Noname manuscript No.
(will be inserted by the editor)

Overcoming Controllability Problems in Distributed Testing from an Input Output Transition System

Robert M. Hierons

Received: date / Accepted: date

Abstract This paper concerns the testing of a system with physically distributed interfaces, called ports, at which it interacts with its environment. We place a tester at each port and the tester at port p observes events at p only. This can lead to controllability problems, where the observations made by the tester at a port p are not sufficient for it to be able to know when to send an input. It is known that there are test objectives, such as executing a particular transition, that cannot be achieved if we restrict attention to test cases that have no controllability problems. This has led to interest in schemes where the testers at the individual ports send coordination messages to one another through an external communications network in order to overcome controllability problems. However, such approaches have largely been studied in the context of testing from a deterministic finite state machine. This paper investigates the use of coordination messages to overcome controllability problems when testing from an input output transition system and gives an algorithm for introducing sufficient messages. It also proves that the problem of minimising the number of coordination messages used is NP-hard.

Keywords Distributed testing; controllability problems; coordination messages; input output transition system.

Robert M. Hierons
Department of Information Systems and Computing,
Brunel University,
Uxbridge, Middlesex, UB8 3PH United Kingdom
E-mail: rob.hierons@brunel.ac.uk

1 Introduction

It is widely accepted that testing is a crucial part of the software development process but also that manual testing is typically expensive and error prone. This has led to significant interest in approaches to test automation including model-based testing (MBT) in which testing is based on a model of the system under test (SUT) or the aspect of the SUT that is being tested [2,6,12–14, 18].

In distributed testing there is a tester at each port (interface) of the SUT and each tester observes only the events at its port. It is known that this introduces controllability and observability problems. Controllability problems occur when a tester cannot know when to apply an input. Consider, for example, the test case illustrated in Figure 1 in which the vertical lines represent time, which progresses as we move down, and there are two ports U and L. Here the tester at U starts the test case by supplying input $?i_U$, the response of the SUT should be the sending of $!o_U$ to U and the tester at L should then send input $?i_L$. The problem here is that the tester at L does not observe the previous input and output and so cannot know when to send $?i_L$.

An observability problem occurs if the global trace (sequence of inputs and outputs) is not one contained in the specification but is indistinguishable from a global trace in the specification as a result of the testers observing only the local projections (local traces). Let us suppose, for example, that the specified response to input sequence $?i_U?i_U$ is $!o_U$ at U and $!o_L$ at L in response to the first input and just $!o_U$ (at U) in response to the second input. Then the tester at U expects to observe $?i_U!o_U?i_U!o_U$ and the tester at L expects to observe $!o_L$. This is still the case if the response to $?i_U$ is $!o_U$

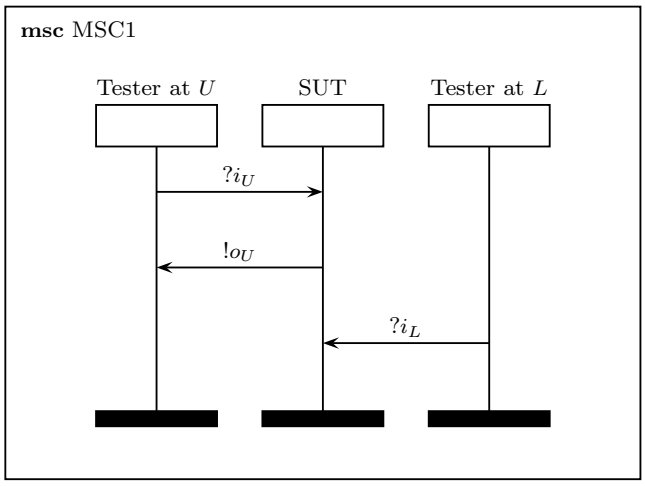

Fig. 1 A controllability problem

at U only and the response to the second input is $!o_U$ at U and $!o_L$ at L.

Controllability problems lead to the situation in which we do not know whether the intended input sequence was received by the SUT. In contrast, observability problems can lead to fault masking: a specified global trace did not occur but the set of local observations is consistent with a specified global trace.

There has been significant interest in controllability and observability problems in distributed testing when testing from a deterministic finite state machine (DFSM) [4,5,7,10,11,21,28,30,31,35,36,38,39]. There has been work that aims to produce test sequences that achieve particular objectives, such as executing a given transition, and that do not have controllability problems and/or observability problems. However, it is known that for a given DFSM and objective, there may be no test case that achieves the objective and does not have controllability problems. As a result, a number of approaches that overcome controllability problems have been devised and these rely on the sending of *coordination messages* between the testers [5,7,16,28,31].

While there has been interest in controllability and observability problems for DFSMs, distributed systems are often non-deterministic and so the restriction to deterministic models is a significant limitation. In addition, in FSMs we have that inputs and outputs alternate. While FSMs are highly suitable for some classes of system, sometimes we need more general models such as input output transition systems (IOTSs). As a result, there has been recent interest in distributed testing from an IOTS [20,19] but, while controllability has been examined in this context [19], very little previous work has explored methods for overcoming controllability problems.

It is important to note that the work on distributed testing from an IOTS has defined new implementation relations that require the distributed observations made regarding the SUT to be consistent with the specification [20,19]. These implementation relations remove the concern regarding observability problems: if the global trace σ occurred in testing and we cannot distinguish between this and an allowed global trace in the specification then we consider σ to be acceptable since users would also not be able to distinguish between σ and an allowed behaviour. Interestingly, this observation has also been made in the context of refinement in CSP [22]. There is also work in which a transition can be triggered by multiple inputs [15,3] but the work in this context has assumed that global observations are made.

Some previous work has used coordination messages to overcome controllability problems in distributed testing from an IOTS. An approach has been devised for distributing a global test case to produce a set of local test cases whose testing is controllable [23]. This work uses an agent, called an *Election Service*, that controls the testing process. However, communications between the testers and the Election Service is synchronous and each event in testing involves the sending and receiving of multiple coordination messages between the testers and the Election Service; instead we would like a scheme that requires relatively few such messages. There are at least two reasons for this. The first is that the requirement to send many coordination messages may lead to the need for a faster and more expensive external network. In addition, if the testers communicate through a network shared with the SUT then this communication can change the behaviour of the SUT. While it has been observed that the scalability issues can be reduced by using a tree structure for connecting distributed testers [9], it is still desirable to use relatively few coordination messages.

An alternative approach is to synchronise the testers through message exchange. This can solve the controllability problems and is similar to the approach given in [23]. Where it is feasible to use mechanisms to synchronise the testers, this approach has the benefit of simplicity and also potentially allows the testers to observe the global trace that occurred. However, similar to the discussion above, this approach may require many messages to be sent between the testers and there are scenarios, such as the testers and the SUT sharing a communications network, in which this is problematic. Note that in this paper we do not consider test cases that contain timing requirements. Since coordination messages introduce (potentially unpredictable) latency, it seems likely that when there are timing constraints we will wish to limit the number of coordination mes-

sages used but also that we will need schemes that take into account the particular timing requirements that are present.

Some related issues have been explored in the context of Message Sequence Charts (MSCs). Specifically, the notion of a controllability problem in testing is similar to those of race and non-local choice in MSCs. An MSC contains a race if the order of certain events is specified in the MSC but this order cannot be ensured [1]. The local choice assumption is that each process only observes the events it is involved in and so a non-local choice occurs when an MSC requires a process to behave in a manner that breaks the local choice assumption [1]. The problem of adding messages to overcome such problems has been explored [29]. However, this previous work only considered a single basic MSC[1]. In addition, in testing we have a restriction: we can only add messages between testers. In contrast, the work on adding messages to MSCs allowed messages to be added between any two processes. As a result, approaches devised for MSCs cannot be applied.

This paper makes the following contributions. First, it discusses the use of coordination messages and how these can be added to test cases for use in testing from an IOTS. It transpires that there are choices to be made here and particularly whether one includes both the sending and receiving of a coordination message in a test case. It also gives an algorithm for deciding whether a global test case is controllable and for characterising the set of controllability problems. This algorithm requires the test case to have a finite set of states but does not require it to allow only finitely many traces. It gives an algorithm for adding coordination messages to a global test case in order to overcome controllability problems. This second algorithm requires that there are only finitely many controllability problems. Both algorithms have polynomial time complexity. Finally, it shows that the problem of finding a minimal set of coordination messages to overcome controllability problems is NP-hard, even if we restrict attention to test cases that are in the form of trees.

In this paper we make several assumptions. First, we do not consider IOTSs that contain infinite paths with only outputs and internal actions. This is a little like outlawing live-locks. We also restrict attention to specifications and implementations that are input-enabled[2]; this simplifies the exposition but it should be straightforward to remove this restriction since we analyse test cases and not implementations or specifications. Importantly, we have to make assumptions regarding the nature of the communications between the testers since different options/assumptions lead to slightly different methods. First, we assume that quiescence[3] can be observed locally in testing and that the observation of quiescence is preceded by the arrival of all coordination messages previously sent. Since quiescence is usually observed through there being no events for a sufficiently long time, this assumption requires us to have information, such as upper bounds, regarding the time it takes a coordination message to reach its destination. However, we show how this restriction can be removed. In addition, we assume that if a global test case contains a coordination message from p to q before an input $?i_q$ at q then the tester at q waits to receive this coordination message before sending $?i_q$. We make this assumption because the only reason for a tester to send a coordination message to another tester is to provide them with information that helps them to determine when to apply an input. Again, we explain how the method can be changed to remove this assumption. Finally, we assume that communications between the testers is asynchronous but that between a tester and the SUT is synchronous. This corresponds to the situation in which the SUT has physically distributed interfaces at which it interacts with its environment but that at each interface the tester interacts directly with the SUT. If communications between testers and the SUT is asynchronous then we can apply the standard approach in which we see testing as synchronous communications between the tester(s) and a system that includes the communications channels [23,34].

The paper is structured as follows. Section 2 defines the notation used, multi-port IOTSs and global test cases. Section 3 discusses controllable testing and Section 4 gives an algorithm for determining whether a global test case is controllable. Section 5 describes coordination messages and introduces notation for adding these to global test cases. Section 6 shows how we can analyse traces of a test case that has had coordination messages added. Section 7 then gives an algorithm that adds coordination messages to a global test case in order to overcome controllability problems. Section 8 proves that the problem of finding a smallest set of coordination messages to overcome controllability problems is NP-hard, even if we restrict attention to test cases that are in the form of trees. Finally, Section 9 draws conclusions and discusses possible future work.

[1] A basic MSC defines one scenario.

[2] A process is input enabled if for every input $?i$ and state q of the process there is a transition from q with label $?i$.

[3] A system is quiescent if it cannot produce output without first receiving input and this situation is usually observed in practice through the use of timers.

2 Preliminaries

2.1 Notation

In this paper we will let I denote the set of inputs that the SUT can receive and O denote the set of outputs that it can produce. Typically, we will precede the name of an input with ? and precede the name of an output with !. We will call a sequence of inputs and outputs a *trace* and given a trace σ, pref(σ) will denote the set of prefixes of σ. Thus, pref(σ) = $\{\sigma' | \exists \sigma''.\sigma = \sigma'\sigma''\}$.

A relation r on a set A is a subset of $A \times A$. Relation r on set A defines a directed graph (digraph) $G(r)$; each element of A is represented by a vertex and there is an edge from the vertex representing $a_1 \in A$ to the vertex representing $a_2 \in A$ if and only if we have that $(a_1, a_2) \in r$. Relation r is a (strict) *partial order* if it is irreflexive, antisymmetric, and transitive and then (A, r) is a *partially ordered set (poset)*. Clearly the transitive closure of an irreflexive relation r is a partial order if and only if $G(r)$ is acyclic. A sequence $\sigma = a_1 \dots a_n$ is a *linearisation* of poset (A, r), $|A| = n$, if $A = \{a_1, \dots, a_n\}$ and for all $1 \leq i, j \leq n$ we have that if $(a_i, a_j) \in r$ then $i < j$. Thus, a sequence is a linearisation of $(A, <)$ if it is a permutation of the elements of A and it is consistent with the partial order.

2.2 Input Output Transition Systems

An input output transition system (*IOTS*) is a model in which there are states and transitions between the states. Each transition has a label, which is either an input, an output or τ (to denote an unobservable event). An IOTS is essentially a labelled transition system in which we differentiate between input and output. For the purpose of testing it is important to distinguish between input and output since the environment controls the input while the SUT controls the output. Usually it is assumed that the environment does not block output from the SUT and the SUT does not block input [33] and so we also make this assumption.

Definition 1 An *input-output transition system* s is defined by (Q, I, O, T, q_0) in which Q is a countable set of states, $q_0 \in Q$ is the initial state, I is a countable set of inputs, O is a countable set of outputs, and $T \subseteq Q \times (I \cup O \cup \{\tau\}) \times Q$ is the transition relation. If there is a transition $(q, a, q') \in T$ then it is possible to move from state q to state q' with action $a \in I \cup O \cup \{\tau\}$. We let $\mathcal{IOTS}(I, O)$ denote the set of IOTSs with input set I and output set O.

State $q \in Q$ is *quiescent* if from q it is not possible to produce output without first receiving input. We can extend T to T_δ by adding (q, δ, q) for each quiescent state q. We let $\mathcal{A}ct = I \cup O \cup \{\delta\}$ denote the set of *observable* actions and so $\tau \notin \mathcal{A}ct$. We say that process s is *input-enabled* if for all $q \in Q$ and $?i \in I$ there exists $q' \in Q$ such that $(q, ?i, q') \in T$. s is *output-divergent* if it can reach a state from which there is an infinite path that contains outputs and internal actions only.

In this paper we assume that processes are not output-divergent. In addition, specifications and implementations are assumed to be input-enabled but test cases need not be. When testing from an IOTS it is normal to assume that quiescence is observable [33] and we make this assumption. In practice quiescence is observed through a timeout: it is assumed that if the SUT produces no output for a given predefined time then it is in a quiescent state. The approach described in this paper also works in cases where quiescence is not considered to be observable.

The following notation is typically used when discussing testing from an IOTS.

Definition 2 Let $s = (Q, I, O, T, q_0)$ be an IOTS.

1. If $(q, a, q') \in T_\delta$, for $a \in \mathcal{A}ct \cup \{\tau\}$, then we write $q \xrightarrow{a} q'$.
2. We write $q \stackrel{\epsilon}{\Longrightarrow} q'$ if there exist states $q_1, \dots, q_m \in Q$, for $m \geq 1$, such that $q = q_1$, $q' = q_m$, $q_1 \xrightarrow{\tau} q_2, \dots, q_{m-1} \xrightarrow{\tau} q_m$.
3. We write $q \stackrel{a}{\Longrightarrow} q'$, for $a \in \mathcal{A}ct$, if there exist states $q_1, q_2 \in Q$ such that $q \stackrel{\epsilon}{\Longrightarrow} q_1$, $q_1 \xrightarrow{a} q_2$, and $q_2 \stackrel{\epsilon}{\Longrightarrow} q'$.
4. We write $q \stackrel{\sigma}{\Longrightarrow} q'$ for $\sigma = a_1 \dots a_m \in \mathcal{A}ct^*$ if there exist states $q_0, \dots, q_m \in Q$, $q = q_0$, $q' = q_m$ such that for all $0 \leq i < m$ we have that $q_i \stackrel{a_{i+1}}{\Longrightarrow} q_{i+1}$.
5. We write $s \stackrel{\sigma}{\Longrightarrow}$ if there exists a state $q' \in Q$ such that $q_0 \stackrel{\sigma}{\Longrightarrow} q'$ and we say that σ is a *trace* of s. We let $\mathcal{T}r(s)$ denote the set of traces of s.

Process s is *deterministic* if for all $\sigma \in \mathcal{A}ct^*$, we have that there is at most one output $!o$ such that $\sigma!o$ is a trace of s.

The elements of $\mathcal{T}r(s)$ are often called *suspension traces* since they contain quiescence. However, we simply call them traces since they are the only type of trace that we consider.

In this paper we will use some results regarding finite automata. A finite automaton is an IOTS in which there is a finite set of states, we do not differentiate between input and output, alphabet X is finite, and there is a set of final states.

Definition 3 A finite automaton (FA) A is defined by a tuple (Q, q_0, X, h, F) in which Q is a finite set of

states, $q_0 \in Q$ is the initial state, X is the finite alphabet, h is the transition relation and F is the set of final states. The transition relation h has type $Q \times (X \cup \{\tau\}) \times Q$, where τ represents empty (internal) moves that do not have associated observations.

The FA $A = (Q, q_0, X, h, F)$ defines the language $L(A)$ of traces that can take A from q_0 to a final state in F. Note that the sequences in $L(A)$ do not contain instances of τ since these do not appear in traces. While we will not use FA to define processes, test cases will have a finite number of states and so can be seen as FA. This will allow us to use standard methods and results for FA.

2.3 Multi-port Input Output Transition Systems

The notion of an IOTS has been extended to the situation in which there are multiple interfaces, called ports. In this paper we consider the case where there are $n > 1$ ports and let $\mathcal{P} = \{1, \ldots, n\}$ denote the set of names of the ports. We assume that the sets of inputs and outputs can be partitioned into those that can be observed at the separate ports. Thus, we partition I into sets $I_1, \ldots, I_n$ of inputs, where for $p \in \mathcal{P}$ we have that I_p denotes the set of inputs that can be received at p. Similarly, we partition O into sets $O_1, \ldots, O_n$. We assume that quiescence is observed locally: each tester observes quiescence. When testing single port systems quiescence is observed through using a timeout: it is assumed that there is a known upper bound on the time it takes for the SUT to produce output. For distributed testing it seems likely that each local tester will need such information but also additional information regarding the test case but the observation of quiescence is not a topic that we will consider in detail. The method given in this paper also works in situations where we do not observe quiescence. Note that if the same values can be sent as input or received as output at different ports then we can ensure that $I_1, \ldots, I_n$ and $O_1, \ldots, O_n$ partition I and O respectively by adding labels.

Some work has considered an alternative approach, in which the output can be a tuple (an output for each port) but restricting outputs to single values simplifies the analysis. We let $\mathcal{A}ct_p = I_p \cup O_p \cup \{\delta\}$, which is the set of observations that can be made at port p. We will typically label inputs and outputs to make their port clear. For example, $?i_1$ is an input at port 1 and $!o_2$ is an output at port 2.

The traces of an IOTS are sequences in $\mathcal{A}ct^*$ and a global tester can observe such traces. We call elements of $\mathcal{A}ct^*$ *global traces* in order to distinguish them from the traces observed at a single port. A *global test case* observes global traces. In contrast, a *local test case* contains a *local tester* at each port and the local tester at port p observes a *local trace* in $\mathcal{A}ct_p^*$. It is straightforward to construct a local trace from a global trace by simply removing all inputs and outputs observed at different ports [20]. In this paper, given a sequence σ of elements from a set A and a subset A' of A we will let $\sigma \downarrow_{A'}$ denote the sequence formed from σ by removing all elements not in A'. If Z contains a set of sequences then we let $Z \downarrow_{A'}$ denote the set of projections of these sequences: $Z \downarrow_{A'} = \{\sigma \downarrow_{A'} | \sigma \in Z\}$.

Given global traces $\sigma, \sigma' \in \mathcal{A}ct^*$ we write $\sigma \sim \sigma'$ if for all $p \in \mathcal{P}$ we have that $\sigma \downarrow_{\mathcal{A}ct_p} = \sigma' \downarrow_{\mathcal{A}ct_p}$. Here we have that if $\sigma \sim \sigma'$ then we cannot distinguish between σ and σ' when only local traces are observed.

2.4 Test cases for distributed testing

Let us suppose that we are testing from a specification $s \in \mathcal{IOTS}(I, O)$. A *global test case* t is a process that interacts with the SUT by synchronising on common actions. As usual, we also restrict the alphabet of a global test case to be that of the SUT, and so $t \in \mathcal{IOTS}(I, O \cup \{\delta\})$. We also assume that global test cases have a finite number of states and cannot block output by the SUT: until testing terminates it must always be able to react to any possible output. In addition, we make the normal assumption that a test case is deterministic as defined below; test cases are usually deterministic and this restriction does not eliminate the possibility of adaptive testing [17,26].

Definition 4 Given $s \in \mathcal{IOTS}(I, O)$, a *global test case* is a process $t \in \mathcal{IOTS}(I, O \cup \{\delta\})$ that has a finite number of states some of which represent the termination of testing ($\perp$), has no transitions with label τ, and that satisfies the following properties.

1. If $t \stackrel{\sigma}{\Longrightarrow} t'$ for some $\sigma \in \mathcal{A}ct^*$ and t' that does not represent termination then for all $!o \in O \cup \{\delta\}$ we have that $t' \stackrel{!o}{\longrightarrow}$.
2. If $t \stackrel{\sigma}{\Longrightarrow} t'$ for some $\sigma \in \mathcal{A}ct^*$ then there is at most one $?i \in I$ such that $t' \stackrel{?i}{\longrightarrow}$.
3. If $t \stackrel{\sigma}{\Longrightarrow} t'$ and $t \stackrel{\sigma}{\Longrightarrow} t''$ for some $\sigma \in \mathcal{A}ct^*$ then $t' = t''$.

The notion of deterministic used here is a little different from that defined for IOTSs. This is because the input $?i$ corresponds to an output for a test case and outputs are received by the test case and not sent by the test case.

A global test case t for s is said to be *reduced* if for all $\sigma?i \in \mathcal{T}r(t)$ such that $?i \in I$ we have that $\sigma \in \mathcal{T}r(s)$.

This says that a reduced global test case will not supply an input after there has been a failure (a trace not in $\mathcal{T}r(s)$). Test cases are normally reduced and we will restrict attention to reduced test cases since it simplifies some of the notation.

A global test case t is said to be *tree-like* if for all $\sigma_1, \sigma_2 \in \mathcal{T}r(t)$ with $\sigma_1 \neq \sigma_2$, we have that $t \stackrel{\sigma_1}{\Longrightarrow} t_1$ and $t \stackrel{\sigma_2}{\Longrightarrow} t_2$ implies that $t_1 \neq t_2$.

When designing a test case to achieve a particular objective it is often desirable to consider global test cases since test objectives are often stated at this level. However, in distributed testing we actually use a *local test case* in which there is one *tester* at each port. The tester at port p is in $\mathcal{IOTS}(I_p, O_p \cup \{\delta\})$. Given a global test case t, we can produce a corresponding local test case by taking the projection of t at each port p [20]. In this paper we are concerned with producing global test cases that are controllable and so will not have to consider local test cases.

Next we define the parallel composition of a system and a global test case.

Definition 5 Given $s \in \mathcal{IOTS}(I, O)$ and a global test case t for s we define $s||t \in \mathcal{IOTS}(I, O)$ to be the application of global test case t to s. The system $s||t$ is formed from s and t by synchronising on actions belonging to $\mathcal{A}ct$. We can therefore define the behaviour of $s||t$ in the following way.

- If $s \stackrel{a}{\longrightarrow} s'$ and $t \stackrel{a}{\longrightarrow} t'$ for $a \in \mathcal{A}ct$ then $s||t \stackrel{a}{\longrightarrow} s'||t'$.
- If $s \stackrel{\tau}{\longrightarrow} s'$ then $s||t \stackrel{\tau}{\longrightarrow} s'||t$.

We let $\mathcal{T}r(s,t)$ denote the set of traces that can result from $s||t$.

This simply says that the synchronous composition of a process and a test case can only proceed through $a \in \mathcal{A}ct$ if both the test case and the process take transitions with label a. Naturally, the process s can always take a transition with label τ since such transitions are not observed by the test case. Given a global test case t for s we clearly have that $\mathcal{T}r(s,t) = \mathcal{T}r(s) \cap \mathcal{T}r(t)$.

The definition of $s||t$ corresponds to synchronous communications between the tester and the SUT. Naturally, in practice communications may be asynchronous. However, this need not be a significant limitation since the tester might be able to choose to wait for a given period of time before supplying an input or may be sent an acknowledgement of the message having been received. It has been observed that where this is not appropriate, asynchronous communications can be represented through adding models of the communications channels

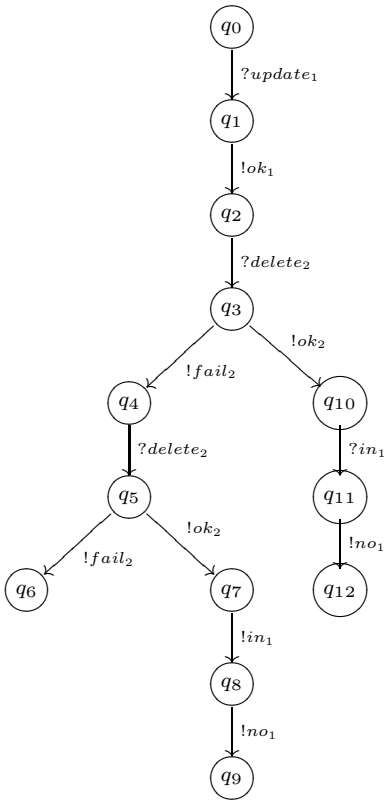

Fig. 2 Global Test Case t_0

[23]. This essentially corresponds to the testers interacting synchronously with a system composed of the SUT and the communications channels.

Consider, for example, the global test case shown in Figure 2, that will be called t_0 throughout this paper. Here there are two ports, 1 and 2, and the SUT is a system that allows data to be stored, accessed and deleted. The test case t_0 represents a scenario in which the tester at port 1 sends a message to add content to the SUT ($?update_1$) and waits for an acknowledgement ($!ok_1$). The tester at port 2 then attempts to delete this content ($?delete_2$). There is a possibility of this failing ($!fail_2$) or succeeding ($!ok_2$). The tester makes at most two attempts; if one leads to output $!ok_2$ then the tester at port 1 should send a message $?in_1$ to determine whether the item is in the SUT and should receive a response $!no_1$ that states that it is not. For each state q, that is not a leaf, and output $!o$ such that there is no transition from q with label $!o$, there is an implicit transition that takes t_0 to a state where it terminates and produces a fail verdict.

If we take the projections of t_0 we obtain the local testers shown in Figures 3 and 4 in which we have kept τ transitions. We can remove the transitions with label τ using standard algorithms that transform a finite au-

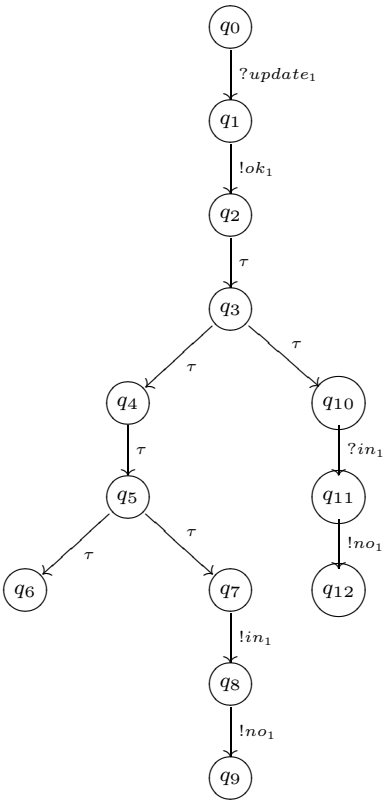

Fig. 3 Local Test Case for Port 1

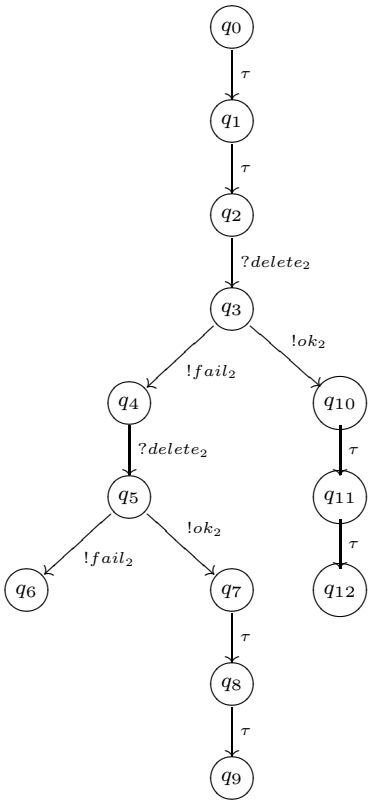

Fig. 4 Local Test Case for Port 2

tomaton with empty transitions to form one that has no empty transitions. This leads to the test cases in Figures 5 and 6.

We can use graph terminology when discussing global test cases. In particular, we will talk about a *path* ρ of a global test case t: this is a sequence of consecutive transitions of t starting at its initial state. A path ρ has a *label*: the sequence of observable events on ρ. For example, t_0 has the path $\rho = (q_0, !update_1, q_1)(q_1, !ok_1, q_2)$ that has label $!update_1!ok_1$.

3 Controllable testing

In distributed testing a tester at port $p \in \mathcal{P}$ observes only the local trace that occurs at p. As a result, the tester at p can only use the observations it has made in deciding when to supply an input. Traditionally, a controllability problem occurs in a test case when testing can lead to a situation in which the tester at a port p does not know whether to supply an input. If such controllability problems occur then there are races in the set of possible interactions between the test case and the specification and the wrong input may be supplied during testing. Ideally we only use test cases that have no such controllability problems and this has motivated much of the work in the area of distributed testing [4, 5, 7, 11, 19, 28, 30, 31, 35, 38, 39].

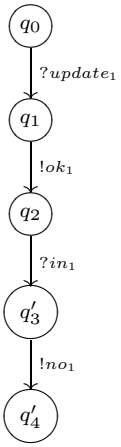

Fig. 5 Local Test Case for Port 1 without τ transitions

It is straightforward to see that t_0 is not controllable since, for example, the tester at port 2 cannot know when to send input $?delete_2$. Naturally, if there is the

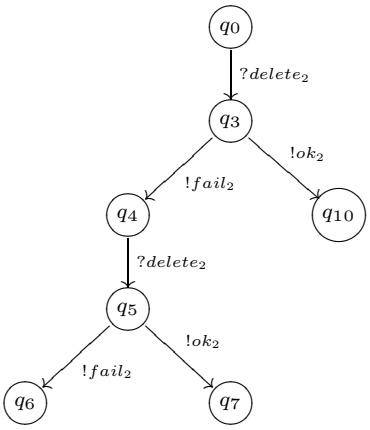

Fig. 6 Local Test Case for Port 2 without τ transitions

possibility that $?delete_2$ arrives before $?update_1$ then it is also possible that the SUT behaves as required and yet the output in response to $?in_1$ is different from that expected: this happens if we try to delete the element before adding it. As a result, a correct implementation might produce an unexpected output.

When testing from a deterministic FSM (DFSM) a test case is typically an input sequence and this defines a global trace. Since there is only one possible global trace for a given input sequence, it is straightforward to give a condition under which an input sequence is controllable. Specifically, if input sequence $x_1, \ldots, x_k$ leads to trace $x_1/y_1, \ldots, x_k/y_k$ then this input sequence is controllable if and only if for all $1 < i \leq k$ we have that the input x_i is at a port p such that the previous input/output pair x_{i-1}/y_{i-1} contains at least one observation (input or output) at port p. The idea here is that if the tester at p observes either input or output in the previous transition (x_{i-1}/y_{i-1}) then it simply sends input x_i after observing this; otherwise it cannot know when to send its input. Notice here that in an FSM there is an atomicity assumption regarding an input/output pair: if the FSM responds to input x with output y then it is not possible to apply another input after x and before the FSM sends output y.

When testing from an IOTS we need a rather different definition of what it means for a test case to be controllable since input and output need not alternate, a process can be non-deterministic, and test cases need not be input sequences. However, the intuition is similar: we need each tester to make observations that allow it to decide when to apply an input. The following defines what it means for a global test case to be controllable [19].

Definition 6 A global test case t is *controllable* for IOTS s if there does not exist port $p \in \mathcal{P}$, $\sigma_1, \sigma_2 \in \mathcal{T}r(s,t)$ and $?i_p \in I_p$ with $\sigma_1 ?i_p \in \mathcal{T}r(s,t)$, $\sigma_2 ?i_p \notin \mathcal{T}r(s,t)$ and $\sigma_1 \downarrow_{\mathcal{A}ct_p} = \sigma_2 \downarrow_{\mathcal{A}ct_p}$.

An alternative way of defining this is to say that for all $\sigma_1, \sigma_2 \in \mathcal{T}r(s,t)$ with the same projections at port p, if there is an input $?i_p \in I_p$ such that $\sigma_1 ?i_p \in \mathcal{T}r(s,t)$ then we must have that $\sigma_2 ?i_p \in \mathcal{T}r(s,t)$.

The definition has to mention the specification since, for any test case t that includes inputs at more than one port, there is some behaviour (global trace) that will lead to a race between inputs at two ports[4].

Proposition 1 below, which was previously proved [19], tells us that if a test case is controllable then each input is supplied by a local tester at the point specified in the test case. One slight caveat is that, as explained above, this need only be the case when the behaviour that occurs is consistent with the specification.

Proposition 1 *Let us suppose that we are testing $i \in \mathcal{IOTS}(I,O)$ with local test cases produced from a global test case t that is controllable for $s \in \mathcal{IOTS}(I,O)$. If an input $?i$ is supplied after $\sigma \in \mathcal{T}r(s,t)$ then $\sigma ?i \in \mathcal{T}r(t)$.*

4 Deciding whether a test case is controllable

In this section we show how we can decide whether a global test case t, that does not contain coordination messages, is controllable. We also show how, if t is not controllable, we can characterise the controllability problems.

Recall that a global test case t is an IOTS and so can define an infinite set of traces. As a result, we need to examine sets of traces rather than individual traces. Let us suppose that $?i_p \in I_p$. We will show how one can decide whether there are circumstances in which the input of $?i_p$ can cause a controllability problem in t. In order to do this, we will define the following languages.

1. $L_T(?i_p, t)$: the set of traces of t that can be followed by $?i_p$ in t.
2. $L_F(?i_p, t)$: the set of traces of t that cannot be followed by $?i_p$ in t.

It is straightforward to define finite automata with these languages by defining appropriate sets of final states for t. For $L_T(?i_p, t)$ we simply take t and make the final states be those from which there are transitions

[4] To see that this must be the case, let us suppose that t can send input $?i_p$ at $p \in \mathcal{P}$ after the local trace σ_p is observed at p and t can send input $?i_q$ at q after the local trace σ_q is observed at port q $(q \neq p)$. A race occurs if the SUT produces any trace σ such that $\sigma \downarrow_{\mathcal{A}ct_p} = \sigma_p$ and $\sigma \downarrow_{\mathcal{A}ct_q} = \sigma_q$.

with label $?i_p$ and let $t_T(?i_p)$ denote this FA. In contrast, for $L_F(?i_p, t)$ we take t and make the final states be those from which there are no transitions with label $?i_p$ and let $t_F(?i_p)$ denote this FA. This approach works because we require test cases to be deterministic: for a trace $\sigma \in \mathcal{T}r(t)$ there cannot be more than one path of t that has label σ.

We can now take the projections at p of these languages $L_T(?i_p, t)$ and $L_F(?i_p, t)$ to form $L^p_T(?i_p, t)$ and $L^p_F(?i_p, t)$ respectively. We do this by making every transition whose label is not in $\mathcal{A}ct_p$ have an empty label τ. Let the associated FA be $t^p_T(?i_p)$ and $t^p_F(?i_p)$ respectively. Finally, there is a controllability problem associated with $?i_p$ if and only if the intersection of $L^p_T(?i_p, t)$ and $L^p_F(?i_p, t)$ is non-empty. We can decide this by taking the product automaton $P(t^p_T(?i_p), t^p_F(?i_p))$ of $t^p_T(?i_p)$ and $t^p_F(?i_p)$ defined as follows.

Definition 7 Let $A = (Q_1, q_{01}, X, h_1, F_1)$ and $B = (Q_2, q_{02}, X, h_2, F_2)$ be finite automata with the same alphabets. Then the product automaton $P(A, B)$ is the finite automaton $(Q_1 \times Q_2, (q_{01}, q_{02}), X, h, F_1 \times F_2)$ in which h is defined by the following.

1. If $(q_1, a, q'_1) \in h_1$ and $(q_2, a, q'_2) \in h_2$ then we have that $((q_1, q_2), a, (q'_1, q'_2)) \in h$.
2. If $(q_1, \tau, q'_1) \in h_1$ then $((q_1, q_2), \tau, (q'_1, q_2)) \in h$ for all $q_2 \in Q_2$.
3. If $(q_2, \tau, q'_2) \in h_2$ then $((q_1, q_2), \tau, (q_1, q'_2)) \in h$ for all $q_1 \in Q_1$.

The intersection of $L^p_T(?i_p, t)$ and $L^p_F(?i_p, t)$ is the language defined by the FA $P(t^p_T(?i_p), t^p_F(?i_p))$. The approach is summarised in Algorithm 1. If there are controllability problems then the algorithm returns $False$ and also the set C that contains tuples of the form $(L, ?i_p)$ in which L contains the set of projections, at p, of traces in $\mathcal{T}r(t)$ that are associated with controllability problems. Specifically, L contains the set of $\sigma_p \in \mathcal{A}ct^*_p$ where there exist $\sigma_1, \sigma_2 \in \mathcal{T}r(t)$ such that $\sigma_p = \sigma_1 \downarrow_{\mathcal{A}ct_p} = \sigma_2 \downarrow_{\mathcal{A}ct_p}$, in t we have that σ_1 can be followed by $?i_p$ and in t we cannot follow σ_2 by $?i_p$.

It is clear that $t^p_T(?i_p)$ and $t^p_F(?i_p)$ can be produced in linear time. In addition, $P(t^p_T(?i_p), t^p_F(?i_p))$ can be produced in time that is quadratic in terms of the size of t. We can decide whether $P(t^p_T(?i_p), t^p_F(?i_p))$ defines the empty language by applying a depth-first search to determine whether any of the final state are reachable. This depth-first search operates in time that is linear in terms of the size of $P(t^p_T(?i_p), t^p_F(?i_p))$ [32] and so quadratic in the size of t. Thus, Algorithm 1 operates in time that is quadratic in the size of t and linear in the size of I.

Given the set C, we can now define the set of controllability problems caused by t: the set of $(\sigma_1, \sigma_2, ?i_p)$, $?i_p \in I_p$, where $\sigma_1 \downarrow_{\mathcal{A}ct_p} = \sigma_2 \downarrow_{\mathcal{A}ct_p}$, in t we have that σ_1 can be followed by $?i_p$ and in t we cannot follow σ_2 by $?i_p$. We will call this $Ctr(t)$ and we now show how $Ctr(t)$ can be produced if it is finite; naturally, $Ctr(t)$ is guaranteed to be finite in the important case where $\mathcal{T}r(t)$ is finite.

Algorithm 1 Finding controllability problems

```
Input global test case t.
Output whether t is controllable and a characterisation of
any controllability problems.
Let ans = True.
Let C = ∅
for all p ∈ P and ?i_p ∈ I_p do
   Produce the FA t^p_T(?i_p) and t^p_F(?i_p).
   Produce P(t^p_T(?i_p), t^p_F(?i_p)).
   if The language L = L(P(t^p_T(?i_p), t^p_F(?i_p))) is non-empty
   then
      ans = False.
      C = C ∪ {(L, ?i_p)}.
   end if
end for
Output ans and C
```

Let us suppose that Algorithm 1 returns a non-empty set C and that $(L, ?i_p)$ is an element of C for finite L. We will show how we can produce the corresponding elements of $Ctr(t)$. Let $\sigma_p = a_1 \dots a_k$ be an element of L. Then we generate the following sets in which $L(\sigma_p)$ is the language $(\mathcal{A}ct \setminus \mathcal{A}ct_p)^*\{a_1\}(\mathcal{A}ct \setminus \mathcal{A}ct_p)^*\{a_2\} \dots \{a_k\}(\mathcal{A}ct \setminus \mathcal{A}ct_p)^*$ of sequences in $\mathcal{A}ct^*$ whose projection at p is σ_p.

1. $L_1(t, \sigma_p, ?i_p) = L_T(?i_p, t) \cap L(\sigma_p)$: the set of traces of t that can be followed by $?i_p$ and whose projection at p is σ_p.
2. $L_2(t, \sigma_p, ?i_p) = L_F(?i_p, t) \cap L(\sigma_p)$: the set of traces of t that cannot be followed by $?i_p$ and whose projection at p is σ_p.

Then we simply add to Ctr the set of $(\sigma_1, \sigma_2, ?i_p)$ such that $\sigma_1 \in L_1(t, \sigma_p, ?i_p)$ and $\sigma_2 \in L_2(t, \sigma_p, ?i_p)$. Clearly, this process takes time that is polynomial in the sizes of t and the resultant set of controllability problems (the sum of the sizes of the traces in tuples in $Ctr(t)$).

In Section 7 we will show how controllability problems in $Ctr(t)$ can be overcome if $Ctr(t)$ is finite. The algorithm will be iterative: in each iteration it will add a coordination message to overcome a controllability problem in $Ctr(t)$ and then remove from $Ctr(t)$ all controllability problems resolved by this. In order to achieve this last step, we will need to be able to decide whether an element of $Ctr(t)$ is still a controllability problem even though coordination messages have already been added. Thus, before we give the algorithm in Section 7 we show how coordination messages can be

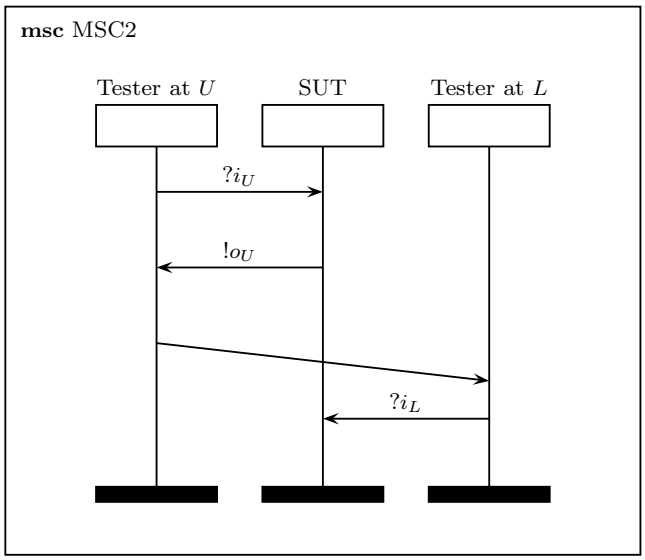

Fig. 7 Resolving a controllability problem

added to a test case in Section 5 and then, in Section 6, we show how we can decide whether an element of $Ctr(t)$ has been resolved by the coordination messages that have been added to t.

5 Adding coordination messages

5.1 Introduction

Work on testing from a DFSM has shown how coordination messages, sent between the testers, can be used to overcome controllability problems [5,16,36,35]. Consider, for example, the situation shown in Figure 1. Here there is a controllability problem because the tester at L does not know when to send input $?i_L$. In order to overcome this it is sufficient for the tester at U to send a message to the tester at L after it has observed output $!o_U$. This is shown in Figure 7 in which the coordination message from U to L is not given a label.

It is straightforward to generalise this approach to overcome controllability problems in any test sequence for a DFSM. This can be achieved in the following way when applying an input sequence $x_1, \ldots, x_k$ such that the specification contains the trace $x_1/y_1, \ldots, x_k/y_k$: for all $1 \leq i < k$, if the input x_{i+1} is at a port $p \in \mathcal{P}$ such that $x_i y_i \downarrow_{\mathcal{A}ct_p} = \epsilon$ then we determine which port q is such that $x_i \in X_q$ and have the tester at q send a coordination message to the tester at p after it sends x_i. Then, the tester at p knows to send x_{i+1} once it receives this coordination message.

As explained in Section 2, the conditions for a test case being controllable are quite different when testing against an IOTS. In addition, typically a test case will not be a single sequence. In this section we first introduce notation for coordination messages and for adding these to a global test case.

5.2 Coordination messages

Coordination messages are sent between testers in order to help overcome controllability problems. In previous work on testing from a DFSM, a coordination message contains no additional information: when a tester observes a coordination message all they know is the identity of the tester that sent this message (see, for example, [5]). In contrast, some previous work [23] allows these messages to include additional information.

Let us suppose that we are using coordination messages that contain no content and we wish to apply the test case t_1 shown in Figure 8 in which $p \neq q$. Here there are two sources of controllability problems:

1. the tester at p does not observe the events before it is meant to apply its inputs and so does not know when to send an input; and
2. the choice of input to be sent at p depends on the output produced at q but the tester at p cannot observe the output.

As a result of the first issue identified, in order to make t_1 controllable we have to add coordination messages from q to p after both $!o_q$ and $!o'_q$. However, in order to overcome the second issue the tester at p has to differentiate between these two cases and cannot do so on the basis of the coordination messages since they both come from q and contain no additional content. In addition, if q tries to overcome this by sending different numbers of messages after $!o_q$ and $!o'_q$, say n_1 and n_2, then there is a new controllability problem: after receiving $min\{n_1, n_2\}$ coordination messages the tester at p does not know whether to send an input or wait for additional coordination messages. Thus, we cannot use coordination messages, that have no additional content, to overcome the controllability problems in t_1. We therefore allow coordination messages to have labels.

5.3 Adding messages to test cases

In a global test case we will represent a coordination message with label l from tester p to tester q by $m_{pq}(l)$ and so a global tester contains instances of $m_{pq}(l)$. We assume that a tester will not send a coordination message to itself since such messages cannot help the testers to overcome controllability problems. We let $\mathcal{L}$ denote the set of labels and we let $\mathcal{M}$ denote the set of coordination messages and so $\mathcal{M} = \{m_{pq}(l) | p, q \in \mathcal{P} \wedge p \neq$

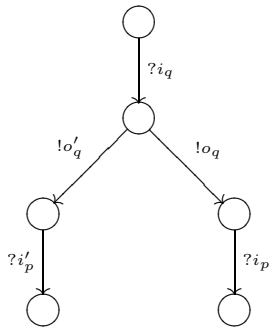

Fig. 8 Test case t_1

$q \wedge l \in \mathcal{L}\}$. We will let $\mathcal{M}_p$ denote the set of coordination messages that can be sent by the tester at p and so $\mathcal{M}_p = \{m_{pq}(l) | q \in \mathcal{P} \setminus \{p\} \wedge l \in \mathcal{L}\}$.

Definition 8 Given $s \in \mathcal{IOTS}(I, O)$ and set $\mathcal{M}$ of coordination messages, a global test case is a process $t \in \mathcal{IOTS}(I \cup \mathcal{M}, O \cup \{\delta\})$ that has a finite number of states, some of which represent termination ($\perp$), has no transitions with label τ, and that satisfies the following properties.

1. If $t \stackrel{\sigma}{\Longrightarrow} t'$ for $\sigma \in (\mathcal{A}ct \cup \mathcal{M})^*$ and t' does not represent termination then for all $!o \in O \cup \{\delta\}$ we have that $t' \stackrel{!o}{\longrightarrow}$.
2. If $t \stackrel{\sigma}{\Longrightarrow} t'$ for $\sigma \in (\mathcal{A}ct \cup \mathcal{M})^*$ then there is at most one $a \in I \cup \mathcal{M}$ such that $t' \stackrel{a}{\longrightarrow}$.
3. If $t \stackrel{\sigma}{\Longrightarrow} t'$ and $t \stackrel{\sigma}{\Longrightarrow} t''$ for $\sigma \in (\mathcal{A}ct \cup \mathcal{M})^*$ then $t' = t''$.

We assume that a label l used in a coordination message from the tester at p to the tester at q is not reused: for all $p, q \in \mathcal{P}$ and $l \in \mathcal{L}$, a global test case t does not have $\sigma \in \mathcal{T}r(t)$ that contains more than one instance of $m_{pq}(l)$. This assumption simplifies the analysis and fits with the algorithms we give for adding coordination messages to overcome controllability problems[5].

Let us suppose that global test case t does not contain coordination messages and that global test case t' does. We will say that t' *has been produced from* t *by adding coordination messages* if $\mathcal{T}r(t') \downarrow_{\mathcal{A}ct} = \mathcal{T}r(t)$.

In this paper we will show how coordination messages can be added to a global test case t in order to overcome controllability problems and we will achieve this by identifying (local) traces after which a particular coordination message must be sent. In order to represent the resultant global test case t', we might either transform t or produce a representation that has t and information regarding when coordination messages must be sent. The former produces a single global test case and might be achieved in the following way, when adding a coordination message m to t after a trace σ. First, we transform t so that there is an acyclic path with label σ from the initial state and no other transitions to the states before the end of this path; this can be achieved by copying states where necessary. Next, if the last transition of this path is (s, a, s') then we introduce a new state s'', remove (s, a, s'), and add transitions (s, a, s'') and (s'', m, s'). This can increase the number of states of t but this increase is bounded above by one plus the length of σ. An alternative, which is likely to be more efficient, is to record when coordination messages must be sent: if coordination message m is to be sent by the tester at p after it has observed σ_p then we store this pair (σ_p, m) along with t. Here the resultant global test case t' is implicitly defined but it is straightforward, for example, to define $\mathcal{T}r(t')$. However, the choices regarding how to represent such a global test case t' is not an issue we will investigate further: we will only be concerned with deciding when to send coordination messages and will assume that $\mathcal{T}r(t')$ is defined.

Since communications between testers is asynchronous, the point at which a coordination message from p to q is sent does not define the point at which it arrives at q. We might therefore have included both the sending and arrival of coordination messages in a global test case. However, this approach has a practical problem: there may be many points in a global test case at which a coordination message could arrive and so if this arrival is explicitly represented then effectively we have to branch on when it arrives. This could lead to an exponential (in the number of coordination messages) increase in the size of the global test case. Thus, we only include the sending of a coordination message in a global test case.

As only the sending of a coordination message is explicitly represented as an event in a global test case, there is a need to specify the causality introduced by a coordination message. Let us suppose, for example, that in $\sigma \in \mathcal{T}r(t')$ there is a coordination message $m_{pq}(l)$ and later there is an input $?i_q$ at q. The question is: does the tester at q wait to receive the coordination message $m_{pq}(l)$ before sending $?i_q$. The tester at p sends a coordination message to the tester at q in order to provide information that will be used in order to decide when to send future input. Thus, in this paper we assume that in such cases the tester at q does wait for the coordination message to arrive; later we discuss how the

[5] It simplifies the analysis by eliminating the issue of whether there can be degeneracy, as described in work on MSCs [1]. Here there is degeneracy if two messages with label l sent from a process p to process q do not arrive in the order in which they were sent.

proposed approach can be adapted to work with different assumptions. As mentioned earlier, we also assume that whenever quiescence is observed all coordination messages that have been sent are received; this corresponds to the system formed from the SUT and its communications channels being quiescent.

Assumption 1 *In a global test case t, if $\sigma \in \mathcal{T}r(t)$ includes an event $a \in I_q \cup \mathcal{M}_q$ then the tester at q should not send a until all coordination messages previously sent to q in σ arrive.*

Assumption 2 *In testing, all coordination messages that have been sent are received before quiescence is observed.*

We now introduce notation for the events associated with a global trace that involves coordination messages. A coordination message leads to two events: the sending of the message and the arrival of the message. In a trace we will represent the sending of $m_{pq}(l)$ (by the tester at p) by event $e^s_{pq}(l)$ and we will represent the receiving of $m_{pq}(l)$ (by the tester at q) by event $e^r_{pq}(l)$. We will let $\mathcal{E}^s_p$ ($p \in \mathcal{P}$) denote the set of events that are the sending of a coordination message by the tester at p and so $\mathcal{E}^s_p = \{e^s_{pq}(l) | q \in \mathcal{P} \setminus \{p\} \wedge l \in \mathcal{L}\}$. We will also let $\mathcal{E}^r_p$ denote the set of events that involve the tester at p receiving a coordination message and so $\mathcal{E}^r_p = \{e^r_{qp}(l) | q \in \mathcal{P} \setminus \{p\} \wedge l \in \mathcal{L}\}$. Similarly, for $p \in \mathcal{P}$ we let $\mathcal{E}_p = \mathcal{E}^s_p \cup \mathcal{E}^r_p$ and we also let $\mathcal{E}^s$ denote the union, over $p \in \mathcal{P}$, of the $\mathcal{E}^s_p$ and $\mathcal{E}^r$ denote the union, over $p \in \mathcal{P}$, of the $\mathcal{E}^r_p$. Finally, we let $\mathcal{E} = \mathcal{E}^s \cup \mathcal{E}^r$.

We now explore properties of global test cases that contain coordination messages and show how we can decide whether controllability problems have been resolve by the addition of these coordination messages.

6 Examining traces that have coordination messages

In Section 7 we will give an algorithm for adding coordination messages in order to overcome the controllability problems in a global test case t that contains a finite number of such controllability problems. The algorithm will be iterative: in each iteration a coordination message will be added in order to overcome a particular controllability problem in $Ctr(t)$. The algorithm will then examine the remaining elements in $Ctr(t)$ in order to determine whether they have also been resolved. In this section we therefore show how we can decide, given a global test case t and $(\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$, whether the tuple $(\sigma_1, \sigma_2, ?i_p)$ still corresponds to a controllability problem in a global test case t' formed from t by adding coordination messages.

Given the trace σ of a global test case t', that possibly contains coordination messages, σ defines a set of *events* such as the sending of an input or the reception of a coordination message. We let $E(\sigma)$ denote the events associated with σ, with labels added if an observation is repeated. A coordination message $m_{pq}(l)$ leads to two events: the sending of the message ($e^s_{pq}(l)$) and the arrival of the message ($e^r_{pq}(l)$). In contrast, each input, output or observation of quiescence in σ has one corresponding event in $E(\sigma)$. We let $E_p(\sigma)$ denote the set of events from $E(\sigma)$ that are observed at p: input at p, output or quiescence observed at p, the sending of a coordination message by the tester at p and the tester at p receiving a coordination message.

Given trace σ, there might be several alternative orders in which the events in $E(\sigma)$ can occur since, for example, a coordination message sent from port p to port q might arrive either before or after an output $!o_q$ is observed at q. We will show how we can represent these alternative events using a partial order on $E(\sigma)$. At port p the sequence of events, ignoring coordination messages, must be $\sigma \downarrow_{\mathcal{A}ct_p}$. In addition, our assumptions regarding coordination messages (Assumptions 1 and 2) introduce further constraints and we know that a coordination message is received after it is sent. We can bring together all of these constraints to define a (strict) partial order $<_\sigma$ on $E(\sigma)$ that defines the causalities between these events. In order to simplify the definition we let input, output or quiescence a_i in σ be represented by event a_i.

Definition 9 Given $\sigma = a_1, \ldots, a_k \in (\mathcal{A}ct \cup \mathcal{M})^*$, we let $<_\sigma$ be the transitive closure of the following relations on $E(\sigma)$.

1. Inputs, outputs, and quiescence observed at p must occur in the order specified in σ: if $a_i, a_j \in \mathcal{A}ct_p$ for some $p \in \mathcal{P}$, $i < j$, then we have that $a_i <_\sigma a_j$.
2. An input at q will not be sent until all coordination messages already sent to q have arrived (Assumption 1): if $a_i = m_{pq}(l)$ and $a_j \in I_q$ with $i < j$ then $e^r_{pq}(l) <_\sigma a_j$.
3. The tester at p will not send a coordination message until all earlier events at p have occurred and all coordination messages already sent to p have arrived (Assumption 1):
   (a) if $a_i \in \mathcal{A}ct_p$ and $a_j = m_{pq}(l)$ with $i < j$ then $a_i <_\sigma e^s_{pq}(l)$;
   (b) if $a_i = m_{p'p}(l')$ and $a_j = m_{pq}(l)$ with $i < j$ then $e^r_{p'p}(l') <_\sigma e^s_{pq}(l)$.
4. A coordination message is received after it is sent: if $a_i = m_{pq}(l)$ then $e^s_{pq}(l) <_\sigma e^r_{pq}(l)$.
5. The observation of quiescence must be after the reception of all previously sent coordination messages

(Assumption 2): if $a_i = m_{pq}(l)$ and $a_j = \delta$, $i < j$, then $e^r_{pq}(l) <_\sigma a_j$.

Given port $p \in \mathcal{P}$, we let $<^p_\sigma$ denote $<_\sigma$ restricted to $E_p(\sigma)$.

For a trace $\sigma \in (\mathcal{A}ct \cup \mathcal{M})^*$ we have that $<_\sigma$ is a strict partial order and so $(E(\sigma), <_\sigma)$ is a partially ordered set (poset). We will let $L(E(\sigma), <_\sigma)$ denote the set of linearisations of $(E(\sigma), <_\sigma)$. Since $<_\sigma$ defines the causalities between events in $E(\sigma)$, we have that $L(E(\sigma), <_\sigma)$ contains the set of all traces that can occur if each tester sees a sequence of events that is consistent with σ.

Consider, for example, $\sigma = ?i_1 !o_1 ?i_2$. This has three events: $?i_1$, $!o_1$, and $?i_2$. In addition, $?i_1 <_\sigma !o_1$ (the first rule) but the other pairs of events are unrelated under $<_\sigma$. Thus, there are several orders: the event $?i_2$ might have occurred before $?i_1$, between $?i_1$ and $!o_1$, or after $!o_1$.

Now consider $\sigma = ?i_1 !o_1 m_{12} ?i_2$. This has five events: $?i_1$, $!o_1$, e^s_{12}, e^r_{12}, and $?i_2$. In addition, we have that $?i_1 <_\sigma !o_1$ (the first rule); $!o_1 <_\sigma e^s_{12}$ (the third rule); $e^s_{12} <_\sigma e^r_{12}$ (the fourth rule); and $e^r_{12} <_\sigma ?i_2$ (the second rule). Thus, there is only one order in which the events can occur.

We now consider the situation in which $(\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$ and t' is a global test case formed from adding coordination messages to t: we wish to determine whether $(\sigma_1, \sigma_2, ?i_p)$ corresponds to a controllability problem in t'. First we define what it means for $(\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$ to correspond to a controllability problem in t'.

In the next section we show how coordination messages can be added to a test case to overcome controllability problems. An important point is that a tester should be able to know when to send an input on the basis of the observations it has made, rather than on the basis of observations it has yet to make but might later make. Let us suppose that we wish to use coordination messages to overcome a controllability problem in which the tester at port p should send input after σ_1 and not after σ_2. Then we need that the tester at p makes an observation, a local trace, after which it knows that it should send the input. This would not be the case if, for example, the tester should send the input after σ but not after $\sigma e^r_{qp}(l)$ since in this situation after observing σ the tester does not know whether to send the input or wait for $e^e_{qp}(l)$. Thus, after coordination messages are added the local traces at p should be different and not through there being extra coordination messages at the end of the trace corresponding to σ_2. By definition, coordination messages that might arrive at p at the end of a trace must have been sent after the last input at p and the last observation of quiescence. We therefore have the following definition of what it means for a pair of traces to correspond to a controllability problem.

Definition 10 Let us suppose that $(\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$ and t' is a global test case formed from t by adding coordination messages. Then $(\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$ *corresponds* to a controllability problem in t' if there exist traces σ^2_1, σ^2_2 such that the following hold.

1. $\sigma^2_1 ?i_p \in \mathcal{T}r(t')$ and there exists $\sigma' \in \mathcal{T}r(t')$ such that σ^2_2 is formed from σ' by removing coordination messages sent to p in σ' that are not in σ^2_1 and that are sent after the last input at p and the last observation of quiescence;
2. $\sigma^2_1 \downarrow_{\mathcal{A}ct} = \sigma_1$ and $\sigma^2_2 \downarrow_{\mathcal{A}ct} = \sigma_2$;
3. There exist linearisations σ'_1 and σ'_2 of $(E(\sigma^2_1), <_{\sigma^2_1})$ and $(E(\sigma^2_2), <_{\sigma^2_2})$ respectively such that $\sigma'_1 \downarrow_{\mathcal{A}ct_p \cup \mathcal{E}_p} = \sigma'_2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{E}_p}$.

We also say that σ^2_1 and σ^2_2 *implement* $(\sigma_1, \sigma_2, ?i_p)$.

This definition requires that there are traces σ^2_1 and σ^2_2 that correspond to σ_1 and σ_2 in the sense that if we remove the coordination messages from σ^2_1 and σ^2_2 then we obtain σ_1 and σ_2. The first condition allows us to remove coordination messages, to form σ^2_2, if these are sent to p but might arrive at the end of the trace; this avoids the possibility of the tester at p having to decide to send $?i_p$ on the basis of not observing such coordination message. The last part requires that there are possible traces in $L(E(\sigma^2_1), <_{\sigma^2_1})$ and $L(E(\sigma^2_2), <_{\sigma^2_2})$, that look identical to the tester at p.

Note that for some such $(\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$ and t', there is only one choice of σ^2_1 when finding σ^2_1 and σ^2_2 that implement $(\sigma_1, \sigma_2, ?i_p)$. This is because a global test case is deterministic and so cannot have a state from which it is possible to either send a coordination message or supply input. There may be several choices of σ' but it does not matter which is used since the resultant σ^2_2 all have the same sets of events at p and the same partial order defined on these events.

We will not want to generate the sets of linearisations of $(E(\sigma_1), <_{\sigma_1})$ and $(E(\sigma_2), <_{\sigma_2})$ since these sets may contain exponentially many sequence. We now define a predicate $Ctr(\sigma_1, \sigma_2, ?i_p, t')$ and below we prove that this is true if and only if $(\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$ corresponds to a controllability problem in t'.

Definition 11 Given $s \in \mathcal{IOTS}(I, O)$, global test case $t \in \mathcal{IOTS}(I \cup \mathcal{M}, O \cup \{\delta\})$, $(\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$, and t' formed from t by adding coordination messages, we let $Ctr(\sigma_1, \sigma_2, ?i_p, t')$ be true if and only if there exist σ^2_1, σ^2_2 with $\sigma^2_1 ?i_p \in \mathcal{T}r(t')$ such that the following hold:

1. There exists $\sigma' \in \mathcal{T}r(t')$ such that σ_2^2 can be produced from σ' by removing coordination messages to p that are not in σ_1^2 and that are sent after the last input at p and the last observation of quiescence;
2. $\sigma_1^2 \downarrow_{\mathcal{A}ct} = \sigma_1$ and $\sigma_2^2 \downarrow_{\mathcal{A}ct} = \sigma_2$;
3. $\sigma_1^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p} = \sigma_2^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p}$;
4. σ_1^2 and σ_2^2 contain the same sets of coordination messages sent to p; and
5. $G(<^p_{\sigma_1^2} \cup <^p_{\sigma_2^2})$ is acyclic.

Notice here that we need to define $<^p_{\sigma_1^2}$ and $<^p_{\sigma_2^2}$ on the same set of events at p; these correspond to vertices of $G(<^p_{\sigma_1^2} \cup <^p_{\sigma_2^2})$. However, this is straightforward since:

1. $\sigma_1^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p} = \sigma_2^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p}$ and so we give corresponding events in $\sigma_1^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p}$ and $\sigma_2^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p}$ the same name.
2. Labels of coordination messages are not repeated so for every coordination message $m_{qq'}(l)$ that is in σ_1^2 and σ_2^2 we use the same names for the events that involve $e^s_{qq'}(l)$ and we use the same names for the events that involve $e^r_{qq'}(l)$.

Before proving that $Ctr(\sigma_1, \sigma_2, ?i_p, t')$ is the predicate we require we prove a result regarding posets.

Lemma 1 *Let us suppose that $(A, <)$ is a poset, A is finite and A' is a subset of A. If σ' is a linearisation of $(A', <)$ then there exists a linearisation σ of $(A, <)$ such that $\sigma' = \sigma \downarrow_{A'}$.*

Proof We will use proof by induction on the size of A. The result clearly holds in the base case where A is empty. Now assume that the result holds whenever A has fewer than $k > 0$ elements and consider a case where A has k elements.

The result holds immediately if σ' has length 1 or less so we will assume that σ' has length at least 2. We can rewrite σ' as $\sigma'_1 a$ and know that a is not before any element of σ'_1 under $<$. Now define the set A_1 that contains all elements of A that are in σ'_1 or are before elements of σ'_1 under $<$. Clearly A_1 does not contain a. Let $A_2 = A \setminus A_1$.

Since σ' has length at least two we know that A_1 and A_2 are both non-empty. From the inductive hypothesis we know that there are linearisations σ_1 and σ_2 of $(A_1, <)$ and $(A_2, <)$ respectively such that $\sigma_1 \downarrow_{A'} = \sigma'_1$ and $\sigma_2 \downarrow_{A'} = a$. In addition, we cannot have $a_1 \in A_1$ and $a_2 \in A_2$ such that $a_2 < a_1$. Thus, $\sigma = \sigma_1 \sigma_2$ is a linearisation of $(A, <)$ and $\sigma \downarrow_{A'} = \sigma_1 \downarrow_{A'} \sigma_2 \downarrow_{A'} = \sigma'_1 a = \sigma'$ as required. The result therefore holds.

We now prove that $Ctr(\sigma_1, \sigma_2, ?i_p, t')$ is the predicate we require.

Proposition 2 *Let us suppose that $\sigma_1, \sigma_2 \in Ctr(t)$ and that t' has been formed by adding coordination messages to t. Then $(\sigma_1, \sigma_2, ?i_p)$ corresponds to a controllability problem in t' if and only if $Ctr(\sigma_1, \sigma_2, ?i_p, t')$ is true.*

Proof First assume that $(\sigma_1, \sigma_2, ?i_p)$ corresponds to a controllability problem in t'. Thus, there exist traces σ_1^2, σ_2^2 as in Definition 10 with $\sigma_1^2 \downarrow_{\mathcal{A}ct} = \sigma_1$ and $\sigma_2^2 \downarrow_{\mathcal{A}ct} = \sigma_2$, and $\sigma_1^2 ?i_p \in \mathcal{T}r(t')$ such that there exist linearisations σ'_1 and σ'_2 of $(E(\sigma_1^2), <_{\sigma_1^2})$ and $(E(\sigma_2^2), <_{\sigma_2^2})$ respectively with $\sigma'_1 \downarrow_{\mathcal{A}ct_p \cup \mathcal{E}_p} = \sigma'_2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{E}_p}$. Clearly, we must therefore have that $\sigma_1^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p} = \sigma_2^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p}$ and that σ_1^2 and σ_2^2 contain the same sets of coordination messages sent to p and so it is sufficient to prove that $G(<^p_{\sigma_1^2} \cup <^p_{\sigma_2^2})$ is acyclic. Let σ'' denote $\sigma'_1 \downarrow_{(\mathcal{A}ct_p \cup \mathcal{E}_p)}$. Thus σ'' is consistent with both $<^p_{\sigma_1^2}$ and $<^p_{\sigma_2^2}$ and so $G(<^p_{\sigma_1^2} \cup <^p_{\sigma_2^2})$ is acyclic as required.

Now assume that $Ctr(\sigma_1, \sigma_2, ?i_p, t')$ is true. Thus, there exist σ_1^2, σ_2^2 as in Definition 11 with $\sigma_1^2 ?i_p \in \mathcal{T}r(t')$ such that $\sigma_1^2 \downarrow_{\mathcal{A}ct} = \sigma_1$ and $\sigma_2^2 \downarrow_{\mathcal{A}ct} = \sigma_2$; $\sigma_1^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p} = \sigma_2^2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{M}_p}$; σ_1^2 and σ_2^2 contain the same sets of coordination messages sent to p; and $G(<^p_{\sigma_1^2} \cup <^p_{\sigma_2^2})$ is acyclic. Consider these σ_1^2, σ_2^2 and it is sufficient to prove that there exist linearisations σ'_1 and σ'_2 of $(E(\sigma_1^2), <_{\sigma_1^2})$ and $(E(\sigma_2^2), <_{\sigma_2^2})$ respectively such that $\sigma'_1 \downarrow_{\mathcal{A}ct_p \cup \mathcal{E}_p} = \sigma'_2 \downarrow_{\mathcal{A}ct_p \cup \mathcal{E}_p}$.

Since $G(<^p_{\sigma_1^2} \cup <^p_{\sigma_2^2})$ is acyclic, there must be some σ'' that is a sequence of the events in $E_p(\sigma_1^2)$ and $E_p(\sigma_2^2)$ that is a linearisation under both $<^p_{\sigma_1^2}$ and $<^p_{\sigma_2^2}$ and so also under $<_{\sigma_1^2}$ and $<_{\sigma_2^2}$. In addition, the sets of events in $E_p(\sigma_1^2)$ and $E_p(\sigma_2^2)$ are identical. Thus, by Lemma 1, there exist linearisations σ'_1 and σ'_2 of $(E(\sigma_1^2), <_{\sigma_1^2})$ and $(E(\sigma_2^2), <_{\sigma_1^2})$ respectively such that $\sigma'_1 \downarrow_{(\mathcal{A}ct_p \cup \mathcal{E}_p)} = \sigma'' = \sigma'_2 \downarrow_{(\mathcal{A}ct_p \cup \mathcal{E}_p)}$ as required.

It is clear that we can decide whether predicate $Ctr(\sigma_1, \sigma_2, ?i_p, t')$ holds in polynomial time. In the next section we use this predicate in an iterative algorithm that adds coordination messages to a test case in order to overcome controllability problems.

In this paper we have assumed that when quiescence is observed then all coordination messages that have been sent are received. If we wished to relax this assumption then, for a trace σ, we would obtain a different definition of $<_\sigma$ but otherwise the method would remain the same. If we do not require a tester to wait to receive coordination messages then again we obtain a slightly different definition of $<_\sigma$.

7 Using coordination messages to overcome controllability problems

This section explores the use of coordination messages to make a global test case t controllable and gives an algorithm for adding such coordination messages. We restrict attention to the case where there are only finitely many controllability problems in the set $Ctr(t)$. When there are no coordination messages, controllability problems occur if the tester at port p should supply an input $?i_p$ after a global trace σ_1 but not after a global trace σ_2 with $\sigma_1 \downarrow_{\mathcal{A}ct_p} = \sigma_2 \downarrow_{\mathcal{A}ct_p}$. However, if another tester q observes a difference then there is the potential for it to send a coordination message to p so that the tester at p can differentiate between σ_1 and σ_2.

First we explore some pathological situations where we cannot expect to be able to overcome controllability problems. Let us suppose that the tester at p should send an input after σ_1 but not after σ_2 and $\sigma_1 \sim \sigma_2$. Then we have a problem: no tester observes a difference. These global traces are, to the environment, equivalent and so we should expect a global test case t to consider them to be equivalent. Further, let us suppose that we want to send input $?i_2$ after $!o_1!o_2$ but not after $!o_1!o_1!o_2$. In order to differentiate between these cases we need to send a coordination message from the tester at port 1 to the tester at port 2 after $!o_1!o_1$ but after observing $!o_2$ the tester at port 2 does not know whether to wait for this coordination message.

The problem in the second example is that the tester at port 2 needs to make a decision based on the absence of an observation (the message from the tester at port 1) rather than on the basis of an observation made. More generally, we have a problem if for every port $q \neq p$ we have that $\sigma_1 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$ is a prefix of $\sigma_2 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$ and the tester at p should send input $?i_p$ after σ_1 but not after σ_2. To overcome such a situation with coordination messages from q to p, the tester at q would have to send a message if the extra observations after $\sigma_1 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$ are made and so we would require the tester at p to send its input based on not observing such (asynchronous) coordination messages. We will not consider such situations: the idea is that the tester should be able to make a decision to send an input or coordination message on the basis of observations it has made, rather than observations it has not made yet but might still make.

Definition 12 A global test case t, that has no coordination messages, is *strongly uncontrollable* for IOTS s if there exists $\sigma_1, \sigma_2 \in \mathcal{T}r(s,t)$ and input $?i_p \in I_p$ with $\sigma_1 ?i_p \in \mathcal{T}r(s,t)$, $\sigma_2 ?i_p \notin \mathcal{T}r(s,t)$, $\sigma_1 \downarrow_{\mathcal{A}ct_p} = \sigma_2 \downarrow_{\mathcal{A}ct_p}$ such that $\sigma_1 \downarrow_{\mathcal{A}ct_q}$ is a prefix of $\sigma_2 \downarrow_{\mathcal{A}ct_q}$ for all $q \in \mathcal{P} \setminus \{p\}$.

We therefore assume that any global test case to be used is not strongly uncontrollable. Below we give an algorithm that adds coordination messages to a global test case that is not strongly uncontrollable. This operates by identifying all of the controllability problems, using Algorithm 1, and then resolving these one at a time.

We now show how we can resolve one controllability problem using coordination messages. In the following we consider two cases for some $c = (\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$ that corresponds to a controllability problem in test case t' formed by adding coordination messages to t. Let σ_1^2 and σ_2^2 be traces of t' that implement c. Let $\sigma_1^q = \sigma_1^2 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$ and $\sigma_2^q = \sigma_q^2 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$ for some port q such that $\sigma_1^2 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$ is not a prefix of $\sigma_2^2 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$. Since we only consider test cases that are not strongly uncontrollable, there must be some such q. In the first case, σ_2^q is a proper prefix of σ_1^q. Thus, if the tester at q observes σ_1^q and then sends a message to the tester at p then the tester at p is able to distinguish between these two traces (σ_1^2 and σ_2^2) through receiving the coordination message and so this controllability problem is resolved.

In the second case, σ_2^q is not a prefix of σ_1^q. However, there must be some longest common prefix of σ_1^q and σ_2^q and we call this σ^q. Further, there must be some $a \in \mathcal{A}ct \cup \mathcal{M}$ such that $\sigma^q a$ is a prefix of σ_1^q. Thus, it is sufficient for the tester at q to send a coordination message to the tester at p after it observes $\sigma^q a$ since this allows the tester at p to distinguish between σ_1^2 and σ_2^2.

Note that in both cases we must have that the tester at q sends the coordination message added whenever it observes the preceding sequence of events (σ_1^q or $\sigma^q a$), even as part of other traces; otherwise the sending of this coordination message introduces a controllability problem.

Proposition 3 *Let us suppose that $s \in \mathcal{IOTS}(I,O)$, t is a global test case that is not strongly uncontrollable and t' is a global test case that has been formed by adding coordination messages to t. Further, assume that the sending of coordination messages in t' cannot lead to controllability problems. Let us suppose that $c = (\sigma_1, \sigma_2, ?i_p) \in Ctr(t)$ corresponds to a controllability problem in t' and that σ_1^2 and σ_2^2 implement c. Let q be a port such that $\sigma_1^q = \sigma_1^2 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$ is not a prefix of $\sigma_2^q = \sigma_2^2 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$. Let t'' be the global test case produced from t' by applying one of the following using a label l not used in coordination messages in t'.*

1. *Case 1: σ_2^q is a proper prefix of σ_1^q. Form t'' by adding to t' the coordination message $m_{qp}(l)$ after every minimal trace σ'' of t' such that $\sigma'' \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q} =$*

σ_1^q. *Here minimality means that no proper prefix of* σ'' *has projection* σ_1^q *on* q.

2. *Case 2:* σ_2^q *is not a prefix of* σ_1^q. *Let* σ^q *denote the longest common prefix of* σ_1^q *and* σ_2^q. *Define* $a \in \mathcal{A}ct \cup \mathcal{M}$ *such that* $\sigma^q a \in pref(\sigma_1^q)$ *and form* t'' *by adding to* t' *the coordination message* $m_{qp}(l)$ *after every minimal trace* σ'' *of* t' *such that* $\sigma'' \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q} = \sigma^q a$.

Then c *does not correspond to a controllability problem in* t''.

Proof Since the sending of coordination messages in t' does not cause controllability problems, a coordination message from a port p' must have been added in a consistent manner: if it is sent after a trace σ' then it must be sent after all σ'' with $\sigma'' \downarrow_{\mathcal{A}ct_{p'}} = \sigma' \downarrow_{\mathcal{A}ct_{p'}}$. Thus, while there may be several choices of σ_2^2, we have that if $\sigma_2^2 \downarrow_{\mathcal{A}ct_q}$ is a prefix of $\sigma_1^2 \downarrow_{\mathcal{A}ct_q}$ for one choice then it is a prefix for all such choices since the choices can only differ in coordination messages. Thus, the choice of σ_2^2 is not important.

There are two cases. In the first case a coordination message is sent to p after q observes σ_1^q and this occurs if the test case follows the path with label σ_1^2 but not if it follows the path with label σ_2^2 and so the result holds.

Similarly, in the second case a coordination message is sent to p after q observes $\sigma^q a$ and this occurs if the test case follows the path with label σ_1^2 but not if it follows the path with label σ_2^2 and so the result holds.

Algorithm 2 Adding coordination messages to global test case t

Input global test case t for s and finite set $C = Ctr(t)$ of controllability problems.
Let $t' = t$.
while $C \neq \emptyset$ **do**
  Choose some $c = (\sigma_1, \sigma_2, ?i_p) \in C$ such that there is no $c' = (\sigma_1', \sigma_2', ?i_p') \in C$ with σ_1' a proper prefix of σ_1.
  Choose σ_1^2 and σ_2^2 that implement c in t'.
  Choose a port $q \neq p$ such that $\sigma_1^q = \sigma_1^2 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$ is not a prefix of $\sigma_2^q = \sigma_2^2 \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q}$.
  Choose a label l that has not been previously used.
  if σ_2^q is a prefix of σ_1^q **then**
    Add to t' the coordination message $m_{qp}(l)$ after every minimal trace σ such that $\sigma \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q} = \sigma_1^q$.
  else
    Let σ^q denote the longest common prefix of σ_1^q and σ_2^q.
    Define $a \in \mathcal{A}ct \cup \mathcal{M}$ such that $\sigma^q a \in \text{pref}(\sigma_1^q)$.
    Add to t' the coordination message $m_{qp}(l)$ after every minimal trace σ such that $\sigma \downarrow_{\mathcal{A}ct_q \cup \mathcal{M}_q} = \sigma^q a$.
  end if
  Remove from C all tuples that do not correspond to controllability problems in t'.
end while
Return t'

Theorem 1 *If Algorithm 2 is given a global test case* t *that is not strongly uncontrollable then it returns a controllable global test case.*

Proof The algorithm adds coordination messages in a way that cannot introduce controllability problems. As noted earlier, for a given c and t' there is only one choice of σ_1^2. Further, the coordination message added distinguishes between σ_1^2 and all of the σ_2^2 such that σ_1^2 and σ_2^2 implement c. Thus, by Proposition 3, an iteration of the algorithm overcomes all controllability problems associated with pairs of traces that implement c. As a result, the algorithm must terminate and does so once the current global test case has no controllability problems and so the result follows.

The number of iterations of Algorithm 2 is bounded above by the size of $Ctr(t)$ (the sum of the lengths of the sequences in this set) and each iteration takes time that is polynomial in terms of the size of $Ctr(t)$ and the size of t. Thus, Algorithm 2 operates in time that is polynomial in terms of the size of $Ctr(t)$ and the size of t.

Algorithm 2 restricts the order in which elements of $Ctr(s,t)$ are considered: we do not choose some $c = (\sigma_1, \sigma_2, ?i_p) \in C$ if there exists $c' = (\sigma_1', \sigma_2', ?i_p') \in C$ with σ_1' a proper prefix of σ_1. This is because in such a situation there is the potential for the coordination message added to overcome the controllability problem identified by c' to also overcome the controllability problem identified by c but the converse is not the case. However, there may still be choices and the potential to make a suboptimal choice. Let us suppose, for example, that we wish to apply a global test case t that has a trace $!o_1!o_2?i_3$. We might identify two elements of $Ctr(s,t)$: $(!o_1!o_2, \epsilon, ?i_3)$ and $(!o_1!o_2, !o_1, ?i_3)$. If we first consider $(!o_1!o_2, \epsilon, ?i_3)$ then we could add the message $m_{13}(l)$ after $!o_1$ for some label l. We then have to add a coordination message for the second element of $Ctr(s,t)$. In contrast, if we first consider $(!o_1!o_2, !o_1, ?i_3)$ then we add the message $m_{23}(l)$ after $!o_1!o_2$ for some label l and this resolves both controllability problems. Further optimisations are a topic for future work but below we prove that the optimisaton problem is NP-hard.

Now let us suppose that we apply Algorithm 2 to t_0. In t_0, controllability problems are caused by the input of the first $?delete_2$ and by each of the instance of $?in_1$. Algorithm 2 might start with $(?update_1!ok_1, \epsilon, ?delete_2)$. This would be resolved by adding a coordination message $m_{12}(l_1)$ after $?update_1!ok_1$. It might then consider traces $?update1!ok_1 m_{12}(l_1)?delete_2!ok_2$ and $?update1!ok_1$ $m_{12}(l_1)?delete_2$ and input $?in_1$ and add $m_{21}(l_2)$ after $?update1!ok_1 m_{12}(l_1)?delete_2!ok_2$. Finally, it might con-

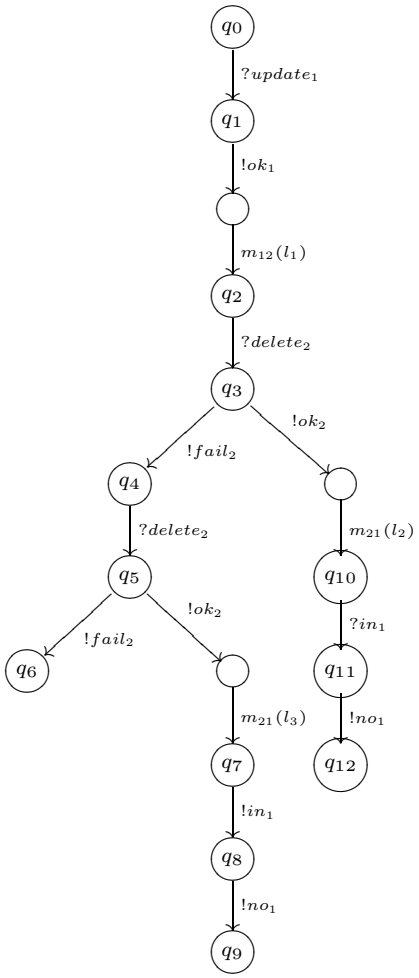

Fig. 9 Global Test Case t_0 With Coordination Messages Added

sider traces $?update1!ok_1m_{12}(l_1)?delete_2!fail_2?delete_2$ $!ok_2$ and $?update1!ok_1m_{12}(l_1)?delete_2!fail_2?delete_2$ and input $?in_1$ and add message $m_{21}(l_3)$ after $?update1$ $!ok_1m_{12}(l_1)?delete_2!fail_2?delete_2!ok_2$. This leads to the controllable test case shown in Figure 9.

In this example, it was sufficient to use three coordination messages for a test case that contained 12 events. In contrast, the approach of [23] requires several coordination messages for each event.

8 The complexity of the optimisation problem

We have described an algorithm that overcomes controllability problems in a test case by adding coordination messages. However, the order in which the controllability problems are considered can vary and this might affect the number of coordination messages used. It is natural to ask how we can minimise the number of coordination messages used and we will prove that this optimisation problem is NP-hard even if we restrict attention to tree-like global test cases. We will do this by showing that an instance of the Hitting Set Problem can be encoded as an instance of our problem.

Definition 13 The *Hitting Set Problem* is: given a universe U and a set of subsets $S_1, \ldots, S_z$ of U, find a smallest subset S of U such that for all $1 \leq i \leq z$ we have that S contains an element of S_i.

The following result has been proved [24].

Theorem 2 *The Hitting Set Problem is NP-complete.*

It is now possible to prove that the problem of finding an optimal (minimal) set of coordination messages to add is NP-hard.

Theorem 3 *The problem of adding a minimum number of coordination messages to a global test case in order to make it controllable is NP-hard and this is the case even if we restrict attention to tree-like global test cases.*

Proof We will assume that we have an instance of the Hitting Set Problem defined by sets $S_1, \ldots, S_z$ and $U = \{x_1, \ldots, x_k\}$. We will also assume that $U = S_1 \cup \ldots \cup S_z$; if this is not the case then we simply remove elements from U to ensure that this property holds. The proof will proceed by producing a global test case t such that a smallest set of coordination messages, whose inclusion overcomes the controllability problems in t, defines a solution to this instance of the Hitting Set Problem.

We will use $k+2$ ports and form a global test case t that starts by branching on outputs $!o^1, \ldots, !o^k$ at port $k+1$. For each $1 \leq i \leq k+2$ we let $!o_i$ denote an output at port i. The controllability problems will be caused by input $?i$ at port $k+2$ and for $1 \leq i \leq k$ we will use the traces in t that start with $!o^i$ to introduce controllability problems that can be solved using a coordination message sent from port p to port $k+2$ if and only if $x_p \in S_i$. We achieve this in the following way.

Let $p_1, \ldots, p_r$ be such that $x_p \in S_i$ if and only if $p \in \{p_1, \ldots, p_r\}$ and so these indices define the set S_i. In t we follow $!o^i$ by two paths and so t branches after $!o^i$. One path has label $!o_{p_1} \ldots !o_{p_r}!o_{k+2}?i$ while the other path has label $!o_{k+2}$. The application of $?i$ causes a controllability problem since the labels of these paths have the same projection at port $k+2$. In addition, we can resolve this controllability problem by adding a coordination message from port p to port $k+2$ if and only if $p \in \{p_1, \ldots, p_r\}$ and the tester at p sends the coordination message after observing $!o_p$.

We have defined a global test case t such that each transition involving the sending of $?i$ leads to a controllability problem. In addition, since these are the only inputs, there are no additional controllability problems in t. Further, for the instance of $?i$ on the branch of t that starts with $!o^i$ we have that the sending of a coordination message from port p can only help overcome the controllability problem if $x_p \in S_i$. Since such a coordination message from the tester at p to the tester at $k+2$ can be used to overcome this controllability problem, and we are interested in minimising the number of coordination messages used, we can assume that all coordination messages used are sent to port $k+2$. Thus, we have that a coordination message can overcome this controllability problem if and only if it is sent from port p to port $k+2$ after $!o_p$ is observed and we have that $x_p \in S_i$.

Now let us assume that we have found a smallest set C of such coordination messages that can be added to t in order to overcome the controllability problems. Each coordination message in C involves the tester at a port p sending a coordination message to the tester at port $k+2$ after observing $!o_p$. Let C' denote the set of such p. For all S_i, $1 \leq i \leq z$, we must have that C' contains some such p with $x_p \in S_i$ and so C' defines a hitting set. Further, every hitting set defines some such set of coordination messages that overcomes the controllability problems in t. Thus, C defines a solution to the instance of the Hitting Set Problem. Finally, note that we can generate t in polynomial time and so the result follows from Theorem 2.

For tree-like test cases the problem is in NP. To see this note that we can place a polynomial upper bound on the number of coordination messages required. Thus, we can reduce the problem to that of deciding, for k, whether the controllability problems can be resolved using k coordination messages. This is in NP since we can decide in polynomial time whether a global test case with k coordination messages is controllable.

While the Hitting Set Problem is NP-complete, it is the dual of the Set Cover Problem and it is known that greedy algorithms are effective in solving the Set Cover Problem. Specifically, a greedy algorithm has an approximation ratio in the order of the log of the size of the largest set [8,27]. Thus, it seems likely that greedy algorithms will be effective in producing controllable test cases with relatively few coordination messages but the performance of such algorithms in practice is a problem for future work.

9 Conclusions

When testing a system that has physically distributed interfaces, called ports, it is normal to place one tester at each port. The tester at port p observes only the events at p. This can lead to controllability problems since the tester at p can only decide when to send input based on events observed at p. This has led to interest in approaches that overcome controllability problems by the sending of coordination messages between the testers. However, almost all previous work in this area has considered testing from a deterministic finite state machine.

This paper investigated the addition of coordination messages to a test case for use in testing against an input output transition system (IOTS). We introduced notation for such coordination messages and defined what it means for a test case with coordination messages to be controllable. Despite controllability being defined in terms of a potentially infinite number of traces, we gave a polynomial time algorithm that decides whether a test case is controllable. The algorithm also returned a characterisation of the set of controllability problem.

We then considered the problem of overcoming controllability problems in a test case. We restricted attention to test cases that have only a finite number of controllability problems. We gave an algorithm that adds coordination messages to a test case in order to overcome controllability problems. This algorithm operated in time that is polynomial in terms of the size of the set of controllability problems (the sum of the lengths of the traces in this set) and the size of the test case. Finally, we proved that the problem of producing a minimal sufficient set of coordination messages is NP-hard, even if we restrict attention to tree-like test cases.

In this paper we have assumed that all coordination messages that have been sent will arrive before quiescence is observed. We also assume that if a path in a test case includes the sending of a coordination message from the tester at p to the tester at q before an input $?i_q$ at q then the tester at q will not supply input $?i_q$ until the coordination message has arrived. An alternative would be to allow the tester to indicate which inputs have this property or even for a given input to not be sent until certain coordination messages have been received. However, it is straightforward to adapt the approach to different assumptions. Essentially, different assumptions lead to slightly different definitions of the set of traces that can result when testing using a global test case t. These changes would require us to make small changes to the algorithms.

There are several lines of future research. First, while we have shown how coordination messages can be used to overcome controllability problems, it would be interesting to develop heuristics that are effective in reducing the number of messages used. There is also the problem of minimising the number of labels required in messages between testers. The motivation for minimising the number of labels used is that using fewer labels can lead to shorter messages. There may also be scope to use similar approaches with models where a transition can have multiple inputs and outputs [15,3]. Some previous work on overcoming controllability problems when testing from a deterministic finite state machine has considered timing properties [25] and it would be interesting to consider such properties in the context of testing from an IOTS. There is also the problem of making a test case controllable when there are infinitely many controllability problems. However, it should be possible to directly extend the method to an important situation: the test case is used for stress testing and is a cycle containing an acyclic test case t. Here we can create a test case t' that involves repeating t a sufficient number of times; we can then apply the method to t'. However, there is a need to determine the required number of repetitions of t to use. Finally, there is a need to integrate this and similar approaches into test languages such as TTCN-3 [37] and to carry out real-world case studies.

Acknowledgments

I would like to thanks Ana Cavalcanti, Marie-Claude Gaudel, and Manuel Núñez for many useful discussions that contributed to the ideas in this paper. I would also like to thank the anonymous reviewers, whose comments improved the paper.

References

1. Alur, R., Etessami, K., Yannakakis, M.: Inference of message sequence charts. IEEE Transactions on Software Engineering **29**(7), 623–633 (2003)
2. Barnett, M., Grieskamp, W., Nachmanson, L., Schulte, W., Tillmann, N., Veanes, M.: Towards a tool environment for model-based testing with AsmL. In: Formal Approaches to Testing, *Lecture Notes in Computer Science*, vol. 2931, pp. 252–266. Springer-Verlag, Montreal, Canada (2003)
3. von Bochmann, G., Haar, S., Jard, C., Jourdan, G.V.: Testing systems specified as partial order input/output automata. In: 20th IFIP TC 6/WG 6.1 International Conference on Testing of Software and Communicating Systems (TestCom/FATES 2008), *Lecture Notes in Computer Science*, vol. 5047, pp. 169–183. Springer (2008)
4. Boyd, S., Ural, H.: The synchronization problem in protocol testing and its complexity. Information Processing Letters **40**(3), 131–136 (1991)
5. Cacciari, L., Rafiq, O.: Controllability and observability in distributed testing. Information and Software Technology **41**(11–12), 767–780 (1999)
6. Cartaxo, E.G., Machado, P.D.L., Neto, F.G.O.: On the use of a similarity function for test case selection in the context of model-based testing. Software Testing, Verification and Reliability **21**(2), 75–100 (2011)
7. Chen, W., Ural, H.: Synchronizable checking sequences based on multiple UIO sequences. IEEE/ACM Transactions on Networking **3**, 152–157 (1995)
8. Chvatal, V.: A greedy heuristic for the set-covering problem. Mathematics of Operations Research **4**, 233–235 (1979)
9. Cunha de Almeida, E., Matynowske, J.E., Sunyé, G., Traon, Y.L., Valduriez, P.: Efficient distributed test architecture for large-scale systems. In: International Conference on Testing Software and Systems (ICTSS 2010), Lecture Notes in Computer Science, p. to appear. Springer-Verlag (2010)
10. Dssouli, R., von Bochmann, G.: Error detection with multiple observers. In: Protocol Specification, Testing and Verification V, pp. 483–494. Elsevier Science (North Holland) (1985)
11. Dssouli, R., von Bochmann, G.: Conformance testing with multiple observers. In: Protocol Specification, Testing and Verification VI, pp. 217–229. Elsevier Science (North Holland) (1986)
12. Farchi, E., Hartman, A., Pinter, S.: Using a model-based test generator to test for standard conformance. IBM systems journal **41**(1), 89–110 (2002). URL www.research.ibm.com/journal/sj/411/farchi.pdf
13. Grieskamp, W.: Multi-paradigmatic model-based testing. In: Formal Approaches to Software Testing and Runtime Verification (FATES/RV 2006), *Lecture Notes in Computer Science*, vol. 4262, pp. 1–19. Springer (2006)
14. Grieskamp, W., Kicillof, N., Stobie, K., Braberman, V.: Model-based quality assurance of protocol documentation: tools and methodology. The Journal of Software Testing, Verification and Reliability **21**(1), 55–71 (2011)
15. Haar, S., Jard, C., Jourdan, G.V.: Testing input/output partial order automata. In: Joint 19th IFIP TC6/WG6.1 Int. Conf. on Testing of Software and Communicating Systems, TestCom'07, and 7th Int. Workshop on Formal Approaches to Software Testing, FATES'07, LNCS 4581, pp. 171–185. Springer (2007)
16. Hierons, R.M.: Testing a distributed system: generating minimal synchronised test sequences that detect output-shifting faults. Information and Software Technology **43**(9), 551–560 (2001)
17. Hierons, R.M.: Testing from a non-deterministic finite state machine using adaptive state counting. IEEE Transactions on Computers **53**(10), 1330–1342 (2004)
18. Hierons, R.M., Bogdanov, K., Bowen, J.P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Kapoor, K., Krause, P., Lüttgen, G., Simons, A.J.H., Vilkomir, S.A., Woodward, M.R., Zedan, H.: Using formal specifications to support testing. ACM Computing Surveys **41**(2) (2009)
19. Hierons, R.M., Merayo, M.G., Núñez, M.: Controllable test cases for the distributed test architecture. In: 6th International Symposium on Automated Technology for Verification and Analysis (ATVA 2008), *Lecture Notes in Computer Science*, vol. 5311, pp. 201–215. Springer (2008)

20. Hierons, R.M., Merayo, M.G., Núñez, M.: Implementation relations for the distributed test architecture. In: 20th IFIP TC 6/WG 6.1 International Conference on the Testing of Software and Communicating Systems (TestCom/FATES 2008), *Lecture Notes in Computer Science*, vol. 5047, pp. 200–215. Springer (2008)
21. Hierons, R.M., Ural, H.: Checking sequences for distributed test architectures. Distributed Computing **21**(3), 223–238 (2008)
22. Jacob, J.L.: Refinement of shared systems. In: J. McDermid (ed.) The Theory and Practice of Refinement: Approaches to the Formal Development of Large-Scale Software Systems, pp. 27–36. Butterworths (1989)
23. Jard, C., Jéron, T., Kahlouche, H., Viho, C.: Towards automatic distribution of testers for distributed conformance testing. In: TC6 WG6.1 Joint International Conference on Formal Description Techniques and Protocol Specification, Testing and Verification (FORTE 1998), *IFIP Conference Proceedings*, vol. 135, pp. 353–368. Kluwer (1998)
24. Karp, R.M.: Reducibility among combinatorial problems. In: R.E. Miller, J.W. Thatcher (eds.) Complexity of Computer Computations. Plenum Press, New York-London (1972). 85–103
25. Khoumsi, A.: A temporal approach for testing distributed systems. IEEE Transactions on Software Engineering **28**(11), 1085–1103 (2002)
26. Lee, D., Yannakakis, M.: Principles and methods of testing finite-state machines - a survey. Proceedings of the IEEE **84**(8), 1089–1123 (1996)
27. Levin, A.: Approximating the unweighted k-set cover problem: Greedy meets local search. SIAM Journal of Discrete Mathematics **23**(1), 251–264 (2008)
28. Luo, G., Dssouli, R., v. Bochmann, G.: Generating synchronizable test sequences based on finite state machine with distributed ports. In: The 6th IFIP Workshop on Protocol Test Systems, pp. 139–153. Elsevier (North-Holland) (1993)
29. Mitchell, B.: Resolving race conditions in asynchronous partial order scenarios. IEEE Transactions on Software Engineering **31**(9), 767–784 (2005)
30. Sarikaya, B., v. Bochmann, G.: Synchronization and specification issues in protocol testing. IEEE Transactions on Communications **32**, 389–395 (1984)
31. Tai, K.C., Young, Y.C.: Synchronizable test sequences of finite state machines. Computer Networks and ISDN Systems **30**(12), 1111–1134 (1998)
32. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM Journal of Computing **1**(2) (1972)
33. Tretmans, J.: Test generation with inputs, outputs and repetitive quiescence. Software – Concepts and Tools **17**(3), 103–120 (1996)
34. Tretmans, J., Verhaard, L.: A queue model relating synchronous and asynchronous communication. In: IFIP TC6/WG6.1 Twelfth International Symposium on Protocol Specification, Testing and Verification, *IFIP Transactions*, vol. C-8, pp. 131–145. North-Holland (1992)
35. Ural, H., Wang, Z.: Synchronizable test sequence generation using UIO sequences. Computer Communications **16**(10), 653–661 (1993)
36. Ural, H., Williams, C.: Constructing checking sequences for distributed testing. Formal Aspects of Computing **18**(1), 84–101 (2006)
37. V3.1.1, E....: Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. ETSI, Sophia, Antipolis, France (2005)
38. Wu, W.J., Chen, W.H., Tang, C.Y.: Synchronizable test sequence for multi-party protocol conformance testing. Computer Communications **21**(13), 1177–1183 (1998)
39. Young, Y.C., Tai, K.C.: Observational inaccuracy in conformance testing with multiple testers. In: IEEE 1st workshop on application-specific software engineering and technology, pp. 80–85 (1998)