Minimizing Coordination Channels in Distributed Testing

Guy-Vincent Jourdan[1], Hasan Ural[1], and Hüsnü Yenigün[2]

[1] School of Information Technology and Engineering (SITE)
University of Ottawa
800 King Edward Avenue
Ottawa, Ontario, Canada, K1N 6N5
{gvj, ural}@site.uottawa.ca
[2] Faculty of Engineering and Natural Sciences
Sabancı University
Tuzla, Istanbul, Turkey 34956
yenigun@sabanciuniv.edu

Abstract. Testing may be used to show that a system under test conforms to its specification. In the case of a distributed system, one may have to use a distributed test architecture, involving p testers in order to test the system under test. These p testers may under some circumstances have to coordinate their actions with each other using external coordination channels. This may require the use of up to $p^2 - p$ unidirectional coordination channels in the test architecture, which can be an extensive and expensive setup. In this paper, we propose a method to generate checking sequences while minimizing the number of required coordination channels, by adapting existing methods that generate checking sequences to be applied in a centralized test architecture. We consider the case of unidirectional and bidirectional coordination channels, and the case of transitive coordination.

1 Introduction

One way to check the conformance of an implementation to a specification is to employ a *checking sequence* [1,2]. Several methods have been proposed over the last two decades to reduce the length of these checking sequences, e.g. [3, 4, 5, 6], but all of these methods focus on centralized systems. When testing distributed systems, a tester is placed at each port (interface) of the system to form a distributed test architecture. During the application of a checking sequence within a distributed test architecture, the existence of multiple remote testers brings out the possibility of two types of coordination problems among testers: *controllability* and *observability* problems. These problems occur if a tester cannot determine either when to apply a particular input to a system under test (SUT), or whether a particular output from a SUT is generated in response to a specific input, respectively.

Controllability refers to the ease of affecting the specified outputs. A *controllability* (*synchronization*) *problem* exists when a tester is required to send an input in the current transition, and because it is not *involved* in the previous transition, i.e., it did not send the input or receive the output in the previous transition, it does not know when to send the input to the SUT.

Observability refers to the ease of determining if specified inputs affect the outputs. An *observability problem* exists when a tester is expecting to receive an output from the SUT in response to either the previous input or the current input and because it did not send the current input, it does not know when to start or stop waiting for the output.

Several possible venues have been explored to deal with these problems. Some authors have provided necessary and sufficient conditions to avoid controllability and/or observability problems [7, 8]. When these problems cannot be avoided, coordination among the remote testers is required through external coordination message exchanges [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]. Other authors have proposed techniques to minimize these coordination messages necessary to facilitate the application of a checking sequence in a distributed test architecture [14, 20, 21].

In this paper, we look at the use of external coordination messages from a different point of view: we adapt the algorithm of [4] to distributed testing and attempt to minimize the *number* of coordination channels required to perform the test. If p testers are involved in a distributed test architecture, potentially every pair of testers will need to exchange a coordination message at one point or another, thus leading to a need of $p^2 - p$ unidirectional coordination channels to be added to the test environment. This can potentially require an extensive and expensive setup just to run the test. Our goal is thus to require as few coordination channels as possible, in contrast to exchanging as few external coordination messages as possible. Once a channel has been set up, one would use it for exchanging as many external coordination messages as necessary rather than incurring the cost of setting up additional channels.

The rest of the paper is organized as follows: Section 2 gives the preliminaries. Section 3 reviews the checking sequence generation algorithm that will be modified to generate a checking sequence while minimizing the number coordination channels required. Section 4 presents the proposed approach. Section 5 gives the concluding remarks.

2 Preliminaries

A multiport deterministic FSM M is defined by a tuple

$$(S, s_1, p, X_1, X_2, \ldots, X_p, \delta, Y_1, Y_2, \ldots, Y_p, \lambda_1, \lambda_2, \ldots, \lambda_p)$$

in which S is a finite set of *states*, $s_1 \in S$ is the *initial state*. The number of states of M is denoted n and the states of M are enumerated, giving $S = \{s_1, \ldots, s_n\}$. $p \geq 1$ is an integer which gives the number of ports of M, and the set of ports of M is denoted $[p]$ to denote the set $\{1, 2, \ldots, p\}$.

X_i is the set of input symbols on port i such that for $j \in [p]$ and $j \neq i$, $X_i \cap X_j = \emptyset$. In other words, the input sets of the ports are disjoint. We use $X = \cup_{i \in [p]} X_i$ to denote the set of all input symbols.

$\delta : S \times X \rightarrow S$ is the *next state function*. If $s' = \delta(s,x)$ for states $s, s' \in S$ and $x \in X_i$ for some $i \in [p]$, this means that, when the machine M is in state s, and input x is applied at port i, then the machine will transfer to state s'.

Y_i is the set of output symbols on port i such that for $i, j \in [p]$ if $i \neq j$ then $Y_i \cap Y_j = \{-\}$, where $-$ is null output.

$\lambda_i : S \times X \rightarrow Y_i$ is the *output function on port* i. Intuitively, if M is at a state s, and an input $x \in X$ is applied to M, then the output $\lambda_i(s,x)$ is observed at port i, unless $\lambda_i(s,x) = -$. Let Y denote the set $Y_1 \times Y_2 \times \cdots \times Y_p \setminus (-,-,\ldots,-)$. We will also use the *output function* $\lambda : S \times X \rightarrow Y$, which is defined as $\lambda(s,x) = (\lambda_1(s,x), \lambda_2(s,x), \ldots, \lambda_p(s,x))$. We use $y|_i$ to denote the output at port $i \in [p]$ in $y \in Y$. The functions δ and λ can be extended to input sequences in a straightforward manner.

An FSM, that will be denoted M_0 throughout this paper, is shown in Figure 1. Here, $S = \{s_1, s_2, s_3, s_4, s_5\}$, $X_1 = \{a\}$, $X_2 = \{b\}$, $X_3 = \{c\}$ and $Y_1 = \{1\}$, $Y_2 = \{2\}$, $Y_3 = \{3\}$.

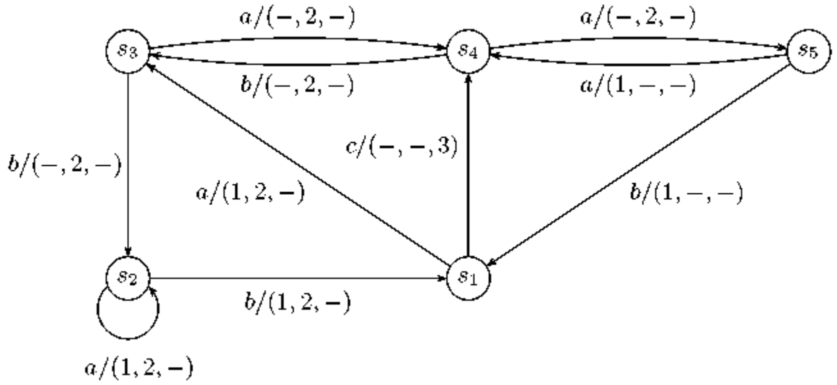

Fig. 1. The FSM M_0

Throughout the paper, we use barred symbols (e.g. $\bar{x}, \bar{P}, \ldots$) to denote sequences, and juxtaposition to denote concatenation. In an FSM M, $s_i \in S$ and $s_j \in S$, $s_i \neq s_j$, are *equivalent* if, $\forall \bar{x} \in X^*$, $\lambda(s_i, \bar{x}) = \lambda(s_j, \bar{x})$. If $\exists \bar{x} \in X^*$ such that $\lambda(s_i, \bar{x}) \neq \lambda(s_j, \bar{x})$ then $\bar{x}$ is said to *distinguish* s_i from s_j. An FSM M is said to be *minimal* if none of its states are equivalent. A *distinguishing sequence* for an FSM M is an input sequence $\bar{D}$ for which each state of M produces a distinct output. More formally, for all $s_i, s_j \in S$ if $s_i \neq s_j$ then $\lambda(s_i, \bar{D}) \neq \lambda(s_j, \bar{D})$. Thus, for example, M_0 has distinguishing sequence ab.

An FSM M can be represented by a directed graph (*digraph*) $G = (V, E)$ where a set of vertices V represents the set S of states of M, and a set of directed edges E represents all transitions of M. Each edge $e = (v_j, v_k, x/y) \in E$ represents a transition $t = (s_j, s_k, x/y)$ of M from state s_j to state s_k with input x and output y where $s_j, s_k \in S$, $x \in X$, and $y \in Y$ such that $\delta(s_j, x) = s_k$, $\lambda(s_j, x) = y$. For a vertex $v \in V$ and $E' \subseteq E$, $indegree_{E'}(v)$ denotes the number of edges from E' that enter v and $outdegree_{E'}(v)$ denotes the number of edges from E' that leave v.

A sequence $\bar{P} = (n_1, n_2, x_1/y_1)(n_2, n_3, x_2/y_2) \ldots (n_{k-1}, n_k, x_{k-1}/y_{k-1})$ of pairwise adjacent edges from G forms a *path* in which each *node* n_i represents a vertex from V and thus, ultimately, a state from S. Here $initial(\bar{P})$ denotes n_1, which is the *initial node* of $\bar{P}$, and $final(\bar{P})$ denotes n_k, which is the *final node* of $\bar{P}$. The sequence $\bar{Q} = (x_1/y_1)(x_2/y_2) \ldots (x_{k-1}/y_{k-1})$ is the *label* of $\bar{P}$ and is denoted $label(\bar{P})$. $\bar{Q}$ is said to be a *transfer sequence* from n_1 to n_k. The path $\bar{P}$ can be represented by the tuple $(n_1, n_k, \bar{Q})$ or by the tuple $(n_1, n_k, \bar{x}/\bar{y})$ in which $\bar{x} = x_1 x_2 \ldots x_{k-1}$ is the *input portion* of $\bar{Q}$ and $\bar{y} = y_1 y_2 \ldots y_{k-1}$ is the *output portion* of $\bar{Q}$. The *cost* of a path is given as the number of pairs of input/output symbols in its label. Two paths $\bar{P}_1$ and $\bar{P}_2$ can be concatenated as $\bar{P}_1\bar{P}_2$ only if $final(\bar{P}_1) = initial(\bar{P}_2)$.

A *tour* is a path whose initial and final nodes are the same. Given a tour $\bar{\Gamma} = e_1 e_2 \ldots e_k$, $\bar{P} = e_j e_{j+1} \ldots e_k e_1 e_2 \ldots e_{j-1}$ is a path formed by *starting* $\bar{\Gamma}$ with edge e_j, and hence by *ending* $\bar{\Gamma}$ with edge e_{j-1}. An *Euler Tour* of G is a tour that contains each edge of G exactly once. A set E' of edges from G is *acyclic* if no tour can be formed using the edges in E'. A digraph $G = (V, E)$ is *symmetric* if for each vertex $v \in V$, *indegree*$_E$*(v)=outdegree*$_E$*(v)*. A minimum-cost symmetric augmentation $G' = (V, E')$ of a graph $G = (V, E)$ is a symmetric digraph where $E' = E \cup \Delta$, where Δ contains a minimum number of replications of some edges in E.

A digraph is *strongly connected* if for any ordered pair of vertices (v_i, v_j) there is a path from v_i to v_j. An FSM is *strongly connected* if the digraph that represents it is strongly connected. It will be assumed that any FSM considered in this paper is deterministic, minimal, and strongly connected.

3 Overview of the Original Algorithm

In this section, we will present an existing approach for generating reduced length checking sequences [4]. The method, in its original form, does not take into account the fact that the resulting checking sequence will be applied in a distributed test architecture. To apply it on a distributed test architecture, external coordination messages must be inserted into the checking sequence. Hence, it may be used to generate a checking sequence applicable within a distributed test architecture which uses more coordination channels than it may actually be needed. We will show in the next section how to modify the method to generate checking sequences minimizing the number of coordination channels required.

Let M be an FSM and let N be an implementation of M. A *checking sequence* is a sequence of inputs to be applied to N that will help determine whether N is a correct implementation of M or not, i.e. whether N is isomorphic to M or not [1]. If M has a distinguishing sequence $\bar{D}$, then $\bar{D}$ can be used in the checking sequence to help to identify the states. Let us call $\bar{T}_i = \bar{D}/\lambda(s_i, \bar{D})\bar{B}_i$ a T*–sequence,* where $\bar{B}_i = \bar{I}_i/\lambda(\delta(s_i, \bar{D}), \bar{I}_i)$ for a possibly empty input sequence $\bar{I}_i$ (i.e. the input portion of a transfer sequence). We call *initial(T)* (resp. *final(T)* the first (resp. last) state of the sequence T. An α'*–sequence* is a sequence $\bar{T}_{k_1}\bar{T}_{k_2} \ldots \bar{T}_{k_{r_k}}$, for some $1 \leq k_1, k_2, \ldots, k_{r_k} \leq n$, such that $\forall i \in \{1, 2, \ldots, r_k - 1\}$, *initial(*$\bar{T}_{k_{i+1}}$*)* $=$

final($\bar{T}_{k_i}$). A T*–set* is a set of T*–sequences*, and an α'*–set* is a set of α'*–sequences* $\{\bar{\alpha}'_1, \bar{\alpha}'_2, \ldots, \bar{\alpha}'_q\}$ satisfying the following condition [4]: for all $i \in \{1, 2, \ldots, n\}$, there exists $j \in \{1, 2, \ldots, n\}$ and $k \in \{1, 2, \ldots, q\}$, such that $\bar{T}_i\bar{T}_j$ is a subsequence of $\bar{\alpha}'_k$.

In [4], the following method is explained to produce a checking sequence. Given a digraph $G = (V, E)$ corresponding to an FSM M, a T–set $\mathcal{T} = \{\bar{T}_1, \bar{T}_2, \ldots, \bar{T}_n\}$, and an α'–set $A = \{\bar{\alpha}'_1, \bar{\alpha}'_2, \ldots, \bar{\alpha}'_q\}$, first another digraph $G' = (V', E')$ is produced by augmenting the digraph G as follows (Figure 2 is the digraph G' corresponding to the digraph G of FSM M_0 given in Figure 1, where the input portion of $\bar{T}_1, \bar{T}_2, \bar{T}_3, \bar{T}_4$ and $\bar{T}_5$ is ab):

a) $V' = V \cup U'$ where $U' = \{v' : v \in V\}$, i.e. for each vertex v in G, there are two copies of v in G'. In Figure 2, the nodes at the bottom are the nodes in V, and the nodes at the top are the nodes in U'.
b) $E' = E_C \cup E_T \cup E_{\alpha'} \cup E''$ where
   i) $E_C = \{(v'_i, v_j, x/y) : (v_i, v_j, x/y) \in E\}$. The solid edges leaving the nodes at the top in Figure 2 are the edges in E_C.
   ii) $E_T = \{(v_i, v'_j, \bar{T}_i) : \bar{T}_i \in \mathcal{T}, s_i = initial(\bar{T}_i), s_j = final(\bar{T}_i)\}$. In Figure 2, these edges are shown with dashed lines.
   iii) $E_{\alpha'} = \{(v_i, v'_j, \bar{\alpha}'_k) : \bar{\alpha}'_k \in A, \bar{\alpha}'_k = \bar{T}_i \ldots \bar{T}_j, initial(\bar{T}_i) = s_i, final(\bar{T}_j) = s_j\}$. For example, in Figure 2 we consider an α'–set $A = \{\bar{\alpha}'_1 = \bar{T}_4\bar{T}_1\bar{T}_2\bar{T}_1, \bar{\alpha}'_2 = \bar{T}_5\bar{T}_3\bar{T}_3\}$. The bold solid edges in Figure 2 are the edges of $E_{\alpha'}$.
   iv) $E'' \subsetneq \{(v'_i, v'_j, x/y) : (v_i, v_j, x/y) \in E\}$. E'' is a subset of the copies of the edges in E placed between the corresponding nodes in U'. E'' is selected in such a way that, $G'' = (U', E'')$ does not have a tour and G' is strongly connected. These edges are shown in Figure 2 with dotted lines.

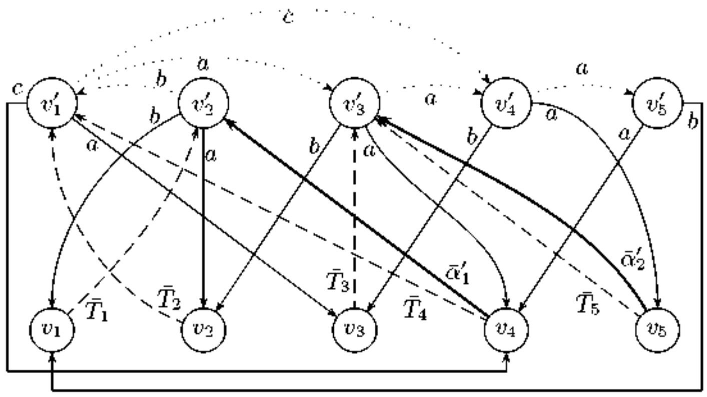

Fig. 2. G' for M_0, with outputs omitted. The edges in $E_{\alpha'}$, E_C, E_T, and E'' are shown as bold solid lines, solid lines, dashed lines, and dotted lines, respectively.

In [4], it is shown that the input portion of the label of a path $\bar{P}$ in G' that starts from v_1 and ends at v_1, that includes all the edges in $E_{\alpha'}$, and all the edges in E_C (that is, the solid lines in Figure 2) and that is followed by $\bar{D}$ is a checking sequence of M.

In order to reduce the length of the resulting checking sequence, an optimization algorithm may be used. The method given in [4] forms a minimum-cost symmetric augmentation G^* of the digraph induced by $E_{\alpha'} \cup E_C$ by adding replications of edges from E' . If G^*, with its isolated vertices removed, is connected, then G^* has an Euler tour. Otherwise, a heuristic such as the one given in [3] is applied to make G^* connected and an Euler tour of this new digraph is formed to find a path from v_1 to v_1. A checking sequence is then constructed based on the Euler tour as the input portion of the label of the path from v_1 to v_1 followed by $\bar{D}$.

$$\begin{array}{llllllll}
(s_1,s_3,a/(1,2,-)) & & (s_3,s_2,b/(-,2,-)) & & (s_2,s_1,b/(1,2,-)) & & (s_1,s_3,a/(1,2,-)) & \\
(s_3,s_2,b/(-,2,-)) & \star & (s_2,s_2,a/(1,2,-)) & & (s_2,s_2,a/(1,2,-)) & & (s_2,s_1,b/(1,2,-)) & \star \\
(s_1,s_4,c/(-,-,3)) & \star & (s_4,s_5,a/(-,2,-)) & & (s_5,s_1,b/(1,-,-)) & & (s_1,s_3,a/(1,2,-)) & \\
(s_3,s_2,b/(-,2,-)) & \star & (s_2,s_2,a/(1,2,-)) & & (s_2,s_1,b/(1,2,-)) & & (s_1,s_3,a/(1,2,-)) & \\
(s_3,s_2,b/(-,2,-)) & & (s_2,s_1,b/(1,2,-)) & & (s_1,s_3,a/(1,2,-)) & & (s_3,s_2,b/(-,2,-)) & \star \\
(s_2,s_2,a/(1,2,-)) & & (s_2,s_1,b/(1,2,-)) & & (s_1,s_3,a/(1,2,-)) & & (s_3,s_4,a/(-,2,-)) & \\
(s_4,s_3,b/(-,2,-)) & \star & (s_3,s_4,a/(-,2,-)) & & (s_4,s_5,a/(-,2,-)) & & (s_5,s_1,b/(-,2,-)) & \star \\
(s_1,s_4,c/(-,-,3)) & \star & (s_4,s_3,b/(-,2,-)) & \star & (s_3,s_4,a/(-,2,-)) & & (s_4,s_3,b/(-,2,-)) & \star \\
(s_3,s_4,a/(-,2,-)) & & (s_4,s_5,a/(-,2,-)) & & (s_5,s_4,a/(1,-,-)) & \star & (s_4,s_3,b/(-,2,-)) & \star \\
(s_3,s_4,a/(-,2,-)) & & (s_4,s_3,b/(-,2,-)) & \star & (s_3,s_4,a/(-,2,-)) & & (s_4,s_3,b/(-,2,-)) & \star \\
(s_3,s_4,a/(-,2,-)) & & (s_4,s_5,a/(-,2,-)) & & (s_5,s_4,a/(1,-,-)) & & (s_4,s_5,a/(-,2,-)) & \\
(s_5,s_1,b/(1,-,-)) & \star & (s_1,s_4,c/(-,-,3)) & \star & (s_4,s_5,a/(-,2,-)) & & (s_5,s_1,b/(1,-,-)) & \\
(s_1,s_3,a/(1,2,-)) & & (s_3,s_2,b/(-,2,-)) & & & & &
\end{array}$$

Fig. 3. The transition sequence on FSM M_0 (given in Figure 1) corresponding to the checking sequence produced from G' (given in Figure 2). Two consecutive transitions with a $\star$ between them have a synchronization problem.

4 The Proposed Approach

A checking sequence constructed by the method reviewed in the previous section requires insertion of the external coordination message exchanges to be applicable by remote testers in a distributed test architecture without encountering controllability and observability problems.

Formally, a (*controllability*) *synchronization problem* occurs when, in the labels x_i/y_i and x_j/y_j of any two adjacent transitions, there exists a tester l that sends x_j that is neither the one sending x_i nor one of those receiving an output belonging to y_i. Let tester k be the tester that sends x_i . In general, the solution to the synchronization problem is to insert an external coordination message exchange relating to controllability between x_i/y_i and x_j/y_j from tester k to tester l to notify tester l to send its input to the SUT [15].

Any two consecutive transitions t_{ij} and t_{jk} whose labels are x_i/y_i and x_j/y_j in a sequence of transitions form a *synchronizable pair* of transitions if t_{jk} can follow t_{ij} without generating a synchronization problem. Any sequence of transitions in which every pair of consecutive transitions is synchronizable is called a *synchronizable transition sequence*. An input/output sequence is said to be *synchronizable* if it is the label of a synchronizable transition sequence.

The observability problem manifests itself as an undetectable *output shift fault*. Formally, given a synchronizable transition sequence $t_1 \ldots t_k$ 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 $1 \leq i < j \leq k$:

1. There exists $m \in [p]$ and $y_i|_m = o \in Y_m \setminus \{-\}$ in M, for all $i < l \leq j$ we have that $y_l|_m = -$ in M, for all $i \leq l < j$ we have that N produces output $-$ at m in response to x_l after $x_1 \ldots x_{l-1}$, and N produces output o at m in response to x_j after $x_1 \ldots x_{j-1}$. Here the output o shifts from being produced in response to x_i to being produced in response to x_j and the (forward) shift is from t_i to t_j.
2. There exists $m \in [p]$ and $o \in Y_m \setminus \{-\}$ such that $y_j|_m = o$ in M, for all $i \leq l < j$ we have that $y_l|_m = -$ in M, for all $i < l \leq j$ we have that N produces output $-$ at m in response to x_l after $x_1 \ldots x_{l-1}$, and N produces output o at m in response to x_i after $x_1 \ldots x_{i-1}$. Here the output o shifts from being produced in response to x_j to being produced in response to x_i and the (backward) shift is from t_j to t_i.

An instance of the observability problem manifests itself as a *potentially undetectable output shift fault* if there is an output shift fault related to $o \in Y_m$ in two transitions with labels x_i/y_i and x_j/y_j, such that $x_{i+1} \ldots x_j \notin X_m$. The tester at m will not be able to detect the faults since it will observe the expected sequence of interactions in response to $x_i \ldots x_j$. Let tester h be the tester that sends x_j. In general, the solution to the observability problem is to insert an external coordination message exchange relating to observability between x_i/y_i and x_j/y_j from tester h to tester m:

- Case 1: ($y_i|_m = o$ in M) By this exchange, tester h informs tester m that it must have received an output from the SUT by now.
- Case 2: ($y_j|_m = o$ in M) By this exchange, tester h informs tester m to expect receiving an output from the SUT [15].

In most cases, insertion of an external coordination message exchange relating to observability can be avoided by appending additional input/output subsequences to the label of the path whose input portion will be used as a checking sequence [8]. Therefore, we will focus only on the controllability problem in the rest of the paper.

The algorithm in [4] is intended for a centralized test architecture, and hence in a distributed test architecture, some portions of the sequence generated by the algorithm may lead to synchronization problems. We thus need to adapt the algorithm to the distributed test architecture by modifying it in two ways: on one hand, we try to select checking sequences that do not cause synchronization problems (recognizing the fact that they might then be longer than the ones requiring synchronization); on the other end we need to add some coordination channels, when a synchronization problem cannot be avoided directly.

Note that there can be different types of coordination channels (unidirectional or bidirectional) and the relaying of coordination messages through other testers using available coordination channels (transitive coordinations between testers) may or may not be allowed. We will first examine the case of unidirectional coordination channels without transitive communications. The other cases will be explored in Section 4.3.

4.1 Modifying the Digraph G'

Briefly, our approach consists of modifying the digraph G' being built so that only possible (synchronizable) transition sequences are available, and use that modified digraph to build a checking sequence in which no controllability problem exists. If it is not possible to generate a checking sequence with the current form of the digraph, then additional coordination channels are added (which in turn modifies the digraph and allows more consecutive pairs of transitions to be executed without synchronization problems), until a digraph is formed that allows building a checking sequence without any synchronization problem.

We first modify the digraph G' by replacing each edge $(v_i, v'_j, x_1x_2\ldots x_k/y_1y_2\ldots y_k) \in E_T \cup E_{\alpha'}$ with a sequence of edges $(v_i, u^i_1, x_1/y_1)$, $(u^i_1, u^i_2, x_2/y_2), (u^i_2, u^i_3, x_3/y_3), \ldots, (u^i_{k-2}, u^i_{k-1}, x_{k-1}/y_{k-1}), (u^i_{k-1}, v'_j, x_k/y_k)$ where $u^i_1, u^i_2, \ldots, u^i_{k-1}$ are new nodes introduced into the digraph. Let us call this new digraph as G''. Note that any path in G' will have a corresponding path in G'' and vice versa. In fact the only difference between G' and G'' is that, we have inserted explicit nodes along the edges in G' whose labels are not single input/output symbols. Therefore, in G'' all the edges will have a single input/output symbol pair as their labels.

Note the algorithm in Section 3 finds a tour over the edges $E_{\alpha'} \cup E_C$. Let us call these edges *the essential edges in* G'. We also call an edge in G'' an *essential edge in* G'', if it is an edge in E_C, or it is an edge that we insert into G'' as we create the edges corresponding to the individual steps along an edge in $E_{\alpha'}$.

Let $e_1 = (u_1, u, x_1/y_1)$ and $e_2 = (u, u_2, x_2/y_2)$ be two edges in G''. Note that the algorithm given in Section 3 may produce a checking sequence in which e_1 is immediately followed by e_2 since e_1 ends at and e_2 starts at vertex u. However, we would like to allow the possibility of having e_1 being followed by e_2 only if it is possible to do so without creating a synchronization problem.

In order to set up the digraph in such a way that, e_1 can be followed by e_2 only without creating a synchronization problem, we derive another digraph $G''' = (V''', E''')$ which is the *interchange graph* (or *line graph*) of G''. In other words, each edge $(u_1, u_2, x/y)$ in G'', becomes a vertex $(u_1, u_2, x/y) \in V'''$. For two nodes $(u_1, u_2, x/y)$ and $(u'_1, u'_2, x'/y')$ in V''', $((u_1, u_2, x/y), (u'_1, u'_2, x'/y'), \epsilon) \in E'''$ if and only if $u'_1 = u_2$. We also have the mapping $R : E''' \rightarrow 2^{[p]\times[p]}$ that maps an edge in E''' to a set of coordination channels. Intuitively, for an edge $((u_1, u, x/y), (u, u'_2, x'/y'), \epsilon) \in E'''$, $R((u_1, u, x/y), (u, u'_2, x'/y'), \epsilon)$ is the set of coordination channels such that if any one of these coordination channels exist, then $(u, u'_2, x'/y')$ can follow $(u_1, u, x/y)$ without a synchronization problem.

A vertex $(u_1, u_2, x/y) \in V'''$ is called an *essential vertex in* G''' if the edge $(u_1, u_2, x/y)$ in G'' is an essential edge in G''.

A subset $C \subseteq [p] \times [p]$ of coordination channels induces a digraph $G'''_C = (V''', E'''_C)$ where $E'''_C \subseteq E'''$, such that for an edge $((u_1, u, x/y), (u, u'_2, x'/y'), \epsilon) \in E'''$, $((u_1, u, x/y), (u, u'_2, x'/y'), \epsilon) \in E'''_C$ iff $R((u_1, u, x/y), (u, u'_2, x'/y'), \epsilon) = \emptyset$ (no coordination channel is required) or $R((u_1, u, x/y), (u, u'_2, x'/y') \cap C \neq \emptyset$ (at least one of the required coordination channels is available).

Then, our problem can be formulated as to find a set $C \subseteq [p] \times [p]$ of coordination channels with minimal cardinality such that G'''_C has a strongly connected component which includes all the essential vertices in G'''. When G'''_C has a strongly connected component which includes all the essential vertices in G''', we can find a tour in this strongly connected component that visits all these essentials vertices. Thanks to the way we constructed G''', this tour indeed corresponds to a tour in G'' that includes all the essential edges of G'', and therefore corresponds to a tour in G' which includes all the edges in E_C and $E_{\alpha'}$ that can thus be used to generate a checking sequence.

Thus we need to build a set $C \subseteq [p] \times [p]$ of coordination channels. If $(i, j) \in C$, for $i, j \in [p]$, this means that a coordination channel exists from the tester at port i to the tester at port j. Two successive transitions with labels x_1/y_1 and x_2/y_2, where $x_1 \in X_i$ and $x_2 \in X_j$ for some $i, j \in [p]$, are *synchronizable* if and only if:

1. $i = j$; or
2. $y_1|_j \neq -$; or
3. $(i, j) \in C$; or
4. $\exists k \in [p]$ such that $y_1|_k \neq -$ and $(k, j) \in C$

In the first two cases, the synchronization is achieved without using any external coordination message exchanges. In the last two cases, the synchronization is done externally, either by having the tester at port i (the sender of the input of the first transition) send an external coordination message to the tester at port j, or by having the tester at some port k, which receives a non–null output due to the first transition, send an external coordination message to the tester at port j.

4.2 A Heuristic to Find the Coordination Channels

As explained in Section 4.1, the original problem has been reformulated as finding a set $C \subseteq [p] \times [p]$ of coordination channels with minimal cardinality such that G'''_C has a strongly connected component which includes all the essential vertices in G'''.

Initially, $C = \emptyset$. If $G'''_\emptyset$ has a strongly connected component which includes all the essential vertices in G''', then we are done and by using $G'''_\emptyset$ we can construct a checking sequence that does not require any coordination channels at all. If that is not the case, then coordination channels must be added in order to add more edges in G'''_C, until such a strongly connected component can be found (note that if $C_1 \subseteq C_2$ then the edges of G'''_{C_1} are included in the edges of G'''_{C_2}, that is, by adding new coordination channels we add edges to G'''). In the worst case scenario, we will add coordination channels between every pair of testers, which will in effect put us back in the case studied in [4].

In order to decide which coordination channels to add, we propose the following heuristic, that converges to a solution while trying to add as few channels as possible. We start from the digraph G'''_C, with $C = \emptyset$. We first create the *condensation* of G'''_C: the condensation of G'''_C is a graph $\widehat{G}'''_C$ containing one vertex for each strongly connected component of G'''_C. Two vertices representing components are joined by an edge in $\widehat{G}'''_C$ if there is an edge in G'''_C from a vertex in one component to a vertex in the other. Such an acyclic condensation can be built in $O(V''' + E''')$ [22]. In [23], it is shown that finding a graph augmentation (that is, adding new edges) that strongly connects $\widehat{G}'''_C$ is equivalent to finding an augmentation that strongly connects G'''_C, by using a mapping between a vertex of $\widehat{G}'''_C$ and any vertex of the corresponding strongly connected component in G'''_C. It is also pointed out in [23] that in order to strongly connect a condensed graph we necessarily need to add outgoing edges to each of its sink and isolated vertices, and incoming edges to each of its source and isolated vertices.

We proceed as follows, starting with $C = \emptyset$:

- Condense the current graph G'''_C into $\widehat{G}'''_C$.
- Identify the set Φ of sources, sinks and isolated vertices in $\widehat{G}'''_C$ that are issued from the condensation of strongly connected components of G'''_C containing essential vertices. If Φ is not empty, then let Φ_1 be the set of such sources, Φ_2 be the set of such sinks and Φ_3 be the set of such isolated vertices. Otherwise (Φ is empty), let Φ_1 be the set of sources, Φ_2 be the set of sinks and Φ_3 be the set of isolated vertices in $\widehat{G}'''_C$.
- For each possible new coordination channel: count the number of elements from Φ_1 and Φ_3 that will have at least one outgoing edge added to them by the inclusion of the channel to C, and the number of elements of Φ_2 and Φ_3 that will have at least one incoming edge added to them by the inclusion of the channel to C; identify the coordination channel c that will maximize this number (that is, identify the coordination channel c that would remove the most sources, sinks and isolated vertices from $\widehat{G}'''_{C\cup\{c\}}$).
- Add this coordination channel c to C.
- Re-calculate the digraph G'''_C based on the new set C. If G'''_C still has not a strongly connected component which includes all the essential vertices in G''', then re-iterate the process. Else, stop.

It is clear that the above process converges toward a solution, since in the worst case we end up adding every pair to C. The solution found will be correct according to [4], will not contain any synchronization issues by construction and may require the addition of fewer coordination channels than simply synchronizing the original solution found in [4] (the solution may however be longer than the one found originally).

We illustrate our approach with the example given in Figure 4. This figure shows a digraph which stands for an example of G'' (although we do not show the labels of the transitions for simplicity). Assume that in this example, the transition d has a synchronization problem with the transition e, and that transition f has a synchronization problem with the transition g; every other transition pair is synchronizable. Without adding any coordination channel, the graph $G'''_{\emptyset}$ shown in Figure 6 is obtained; note that d is not connected to e, and f is not connected to g. The digraph is then condensed into $\widehat{G}'''_{\emptyset}$ shown in Figure 7. It has four vertices, showing that $G'''_{\emptyset}$ is not strongly connected.

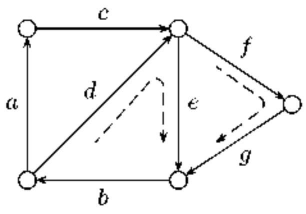

Fig. 4. A sample graph G'', with i/o labels not shown. Assume that transition pairs (d, e) and (f, g) have synchronization problems.

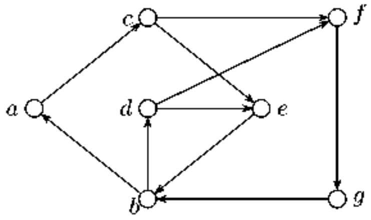

Fig. 5. The graph G''' corresponding to G'' of Figure 4

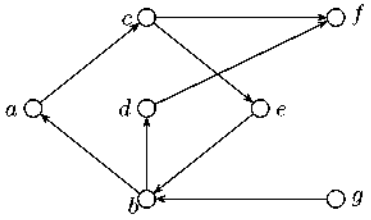

Fig. 6. The graph $G'''_{\emptyset}$ corresponding to G'' of Figure 4

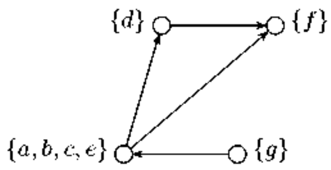

Fig. 7. The graph $\widehat{G}'''_{\emptyset}$ corresponding to $G'''_{\emptyset}$ of Figure 6

Assume now that the addition of the coordination channel c_1 allows the synchronization of (d, e) (Figure 8). Adding such a coordination channel would not add any outgoing edges from sinks nor incoming edges to sources of $\widehat{G}'''_{\emptyset}$. The digraph $\widehat{G}'''_{\{c_1\}}$ is shown in Figure 9.

Assume that the addition of the coordination channel (c_2) allows the synchronization of (f, g) (Figure 10). This time, adding such a coordination channel would add an outgoing edge from sink $\{f\}$ and an incoming edge to source $\{g\}$ in $\widehat{G}'''_{\emptyset}$. Thus c_2 will be chosen over c_1, and the resulting digraph $\widehat{G}'''_{\{c_2\}}$ is shown in Figure 11: everything collapses in a single vertex, showing that $G'''_{\{c_2\}}$ is now strongly connected, and an Euler path can be found.

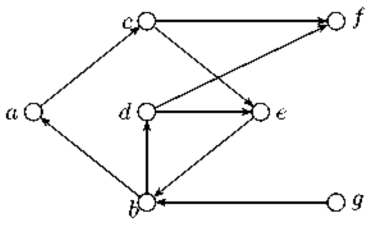

Fig. 8. The graph $G'''_{\{c_1\}}$: the coordination channel c_1 resolves the synchronization problem in transition pair (d, e)

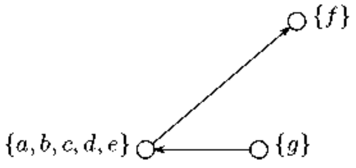

Fig. 9. The graph $\widehat{G}'''_{\{c_1\}}$ corresponding to $G'''_{\{c_1\}}$ of Figure 8. Sources and sinks are not impacted by the addition

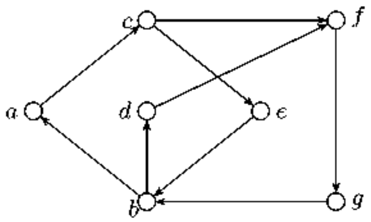

Fig. 10. The graph $G'''_{\{c_2\}}$: the coordination channel c_2 resolves the synchronization problem in transition pair (f, g)

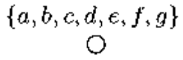

Fig. 11. The graph $\widehat{G}'''_{\{c_2\}}$ corresponding to $G'''_{\{c_2\}}$ of Figure 10: the graph is now strongly connected

Going back to the example related to FSM M_0 (given in Figure 1), the following set of coordination channels is required to make the checking sequence given in Figure 3 synchronized:

$$C = \{(1, 2), (2, 1), (2, 3), (3, 2), (3, 1)\}$$

However, for the same FSM, applying the proposed approach to the digraph G' given in Figure 2 will yield the following set of coordination channels to make the checking sequence given in Figure 12 synchronized:

$$C = \{(1, 2), (2, 1), (3, 1), (3, 1)\}$$

Note that, the length of the checking sequence given in Figure 3 is 50. The checking sequence given in Figure 12 requires one less coordination channel, but one more input symbol, making its length 51. Also note that, the number of coordination messages exchanged is reduced by 1. However this reduction is only coincidental, as the method proposed does not aim at reducing the number of coordination messages.

4.3 Different Types of Synchronization Strategies

In the discussion presented so far, it is assumed that coordination channels are unidirectional, and thus coordination occurs only in one direction over a single

$$\begin{array}{llllllll}
(s_1,s_3,a/(1,2,-)) & & (s_3,s_2,b/(-,2,-)) & & (s_2,s_1,b/(1,2,-)) & & (s_1,s_3,a/(1,2,-)) & \\
(s_3,s_2,b/(-,2,-)) & \star & (s_2,s_2,a/(1,2,-)) & & (s_2,s_2,a/(1,2,-)) & & (s_2,s_1,b/(1,2,-)) & \star \\
(s_1,s_4,c/(-,-,3)) & \star & (s_4,s_5,a/(-,2,-)) & & (s_5,s_1,b/(1,-,-)) & & (s_1,s_3,a/(1,2,-)) & \\
(s_3,s_2,b/(-,2,-)) & \star & (s_2,s_2,a/(1,2,-)) & & (s_2,s_1,b/(1,2,-)) & & (s_1,s_3,a/(1,2,-)) & \\
(s_3,s_2,b/(-,2,-)) & & (s_2,s_1,b/(1,2,-)) & & (s_1,s_3,a/(1,2,-)) & & (s_3,s_2,b/(-,2,-)) & \star \\
(s_2,s_2,a/(1,2,-)) & & (s_2,s_1,b/(1,2,-)) & & (s_1,s_3,a/(1,2,-)) & & (s_3,s_4,a/(-,2,-)) & \\
(s_4,s_3,b/(-,2,-)) & \star & (s_3,s_4,a/(-,2,-)) & & (s_4,s_5,a/(-,2,-)) & & (s_5,s_1,b/(-,2,-)) & \star \\
(s_1,s_3,a/(1,2,-)) & & (s_3,s_4,a/(-,2,-)) & & (s_4,s_3,b/(-,2,-)) & \star & (s_3,s_4,a/(-,2,-)) & \\
(s_4,s_3,b/(-,2,-)) & \star & (s_3,s_4,a/(-,2,-)) & & (s_4,s_5,a/(-,2,-)) & & (s_5,s_4,a/(1,-,-)) & \star \\
(s_4,s_3,b/(-,2,-)) & \star & (s_3,s_4,a/(-,2,-)) & & (s_4,s_3,b/(-,2,-)) & \star & (s_3,s_4,a/(-,2,-)) & \\
(s_4,s_3,b/(-,2,-)) & \star & (s_3,s_4,a/(-,2,-)) & & (s_4,s_5,a/(-,2,-)) & & (s_5,s_4,a/(1,-,-)) & \\
(s_4,s_5,a/(-,2,-)) & & (s_5,s_1,b/(1,-,-)) & \star & (s_1,s_4,c/(-,-,3)) & \star & (s_4,s_5,a/(-,2,-)) & \\
(s_5,s_1,b/(1,-,-)) & & (s_1,s_3,a/(1,2,-)) & & (s_3,s_2,b/(-,2,-)) & & &
\end{array}$$

Fig. 12. The transition sequence on FSM M_0 (given in Figure 1) of the checking sequence produced from G'''. Two consecutive transitions with a $\star$ between them have a synchronization problem.

channel. In other words, C contains *ordered* pairs of testers (i, j), allowing an external coordination message to be sent from i to j but not from j to i. Also having (i, j) and (j, k) in C does not insure that a pair of transitions involving testers i and k can necessarily be synchronized.

It is however possible to modify our approach to accommodate for both cases.

Bidirectional Synchronization Channels. If coordination channels are bidirectional, then once a pair of testers (i, j) is added to C, any pair of transitions involving testers i and j can be synchronized, be it from i to j or from j to i (in both directions). Thus, in that scenario, two successive transitions with labels x_1/y_1 and x_2/y_2, where $x_1 \in X_i$ and $x_2 \in X_j$, $i, j \in [p]$, are synchronizable if and only if (let C^s denote the symmetric closure of C below):

1. $i = j$; or
2. $y_1|_j \neq -$; or
3. $(i, j) \in C^s$; or
4. $\exists k \in [p]$ such that $y_1|_k \neq -$ and $(k, j) \in C^s$

Adapting the algorithm to this new definition is straightforward. We only need to consider G'''_{C^s} instead of G'''_C while checking if all the essential nodes are in a single strongly connected component.

Transitive Synchronizations. It is possible for the testers to organize a synchronization strategy allowing the coordination of two testers in the absence of direct coordination channels between them, by using a chain of external coordination messages involving other testers. Such a transitive synchronization strategy can be considered both with unidirectional and bidirectional coordination channels.

When we consider unidirectional channels with transitive strategy, two successive transitions with labels x_1/y_1 and x_2/y_2, where $x_1 \in X_i$ and $x_2 \in X_j$,

$i, j \in [p]$, are synchronizable if and only if (let C^t denote the transitive closure of C below) :

1. $i = j$; or
2. $y_1|_j \neq -$; or
3. $(i, j) \in C^t$; or
4. $\exists k \in [p]$ such that $y_1|_k \neq -$ and $(k, j) \in C^t$

When we consider bidirectional channels with transitive strategy, two successive transitions with labels x_1/y_1 and x_2/y_2, where $x_1 \in X_i$ and $x_2 \in X_j$, $i, j \in [p]$, are synchronizable if and only if (let C^{st} denote the symmetric and transitive closure of C below) :

1. $i = j$; or
2. $y_1|_j \neq -$; or
3. $(i, j) \in C^{st}$; or
4. $\exists k \in [p]$ such that $y_1|_k \neq -$ and $(k, j) \in C^{st}$

Adapting the algorithm to this new definition is also straightforward. We only need to consider G'''_{C^t} or $G'''_{C^{st}}$ instead of G'''_C while checking if all the essential nodes are in a single strongly connected component.

Note that if establishing new coordination channel is considered costly, then following such a transitive synchronization strategy is certainly worthwhile, since it can significantly lower the number of coordination channels required.

5 Conclusions

We have presented an approach to minimize the number of coordination channels in a distributed test architecture for the application of a checking sequence. The proposed approach is presented as a modification of an existing method for constructing a checking sequence, but can be adapted to work with any other method that constructs a checking sequence by finding a tour on an auxiliary graph derived from a finite state machine specification of the application.

The heuristic algorithm explained above finds a set of coordination channels C such that G'''_C has the required property (i.e. having all the essential vertices in a single strongly connected component). However, since it is a greedy heuristic algorithm, it may accumulate a set of coordination channels which may have a subset C' that will yield $G'''_{C'}$ with the required property. To find such a subset of C, will require a post–processing phase. The nature of this processing phase is explained as follows:

Note that inclusion of a coordination channel $(i, j) \in [p] \times [p]$ in C inserts a set of edges in G'''_C. Namely, an edge $e \in E'''$ is inserted in G'''_C due to the inclusion of (i, j) in C, if $(i, j) \in R(e)$. The same edge e can be inserted by the inclusion of some other coordination channels in $R(e)$ into C as well.

Let $R^{-1}(i, j) = \{e \in E''' \mid (i, j) \in R(e)\}$, i.e. $R^{-1}(i, j)$ is the set of edges inserted by the coordination channel (i, j). Given a set of coordination channels

C, let $R^{-1}(C) = \cup_{(i,j)\in C} R^{-1}(i,j)$. For two sets of coordination channels C' and C such that $C' \subseteq C$ but $R^{-1}(C') = R^{-1}(C)$, it is obvious that $G'''_{C'}$ has all the essential vertices in a single strongly connected component iff G'''_{C} does, because both C' and C insert the same set of edges. This is an instance of the set cover problem, which is known to be NP–complete. An existing heuristic algorithm for the set cover problem can be used to find a minimal subset C' of C such that $R^{-1}(C') = R^{-1}(C)$.

Acknowledgments

This work is supported in part by the Natural Science and Engineering Research Council of Canada under grants RGPIN 976 and RGPIN 312018, CITO/OCE of the Government of Ontario, and a grant by Sabancı University.

References

1. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines - A survey. In: Proceedings of the IEEE. Volume 84. (1996) 1090–1126
2. Gill, A.: Introduction to The Theory of Finite State Machines. McGraw Hill, New York (1962)
3. Ural, H., Wu, X., Zhang, F.: On minimizing the length of checking sequences. IEEE Transactions on Computers **46** (1997) 93–99
4. Hierons, R.M., Ural, H.: Reduced length checking sequences. IEEE Transactions on Computers **51** (2002) 1111–1117
5. Tekle, K.T., Ural, H., Yalcin, M.C., Yenigun, H.: Generalizing redundancy elimination in checking sequences. In: ISCIS 2005, LNCS 3733. (2005) 915–926
6. Yao, M., Petrenko, A., v. Bochmann, G.: Conformance testing of protocol machines without reset. In: Protocol Specification, Testing and Verification. Volume XIII. (1993) 241–256
7. Chen, J., Hierons, R., Ural, H.: Conditions for resolving observability problems in distributed testing. In: FORTE 2004, LNCS 3235. (2004) 229–242
8. Chen, X.J., Hierons, R.M., Ural, H.: Resolving observability problems in distributed test architecture. In: IFIP FORTE 2005, LNCS 3731. (2005) 219–232
9. Sarikaya, B., v. Bochmann, G.: Synchronization and specification issues in protocol testing. IEEE Transactions on Communications **32** (1984) 389–395
10. Luo, G., Dssouli, R., Bochmann, G.V., Venkataram, P., Ghedamsi, A.: Test generation with respect to distributed interfaces. Comput. Stand. Interfaces **16** (1994) 119–132
11. Tai, K., Young, Y.: Synchronizable test sequences of finite state machines. Computer Networks and ISDN Systems **30** (1998) 1111–1134
12. Hierons, R.M.: Testing a distributed system: Generating minimal synchronised test sequences that detect output-shifting faults. Information and Software Technology **43** (2001) 551–560
13. Khoumsi, A.: A temporal approach for testing distributed systems. Software Engineering, IEEE Transactions on **28** (2002) 1085–1103
14. Wu, W.J., Chen, W.H., Tang, C.Y.: Synchronizable for multi-party protocol conformance testing. Computer Communications **21** (1998) 1177–1183

15. Cacciari, L., Rafiq, O.: Controllability and observability in distributed testing. Inform. Software Technol. **41** (1999) 767–780
16. Boyd, S., Ural, H.: The synchronization problem in protocol testing and its complexity. Information Processing Letters **40** (1991) 131–136
17. Dssouli, R., von Bochmann, G.: Error detection with multiple observers. In: Protocol Specification, Testing and Verification. Volume V., Elsevier Science (North Holland) (1985) 483–494
18. Dssouli, R., von Bochmann, G.: Conformance testing with multiple observers. In: Protocol Specification, Testing and Verification. Volume VI., Elsevier Science (North Holland) (1986) 217–229
19. Rafiq, O., Cacciari, L.: Coordination algorithm for distributed testing. The Journal of Supercomputing **24** (2003) 203–211
20. Hierons, R.M., Ural, H.: Uio sequence based checking sequence for distributed test architectures. Information and Software Technology **45** (2003) 798–803
21. Chen, J., abd H. Ural, R.M.H.: Overcoming observability problems in distributed test architectures. (Information Processing Letters) to appear.
22. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. 1 (1972) 146–160
23. Eswaran, K.P., Tarjan, R.E.: Augmentation problems. SIAM J. Comput. **5** (1976) 653–665