transactions on Communications*, 32:389–395, April 1984.
17. K.C. Tai and Y.C. Young. Synchronizable test sequences of finite state machines. *Computer Networks*, 13:1111–1134, 1998.
18. H. Ural and Z. Wang. Synchronizable test sequence generation using UIO sequences. *Computer Communications*, 16:653–661, 1993.
19. H. Ural, X. Wu, and F. Zhang. On minimizing the lengths of checking sequences. *IEEE Transactions on Computers*, 46:93–99, 1997.
20. Y.C. Young and K.C. Tai. Observation inaccuracy in conformance testing with multiple testers. In *Proc. of IEEE WASET*, pages 80–85, 1998.