Generating Complete Controllable Test Suites for Distributed Testing

Robert M. Hierons, *Senior Member, IEEE*

Abstract—A test suite is m-complete for finite state machine (FSM) M if it distinguishes between M and all faulty FSMs with m states or fewer. While there are several algorithms that generate m-complete test suites, they cannot be directly used in distributed testing since there can be additional controllability and observability problems. Indeed, previous results show that there is no general method for generating an m-complete test suite for distributed testing and so the focus has been on conditions under which this is possible. This paper takes a different approach, which is to generate what we call c_m-complete test suites: controllable test suites that distinguish an FSM N with no more than m states from M if this is possible in controllable testing. Thus, under the hypothesis that the system under test has no more than m states, a c_m-complete test suite achieves as much as is possible given the restriction that testing should be controllable. We show how the problem of generating a c_m-complete test suite can be mapped to the problem of generating an m-complete test suite for a partial FSM. Thus, standard test suite generation methods can be adapted for use in distributed testing.

Index Terms—Software engineering/software/program verification, software engineering/testing and debugging, systems and software, distributed testing, test suite generation, checking experiment.

1 Introduction

Testing is one of the most important parts of the software development process but is typically manual, error prone and expensive. This has led to interest in automation, with one of the most promising approaches being model based testing (MBT) where automation is based on a model. This model might be a specification of the system under test (SUT) or some aspect of the behaviour that is of interest to the tester. Industrial experience suggests that MBT can be significantly more efficient than manual testing [1].

Most MBT models are behavioural and state-based: they describe the allowed sequences of inputs and outputs using a model that has an internal state. While there are many different languages that can be used, the semantics are typically described using finite state machines (FSMs) or input output transition systems (IOTSs) (possibly with additional information such as time). There has thus been interest in automating testing from an FSM [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] or an IOTS [13], [14], [15], [16], [17]. Interest in FSM-based testing goes back to Moore's 1956 paper on Gedanken Experiments [7], with Hennie introducing an automated test generation algorithm in 1964 [5].

There has been interest in methods that generate a test suite that is guaranteed to determine whether the SUT is correct, under the assumption that the SUT satisfies certain conditions. The initial work assumed that the SUT is an unknown FSM N with no more states than the specification [5]. This was generalised to there being a known upper bound m on the number of states of N, with test suite $\mathcal{T}$ being *m-complete* if any faulty FSM with no more than m states fails $\mathcal{T}$. The first published technique to generate m-complete test suites was for deterministic FSMs [2], [18]. Later state counting was introduced for testing from a non-deterministic finite state machine (NFSM) [8], [9], [12], [19] and then used for testing from a partial deterministic FSM [10].

MBT work typically assumes that a single tester interacts synchronously with the SUT. However, in practice there may be multiple physically distributed testers, each interacting with a separate *port* (interface) of the SUT: we might have *distributed testing*. Each tester observes the events in which it participates and so the global sequence of inputs and outputs is not observed. In practice there is no global clock and if we cannot synchronise the testers through an external mechanism then we have the ISO standardised distributed test architecture [20]. This paper considers the problem of testing from a deterministic FSM that has multiple ports (a *multi-port FSM*) when using the distributed test architecture. The distributed test architecture can lead to *controllability problems*, where a local tester at port p cannot know when to supply its inputs since it does not observe inputs and outputs at other ports [3], [11]. We then cannot guarantee that the inputs arrive in the correct order. As is usual, we use input sequences as test cases. It is worth noting that there are more general notions of test cases, such as decision trees, automata, and game strategies. Since we are testing from an FSM, input and output alternate. Thus, for each of the above types of test cases we have that at each point in a test case t, an input is applied and then the resultant output determines the next state of the test case t and so its future behaviour. As a result, since the specification is deterministic, for any (more general) such test case t we have only one allowed input sequence $\bar{x}$: the input

- R. M. Hierons is with the Department of Computer Science, Brunel University, UK.
E-mail: rob.hierons@brunel.ac.uk

sequence that results from applying t to the specification. Further, the SUT fails test case t if and only if it fails $\bar{x}$. Since the focus of this paper is checking whether the SUT conforms to the specification, no additional value is provided by using such more general test cases.

This paper adapts state counting to testing from a multi-port deterministic FSM M. We say that test suite $\mathcal{T}$ is c_m-complete if the test cases in $\mathcal{T}$ are controllable for M and for every FSM N with the same sets of ports, inputs and outputs as M, if N has no more than m states and can be distinguished from M using a controllable input sequence then N fails $\mathcal{T}$. This differs from the normal notion of a test suite being m-complete by requiring that testing achieves *as much as possible* while being controllable. The restriction to controllable test cases is often desirable since it avoids races leading to non-determinism in testing (as will be explained in greater detail in Section 2) and the testers know the order in which inputs were received in testing, simplifying debugging and aiding traceability between test cases and parts of models. Most methods for generating test suites from FSMs aim to return controllable test cases (see, for example, [6], [21], [22], [23], [24], [25], [26]). In addition, determining whether FSM N can be distinguished from FSM M in distributed testing is undecidable [27] and so there is no general method for producing an m-complete test suite for distributed testing.

This work is relevant whenever there is a need to test a system that has physically distributed interfaces and either it is not possible to synchronise the testers or this is undesirable (see Section 2). The work in this area initially concerned protocol conformance testing and here we have two interfaces: an upper tester that acts as the layer above the SUT (uses features of the SUT) and a lower tester that is on a separate machine. There may also be timeouts that make it impossible to synchronise testing through the testers exchanging messages. Web services provide another application domain and here many different participants may be involved in a scenario. Similar issues are encountered with online games, though here the interaction is likely to involve real-time constraints that make it even more difficult to synchronise the testers. The growing interest in cloud systems is likely to increase the importance of this topic, as are developments in wireless sensor networks.

Much of the MBT work in distributed testing has concerned testing from a multi-port deterministic FSM (see, for example, [3], [4], [6], [11], [21], [24], [25], [26], [28], [29], [30]). Under this formalism a transition is triggered by a single input but may send output to more than one port. The focus has largely been on protocol conformance testing and has used a variety of protocols as case studies, with these including X.25 DTE [11], the ISO class 0 transport protocol [11], the ISO class 4 transport protocol [25], the ISDN Q.931 network protocol [21], and the quorum protocol [24]. Similar formalisations have also been used for train control systems [31]. However, FSMs have been used in a much wider range of scenarios such as automotive systems [32] and so it seems likely that the approach is more widely applicable. The interest in FSMs has been partially motivated by the fact that specification languages such as SDL, Estelle, and Statecharts can be represented in terms of *extended FSMs*: FSMs with data added. It is then often possible to apply FSM based test techniques by either expanding out the data or abstracting away the data (see, for example, [33]).

This paper makes the following contributions. First, it defines the notion of a test suite $\mathcal{T}$ being c_m-complete for an FSM M. It then proves that the problem of generating a c_m-complete test suite for an FSM M can be mapped to the problem of generating an m-complete test suite for a partial (single-port) FSM $\chi_{min}(M)$. Thus, techniques for generating m-complete test suites for partial FSMs can be adapted. Most approaches for generating an m-complete test suite from a partial FSM are based on state counting and here it is desirable to find maximal sets of states that are pairwise distinguishable. We prove that this problem is NP-complete for distributed testing and also testing from a partial FSM. Finally, we adapt state counting for use in controllable distributed testing. While the focus of the paper is on distributed testing, some of the results have consequences for testing that is not distributed.

The paper is structured as follows. Section 2 describes related work and Section 3 defines FSMs, associated terminology and notation. Section 4 discusses the problem of finding controllable test cases to reach states and explains how $\chi_{min}(M)$ can be generated. Section 5 discusses the problem of distinguishing states in distributed testing and defines the notion of a test suite being c_m-complete. In Section 6 we prove that the problem of finding a largest set of pairwise distinguishable states is NP-complete for distributed testing and testing from a partial FSM. Section 7 then shows how state counting can be used to generate a c_m-complete test suite. Finally, we conclude and discuss potential future work.

2 RELATED WORK

Interest in testing in the distributed test architecture goes back to work on protocol conformance testing [3], [4], [6], [11], [28], [34] (Section 1 outlines some previous cases studies in this area). This modelled the specification as an FSM, where a transition is triggered by a single input but can lead to outputs at more than one port. The initial work showed that distributed testing can lead to additional controllability problems, where a tester does not know when to supply an input [3], [11]. Let us suppose, for example, that the tester at port 1 should send input x_1, it is expected that the SUT will respond by sending output y_1 to port 1, and then the tester at port 2 should send x_2. This scenario is shown in Figure 1 in which vertical lines represent processes, time progresses as we move down, and arcs represent messages. The tester at port 2 does not know when to send its input since it does not observe the previous input and output.

Distributed testing can also lead to observability problems: the behaviours of the SUT and the specification are

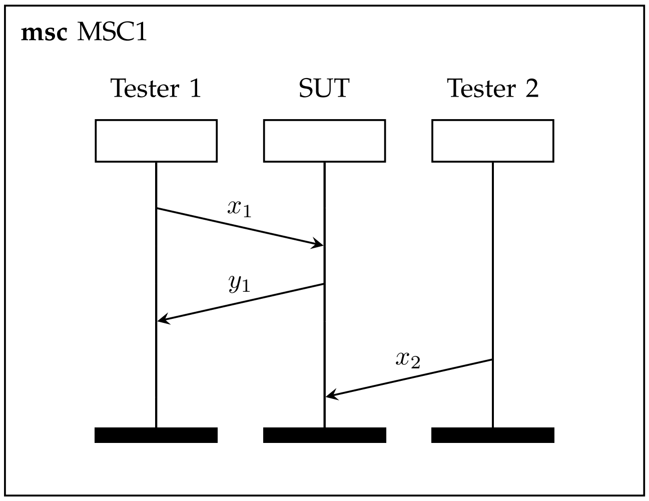

Fig. 1. A controllability problem

different but no tester observes the difference [4]. Let us suppose, for example, that the tester at port 1 sends input x_1, this should lead to output y_1 at port 1 and y_2 at port 2, the tester at port 1 then sends x_1 and this should lead to y_1 at port 1. The observations are $x_1y_1x_1y_1$ at port 1 and y_2 at port 2. This is also the case if y_2 was produced in response to the second input instead of the first. These scenarios are shown in Figure 2.

There has been interest in approaches that choose test cases that cause no controllability problems [21], [22], [23], [24], [25], [26]. However, it is straightforward to construct an FSM M where there are parts of M that cannot be covered by any controllable test case. As a result, methods that use controllable test cases to test whether the SUT is equivalent to the specification (the normal notion of conformance for deterministic FSMs) lack generality. The conditions that allow controllability problems to be overcome also appear not to correspond to simple features of the SUT, with the exception of the case where all transitions send output to all ports. In this paper we apply a different approach, which is to test *as much as possible* given the constraint that test cases are controllable. Thus, FSM N that models a potential SUT conforms to M if and only if N and M produce the same output sequence for every test case that is controllable for M. This corresponds to the previously defined notion of local synch-conformance [6]. This appears to be the first paper to consider testing for local synch-conformance and introduces the notion of a test suite being c_m-complete. Interestingly, given a multi-port FSM M we can construct an NFSM $\chi_{max}(M)$ in polynomial time such that $\chi_{max}(M)$ defines the set of traces of FSMs that cannot be distinguished from M in controllable testing [35]. Thus, the traces of $\chi_{max}(M)$ that are not traces of M are exactly those that an SUT might have despite passing all controllable test cases. It is thus possible to reason about the effectiveness of controllable testing on the basis of $\chi_{max}(M)$ and determine whether

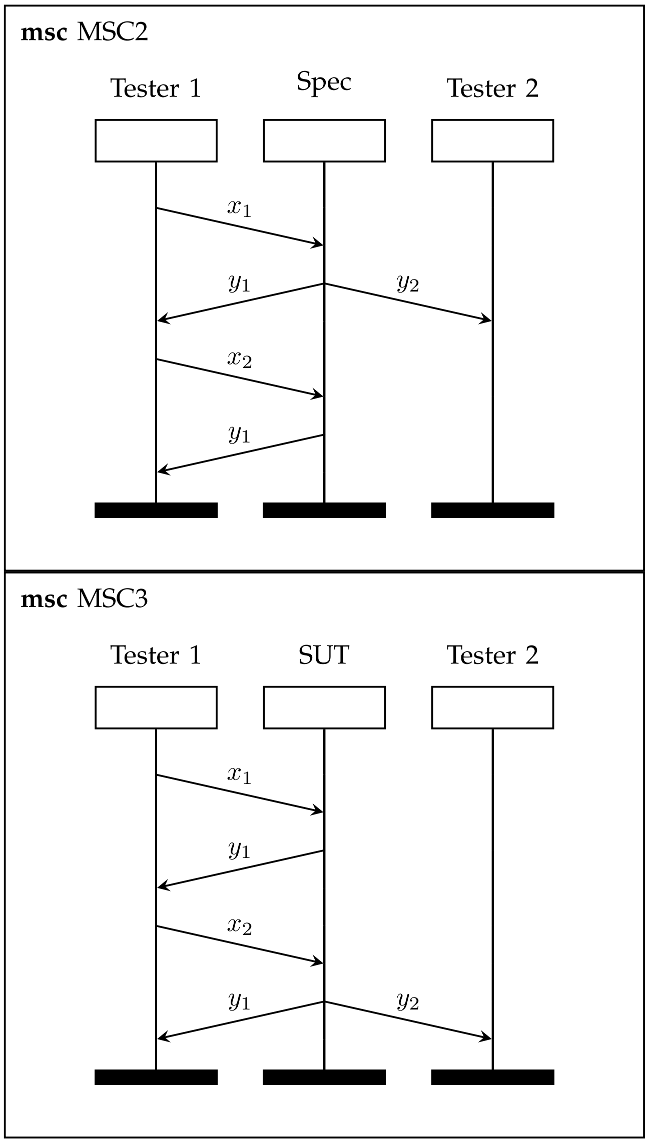

Fig. 2. Observationally equivalent scenarios

controllable testing is suitable.

It is sometimes possible to synchronise testers through the exchange of coordination messages [21], [30], [36]; it is then possible to add messages that overcome controllability problems. For example, if x_i is supplied by the tester at p and then x_{i+1} is to be supplied by the tester at $q \neq p$ then a corresponding controllability problem can be resolved by the tester at p sending a message to the tester at q after it supplies x_i. Similarly, it is possible to overcome observability problems. This led to interest in the problems of minimising the number of coordination messages required [37], [38] and also minimising the number of channels between testers [39], [40]. However, the exchange of coordination messages can increase the cost of testing through testing taking longer and requiring an additional network infrastructure to be built. It also may not be feasible if there are timing constraints. In addition, if message exchange uses the same network as the SUT then message exchange can change the behaviour of the SUT and testing can lead to false positives or false negatives.

Most work on distributed testing has focussed on testing from a multi-port FSM. However, implementation relations have been defined for distributed testing from an input/output transition system (IOTS) [14], [15]. Two types of model have been considered: those where each transition is labelled with a single input or output; and those where a transition is labelled with either an input or a tuple of outputs (at most one per port). Thus, IOTSs are similar to FSMs except that input and output need not alternate and the states set, input alphabet, and output alphabet need not be finite. There appears to be no work that looks at the problem of generating a test suite with guaranteed fault detection power for distributed testing from an IOTS.

The FSM and IOTS models are sequential in nature and capture the distributed nature of testing through using a suitable implementation relation. In contrast, there is work that uses models (Partial Order Automata) in which a transition is labelled by a partial order on inputs and outputs [41], [42]. There has also been work on distributed testing from Petri Nets [43]. Both of these approaches capture the distributed nature of a system through *true concurrency* in the model. This contrasts with most other formalisms in which concurrency is modelled through either synchronisation on events or interleaving of transitions. The potential benefit of using true concurrency in the model is that it can provide a compact description of a highly concurrent system. However, the potential disadvantage is that the formalisms are quite different from those typically used by developers. This paper concerns testing from an FSM but it would be interesting to further explore testing from Partial Order Automata or Petri Nets.

Issues similar to controllability have been explored in the context of message sequence charts (MSCs). An MSC model contains a set of basic MSCs, each defining a scenario in which a set of agents interact. It is typically assumed that an agent can only observe the events in which it is involved (sending and receiving messages) and so can only decide on a next action on the basis of such observations (the local choice assumption). This has led to the notion of a non-local choice: an MSC that breaks this local choice assumption [44]. Non-local choices and controllability problems are very similar concepts, the difference being that in testing there is a specific architecture in which all communication is between the testers and the SUT. There are also approaches that check whether an MSC design is realisable: whether the automata defined for each process provide the same set of scenarios as the original design [45]. The MSC related work explores similar concepts to those considered in distributed testing but appears not to look at issues that correspond to generating a test suite with a given guaranteed effectiveness.

This paper builds on two main areas. One area is the underlying theory in distributed testing and we use two main results from this. The first result, considered when defining the implementation relation local synch-conformance, shows that it is possible to decide in polynomial time whether there is an input sequence that distinguishes two states when using controllable test cases [6]. The second result shows how, given FSM M, we can define a partial FSM $\chi_{min}(M)$ that models the behaviour of M when given controllable test cases [35]. Note that neither paper investigated test generation. The second area is using state counting to generate test suites from a (single-port) FSM. This was developed for testing from a (single-port) NFSM [8], [9], [12], [19] and then for testing from a partial deterministic (single-port) FSM [10]. In this paper we use a state counting approach to drive test suite generation when testing from a multi-port FSM. This is achieved by proving that an FSM N is a correct implementation of FSM M if and only if N conforms to FSM $\chi_{min}(M)$ under the reduction implementation relation (used for single-port FSMs); we then apply state counting as developed by Petrenko and Yevtushenko [10] and show how properties of $\chi_{min}(M)$ affect this.

3 Preliminaries

In this paper we let X denote the set of inputs and Y denote the set of outputs. Given a set A, A^* denotes the set of finite sequences of elements of A and A^n denotes the set of sequences from A^* that have length n. We let ϵ denote the empty sequence. An element of X^* will be called an input sequence or a test case, depending on the context. Given sequence σ we let $pref(\sigma)$ denote the set of prefixes of σ. Similarly, given set Σ of sequences we let $pref(\Sigma)$ denote the set of prefixes of sequences from Σ: $pref(\Sigma) = \cup_{\sigma \in \Sigma} pref(\sigma)$. A sequence $\sigma = x_1/y_1 \dots x_a/y_a$ in which $x_1, \dots, x_a \in X$ and $y_1, \dots, y_a \in Y$ is a *trace* and $x_1 \dots x_a$ is the *input portion* of σ. If $\bar{x} = x_1 \dots x_a$ and $\bar{y} = y_1 \dots y_a$ then $\bar{x}/\bar{y}$ represents the trace $x_1/y_1 \dots x_a/y_a$.

Definition 1: A deterministic multi-port FSM is defined by a tuple $(P, S, s_0, X, Y, \delta, \lambda)$ in which

1) $P = \{1, \dots, k\}$ is the finite set of ports.
2) S is the finite set of states and $s_0 \in S$ is the initial state.
3) X is the finite input alphabet, which is partitioned into $X_1, \dots, X_k$ where X_p is the set of inputs that can be received at port p ($1 \le p \le k$).
4) Y is the finite output alphabet, where each element of Y is a member of $(Y_1 \cup \{-\}) \times \dots \times (Y_k \cup \{-\})$ with Y_p being the set of outputs that can be observed at port p ($1 \le p \le k$) and $-$ denoting no output being observed. We assume that the Y_i are pairwise disjoint and are also disjoint from the X_j.
5) δ is the (possibly partial) state transfer function of type $S \times X \to S$.
6) λ is the (possibly partial) output function of type $S \times X \to Y$ and is defined on the same set of tuples as δ.

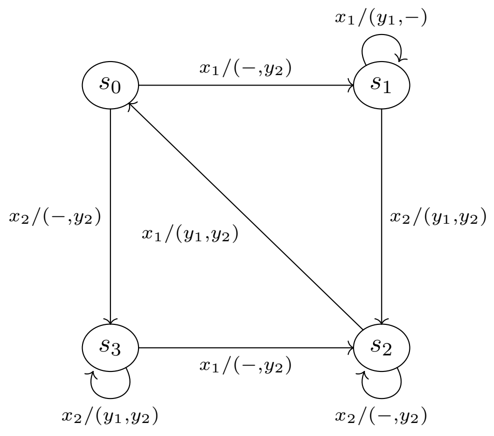

Fig. 3. Finite State Machine M_0

If M receives input x when in state s then it moves to state $s' = \delta(s,x)$ and produces output $y = \lambda(s,x)$ (if these are defined). This defines the *transition* $(s, s', x/y)$.

Throughout this paper we use the term FSM to denote a deterministic multi-port FSM and use the term single-port FSM for deterministic FSMs that have only one port. Figure 3 gives an FSM with two ports that will be called M_0 and will be used as a running example. Here input x_1 is at port 1 and x_2 is at port 2.

If δ and λ are total functions (they are defined on all pairs in $S \times X$) then M is *completely-specified* and otherwise it is *partial*. Given function f, we will use $dom\ f$ to denote the input domain of f: the set of values on which f is defined (so $dom\ \delta = dom\ \lambda$). We will assume that the specification FSM M provided is completely-specified and that the SUT behaves like an unknown completely-specified FSM N. However, we will define partial FSMs that will be used to reason about testing. Given an FSM M we let $\Omega(M)$ be the set of input sequences on which M is defined and given state s of M we let $\Omega_M(s)$ be the set of input sequences on which M is defined when starting in state s. We therefore have that $\Omega(M) = \Omega_M(s_0)$. More formally, we have the following.

$$\Omega_M(s) = \{\epsilon\} \cup \{x\bar{x} | (s,x) \in dom\ \delta \wedge \bar{x} \in \Omega_M(\delta(s,x))\}$$

We can extend δ and λ to input sequences as follows. The base case is: $\delta(s,\epsilon) = s$ and $\lambda(s,\epsilon) = \epsilon$. The recursive case is: given $s \in S$ and $x\bar{x} \in \Omega_M(s)$ with $x \in X$ and $\bar{x} \in X^*$, $\delta(s, x\bar{x}) = \delta(\delta(s,x),\bar{x})$ and $\lambda(s,x\bar{x}) = \lambda(s,x)\lambda(\delta(s,x),\bar{x})$. For example, in M_0 we have that $\lambda(s_0, x_1x_2) = (-,y_2)(y_1,y_2)$ and $\delta(s_0,x_1x_2) = s_2$. If $\bar{x} \in \Omega(M)$ then $\bar{x}/\lambda(s_0,\bar{x})$ is a trace of M and we let $L(M)$ denote the set of traces of M. Given state s of M we will let $L_M(s)$ denote the set of traces of the FSM formed by making s the initial state of M.

State s is *reachable* if there is an input sequence $\bar{x}$ that takes M from s_0 to s; $\bar{x}$ *reaches* s. Thus, $\bar{x}$ reaches s if and only if $\bar{x} \in \Omega(M)$ and $s = \delta(s_0,\bar{x})$. An FSM is *initially connected* if all of its states are reachable. Sequence $\bar{\rho} = (s_1,s_2,x_1/y_1)(s_2,s_3,x_2/y_2)\ldots(s_a,s_{a+1},x_a/y_a)$ of consecutive transitions is said to be a *path*. Further, $label(\bar{\rho}) = x_1/y_1 \ldots x_a/y_a$ is the *label* of $\bar{\rho}$ and $x_1 \ldots x_a$ is the *input portion* of $x_1/y_1 \ldots x_a/y_a$. For example, $(s_0,s_1,x_1/(-,y_2))(s_1,s_2,x_2/(y_1,y_2))(s_2,s_3,x_1/(-,y_2))$ is a path of M_0 with label $x_1/(-,y_2)x_2/(y_1,y_2)x_1/(-,y_2)$, that has input portion $x_1x_2x_1$.

We will need to reason about the ports at which events (inputs and outputs) occur. Given input x, $port(x)$ denotes the port p such that $x \in X_p$. Given output $y = (y_1,\ldots,y_k)$, $ports(y)$ denotes the set of ports at which output is observed: $ports(y) = \{1 \leq p \leq k | y_p \neq -\}$. Similarly, we let $ports(x/y) = \{port(x)\} \cup ports(y)$. Given a transition $\tau = (s,s',x/y)$ we let $ports(\tau) = ports(x/y)$.

When testing from an FSM M an input sequence $\bar{x} = x_1 \ldots x_a \in \Omega(M)$ is controllable if for all $1 < i \leq a$ the tester that applies x_i observes input and/or output in the previous transition [3], [11]. In such a situation, the tester that supplies x_i waits to observe the expected values and then sends x_i.

Definition 2: When testing from an FSM M, input sequence $\bar{x} = x_1 \ldots x_a \in \Omega(M)$ is *controllable* if $\lambda(s_0,\bar{x}) = y_1 \ldots y_a$ is such that for all $1 < i \leq a$ if $x_i \in X_p$ then $p \in ports(x_{i-1}/y_{i-1})$.

Consider now what can happen if this condition does not hold; the tester at p is to supply input x_i ($i > 1$) but did not observe input or output in the previous input/output pair x_{i-1}/y_{i-1}. The problem here is that the tester at p sends its input after some earlier observations at p but cannot know when x_{i-1} has been supplied. As a result, the tester might erroneously supply input x_i before x_{i-1} has been sent. An example of this is given in Figure 4. Here the tester at port 2 observes previous input and output but it makes no observations after y_2. Thus, the observations made by tester 2 are not sufficient for it to know when to send x_2': there is a possibility that x_2' will arrive before x_1.

In distributed testing, the tester at p observes only the events at p. Thus, if we define $\pi_p(\sigma)$ to be the projection of trace σ on port p and the SUT produces σ then the tester at p observes $\pi_p(\sigma)$. The projection is defined by the following in which $y = (y_1,\ldots,y_k)$ [6].

$$\begin{aligned}
\pi_p(\epsilon) &= \epsilon \\
\pi_p((x/y)\sigma) &= \pi_p(\sigma) \text{ if } x \notin X_p \wedge y_p = - \\
\pi_p((x/y)\sigma) &= x\pi_p(\sigma) \text{ if } x \in X_p \wedge y_p = - \\
\pi_p((x/y)\sigma) &= y_p\pi_p(\sigma) \text{ if } x \notin X_p \wedge y_p \neq - \\
\pi_p((x/y)\sigma) &= xy_p\pi_p(\sigma) \text{ if } x \in X_p \wedge y_p \neq -
\end{aligned}$$

Two traces are observationally equivalent if they lead to the same observation at each port. More formally, given traces σ and σ', $\sigma \sim \sigma'$ if for all $p \in P$ we have that $\pi_p(\sigma) = \pi_p(\sigma')$. For example, if we let $\sigma = x_1/(y_1,-)x_1/(-,y_2)$ and $\sigma' = x_1/(y_1,y_2)x_1/(-,-)$ then $\pi_1(\sigma) = \pi_1(\sigma') = x_1y_1x_1$ and $\pi_2(\sigma) = \pi_2(\sigma') = y_2$ and so $x_1/(y_1,-)x_1/(-,y_2) \sim x_1/(y_1,y_2)x_1/(-,-)$.

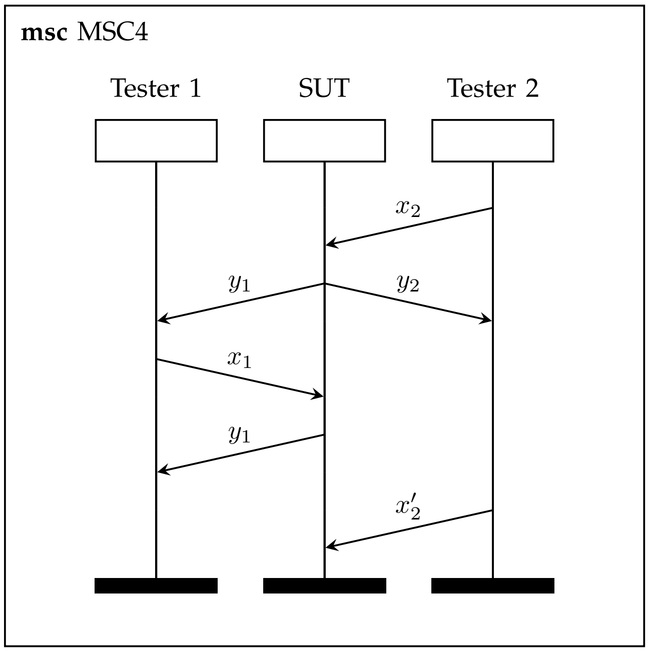

Fig. 4. A controllability problem despite previous observations

We now define terminology used with partial FSMs [10], [46]. Two states s_i and s_j of M are equivalent if the same sets of input sequence are defined from them and the corresponding outputs are identical.

Definition 3: States s_i and s_j of FSM M are *equivalent* if $\Omega_M(s_i) = \Omega_M(s_j)$ and for all $\bar{x} \in \Omega_M(s_i)$ we have that $\lambda(s_i, \bar{x}) = \lambda(s_j, \bar{x})$. Two FSMs are equivalent if and only if their initial states are equivalent.

Under quasi-equivalence states can have different sets of possible input sequences: s_i is quasi-equivalence to s_j if all input sequence defined from s_j are also defined from s_i and the corresponding outputs are identical.

Definition 4: State s_i is *quasi-equivalent* to state s_j ($s_j \sqsubseteq s_i$) if $\Omega_M(s_j) \subseteq \Omega_M(s_i)$ and for all $\bar{x} \in \Omega_M(s_j)$ we have that $\lambda(s_i, \bar{x}) = \lambda(s_j, \bar{x})$. Given FSMs M and N with initial states s_0^M and s_0^N respectively, N is quasi-equivalent to M if and only if s_0^N is quasi-equivalent to s_0^M.

Note that $\sqsubseteq$ is a partial order and $s_j \sqsubseteq s_i$ if and only if $L_M(s_j) \subseteq L_M(s_i)$. When comparing FSMs N and M where M is the specification and N is a possible behaviour of the SUT, the notion of N being quasi-equivalent to M allows M to be partial and for the SUT N to have any behaviour where M is not specified.

An FSM M is *minimal* if no FSM with fewer states than M is equivalent to M. If an FSM is not initially connected then unreachable states can be removed and so minimal FSMs are initially connected. In this paper we assume that the specification FSM M and the unknown FSM N that represents the behaviour of the SUT are minimal and completely specified. The restriction to completely specified FSMs is relatively common but extending the method to partially specified FSMs is a potentially interesting line of future work. Any completely specified FSM is equivalent to a minimal completely specified FSM, which can be generated in low order polynomial time [47], and so assuming that M and N are minimal is not restrictive. This assumption, that M and N are minimal, is made in order to simplify the exposition.

Since quasi-equivalence is a partial order, we require notation regarding partially ordered sets. A partially ordered set is defined by a pair $(A, \leq)$, where $\leq$ is a partial order on set A ($\leq$ is reflexive, transitive and anti-symmetric). For partially ordered set $(A, \leq)$, $A' \subseteq A$ is a *chain* if there is an order $a_1, \ldots, a_i$ of the elements of A' such that $a_1 \leq a_2, \ldots, a_{i-1} \leq a_i$. $A' \subseteq A$ is an *anti-chain* if no two distinct elements of A' are related under $\leq$.

4 Reaching states in controllable testing

This section discusses the problem of finding controllable input sequences that reach particular states of the specification. The approach described is based on work [35] that has shown how, given an FSM M, we can define a partial FSM $\chi_{min}(M)$ that models the behaviour of M when given controllable test cases.

For each state $s_i \in S$ and port $p \in P$, $Depart^p(s_i) = \{(s_i, s_j, x/y) | x \in X_p, y = \lambda(s_i, x), s_j = \delta(s_i, x)\}$ is the set of transitions of M with starting state s_i whose input is at p [22]. For example, in M_0 we have that $Depart^1(s_2) = \{(s_2, s_0, x_1/(y_1, y_2))\}$. Similarly, given $\mathcal{P} \subseteq P$, $Arrive^{\mathcal{P}}(s_i) = \{(s_j, s_i, x/y) | ports(x/y) = \mathcal{P}, y = \lambda(s_i, x), s_i = \delta(s_j, x)\}$ is the set of transitions of M with ending state s_i that involve the set $\mathcal{P}$ of ports. For example, in M_0 we have that $Arrive^{\{1,2\}}(s_2) = \{(s_1, s_2, x_2/(y_1, y_2)), (s_3, s_2, x_1/(-, y_2))\}$. We have the following consequence of these definitions.

Proposition 1: A transition τ from $Arrive^{\mathcal{P}}(s_i)$ can only be followed by input x in controllable testing if $x \in X_p$ for some $p \in \mathcal{P}$.

As a result, $\tau \in Arrive^{\mathcal{P}}(s_i)$ can be followed by τ' if and only if $\tau' \in Depart^p(s_i)$ for some $p \in \mathcal{P}$.

We can now define the partial FSM $\chi_{min}(M) = (S', s'_0, X, Y, \delta', \lambda')$ [35]. Let $S = \{s_1, \ldots, s_n\}$. For each $s_i \in S$ and $\mathcal{P} \subseteq P$ there can be vertex $s_i^{\mathcal{P}}$ representing the situation in which the state is s_i and the next input must be at a port in $\mathcal{P}$. The set S' of states of $\chi_{min}(M)$ is defined by the following.

1) For all $1 \leq i \leq n$ and $\mathcal{P} \subseteq P$, $s_i^{\mathcal{P}} \in S'$ if and only if $Arrive^{\mathcal{P}}(s_i) \neq \emptyset$.
2) State s_0^P is in S' and $s'_0 = s_0^P$ is the initial state of $\chi_{min}(M)$.

The state s_0^P represents the situation before testing starts: since no inputs have been applied the first input can be applied at any port without causing a controllability problem. Given state $s_i^{\mathcal{P}}$ we let δ' and λ' be defined on $(s_i^{\mathcal{P}}, x)$ if and only if $x \in X_p$ for some $p \in \mathcal{P}$. Given such $(s_i^{\mathcal{P}}, x)$ we let $\lambda'(s_i^{\mathcal{P}}, x) = \lambda(s_i, x)$ and $\delta'(s_i^{\mathcal{P}}, x) = s_j^{\mathcal{P}'}$ for the state $s_j^{\mathcal{P}'}$ such that $s_j = \delta(s_i, x)$ and $\mathcal{P}' = ports(x/\lambda(s_i, x))$. The above FSM need not be

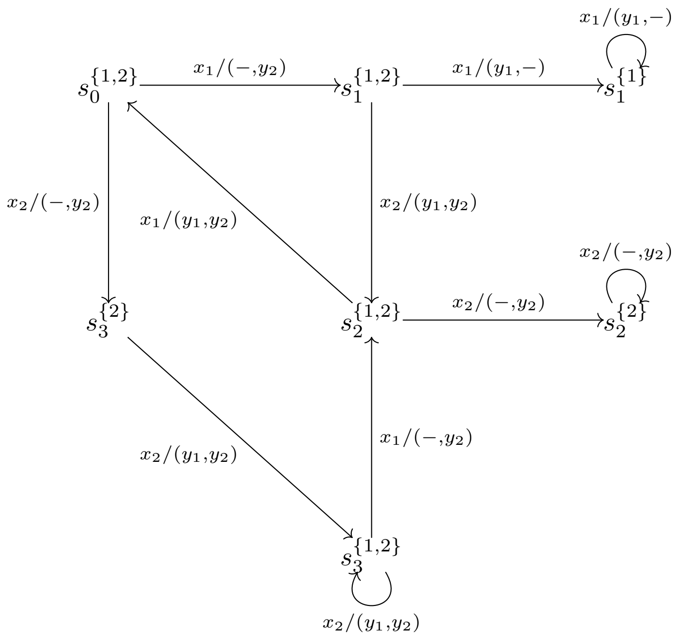

Fig. 5. Partial FSM $\chi_{min}(M_0)$

initially connected so we let $\chi_{min}(M)$ denote the corresponding FSM in which all unreachable states have been removed. It is straightforward to see that the number of states of $\chi_{min}(M)$ is bounded above by the number of transitions of M plus 1. Figure 5 gives $\chi_{min}(M_0)$.

The following results are known [35] and use the notion of a path being controllable, which is the case if and only if its label is controllable.

Proposition 2: For each controllable path $\bar{\rho}$ in M that starts at s_0, there is a unique path $\bar{\rho}'$ in $\chi_{min}(M)$ that starts at s_0^P such that $label(\bar{\rho}) = label(\bar{\rho}')$.

Proposition 3: For each path $\bar{\rho}'$ in $\chi_{min}(M)$ that starts at s_0^P, there is a unique controllable path $\bar{\rho}$ in M that starts at s_0 such that $label(\bar{\rho}) = label(\bar{\rho}')$.

Given FSM M, $C(M)$ will denote the set of input sequences that are controllable for M: these are the input portions of the labels of paths of $\chi_{min}(M)$ that start at the initial state. Thus, in controllable testing we use input sequences from $C(M)$. The following is clear.

Proposition 4: Given FSM M we have that $C(M) = \Omega(\chi_{min}(M))$.

Note that in M_0, for every state s_i the state $s_i^{\{1,2\}}$ is reachable. Thus, every transition of M_0 can be included in controllable test cases and so the impact of restricting testing to controllable test cases will be limited. However, it is straightforward to construct examples in which controllable testing can achieve much less. Consider, for example the FSM M_0' shown in Figure 6. This differs from M_0 only in the outputs of the transition from s_0 to s_1 and the self-loop transition in state s_3. If a controllable input sequence for M_0' starts with input x_1 then it takes M_0' from s_0 to s_1 and leads to output y_1 at port 1 only. Observations were only made at port 1 and so the only

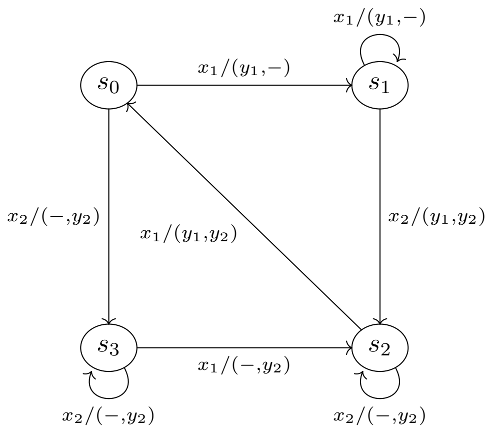

Fig. 6. Finite State Machine M_0'

transition then possible in controllable testing is the self-loop transition in s_1 with input x_1 and output y_1. It is clear that controllable testing is then 'stuck' in state s_1: any controllable test case that starts with x_1 is of the form x_1^k for some k. Similarly, any controllable test case that starts with x_2 is of the form x_2^k for some k. Thus, no controllable test case for M_0' contains both x_1 and x_2 and only four transitions of M_0' can be executed in controllable testing. It should also clear that we preserve this property if, for example, we change the self-loop transition in s_2 so that it takes M_0' to some new state s_4. Thus, we can make M_0' arbitrarily large while preserving this property (there are only four transitions that can be executed in controllable testing). There is a need for research that looks at classes of real systems and explores what can be achieved using controllable test cases.

5 Distinguishing states and FSMs

This section explores the problem of distinguishing states or FSMs in controllable distributed testing and defines the notion of a c_m-complete test suite. The test suite generation algorithm will utilise sets of states that can be distinguished in controllable testing, along with input sequences that reach states (discussed in the previous section). The following defines the condition under which an input sequence distinguishes two states of M in controllable testing; it requires that the input sequence causes no controllability problems and a tester observes a difference [6]. In the following, $M(s)$ denotes the FSM formed from M by making s its initial state.

Definition 5: Input sequence $\bar{x}$ *locally synch-distinguishes* states s_1 and s_2 of M *at port* $p \in P$ if $\bar{x}$ is controllable from both s_1 and s_2 ($\bar{x} \in C(M(s_1)) \cap C(M(s_2))$) and $\pi_p(\bar{x}/\lambda(s_1, \bar{x})) \neq \pi_p(\bar{x}/\lambda(s_2, \bar{x}))$. Further, $\bar{x}$ *locally synch-distinguishes* states s_1 and s_2 of M if $\bar{x}$ locally synch-distinguishes s_1 and s_2 at p for some $p \in P$.

Consider again M_0. Here x_2 locally synch-distinguishes s_0 and s_3 (at port 1) since $\lambda(s_0, x_2) = (-, y_2)$ and $\lambda(s_3, x_2) = (y_1, y_2)$. However,

x_1 does not locally synch-distinguish s_0 and s_3 since $\lambda(s_0, x_1) = (-, y_2)$ and $\lambda(s_3, x_1) = (-, y_2)$.

The above leads to an implementation relation (notion of correctness) for controllable distributed testing.

Definition 6: Given FSMs M and N with the same input and output alphabets and the same set of ports, N locally synch-conforms to M if for every $\bar{x} \in C(M)$, $\bar{x}$ does not locally synch-distinguish N and M. Further, N locally synch-conforms to M on input sequence $\bar{x}$ if $\bar{x}$ is controllable for M and N and does not locally synch-distinguish N and M.

Since we are interested in controllable testing we want to only use test cases that are controllable for M. In addition, we only need to distinguish an FSM N, that models a possible SUT, from M if N does not conform to M in controllable testing; if it is possible to distinguish N from M in controllable testing. We now define the notion of a c_m-complete test suite.

Definition 7: Given FSM M and integer m, a test suite $\mathcal{T}$ is *c_m-complete* for M if the following conditions hold.

- All elements of $\mathcal{T}$ are controllable when applied from the initial state of M ($\mathcal{T} \subseteq C(M)$); and
- For every FSM N with the same sets of ports, inputs and outputs as M and no more than m states, if N does not locally synch-conform to M then some $t \in \mathcal{T}$ locally synch-distinguishes N from M.

A c_m-complete test suite can contain multiple test cases. In practice there may be a need to reset the SUT between the application of different test cases and in this paper we assume that there is a *reliable reset*: a process that is known to correctly reset the SUT [2], [8], [9], [10]. For some systems this is simply switching the SUT off and then on again but the reset might be much more involved. There is also a need to move to a situation in which the testers are synchronised before the next test case is applied: they are all aware that a new test case is to begin[1]. One possible approach to synchronisation is to allow the testers to communication with one another between tests. Another is to introduce a sufficiently long delay. The method used to achieve such synchronisation is likely to depend on the setup for testing and will not be explored further.

The following from [6] shows that if there is an input sequence that leads to different output sequences from two states and does not cause controllability problems from these states then there is an input sequence that locally synch-distinguishes the states.

Proposition 5: Let us suppose that $\bar{x}$ is controllable from states s_1 and s_2 of M ($\bar{x} \in C(M(s_1)) \cap C(M(s_2))$) and $\lambda(s_1, \bar{x}) \neq \lambda(s_2, \bar{x})$. If $\bar{x}_1$ is a minimal prefix of $\bar{x}$ such that $\lambda(s_1, \bar{x}_1) \neq \lambda(s_2, \bar{x}_1)$ then $\bar{x}_1$ locally synch-distinguishes s_1 and s_2.

An important consequence of this is that if we include all prefixes of each input sequence used then we do not have to consider possible observability problems when distinguishing states. This will allow us to reason about test cases that distinguish states, and so FSMs, in terms of the traces being different (as opposed to the set of projections of the traces being different).

A polynomial upper bound on the length of a minimal input sequence that locally synch-distinguishes two states has been given as has an $O(kn^2)$ time algorithm for finding such sequences [6].

Theorem 1: Let M denote an FSM with n states and k ports. Given states s_1, s_2 of M and port $p \in P$, if s_1 and s_2 are locally synch-distinguished by an input sequence starting with an element of X_p then they are locally synch-distinguished by an input sequence of length at most $k(n-1)$ that starts with an element of X_p.

The following shows how the notion of locally synch-distinguishing relates to definitions regarding partial FSMs and also shows that conformance is actually an equivalence relation. Importantly, this shows that methods for testing from a partial single-port FSM can be applied in testing from an FSM.

Theorem 2: Given FSMs $M = (P, S_M, s_0^M, X, Y, \delta_M, \lambda_M)$ and $N = (P, S_N, s_0^N, X, Y, \delta_N, \lambda_N)$ with the same sets of ports and input and output alphabets, N locally synch-conforms to M if and only if $\chi_{min}(N)$ and $\chi_{min}(M)$ are equivalent.

Proof: First let us suppose that N locally synch-conforms to M and we need to prove that $\chi_{min}(N)$ and $\chi_{min}(M)$ are equivalent. Proof by contradiction: assume that $\chi_{min}(N)$ and $\chi_{min}(M)$ are not equivalent. Thus, there is an input sequence $\bar{x}$ such that either $\bar{x}$ is in one of $\Omega(\chi_{min}(N))$ and $\Omega(\chi_{min}(N))$ but not the other or $\bar{x} \in \Omega(\chi_{min}(N)) \cap \Omega(\chi_{min}(N))$ and $\chi_{min}(N)$ and $\chi_{min}(M)$ produce different output sequences when given $\bar{x}$. Let us suppose that $\bar{x}$ is a minimal such input sequence and so $\bar{x} = \bar{x}'x$ for some $x \in X$ and $\bar{x}' \in X^*$. By the minimality of $\bar{x}$, $\bar{x}' \in \Omega(\chi_{min}(N)) \cap \Omega(\chi_{min}(N))$ and so the application of $\bar{x}'$ causes no controllability problems in N and M. Further, $\chi_{min}(N)$ and $\chi_{min}(M)$ produce the same output when given $\bar{x}'$ (we have that $\lambda_N(s_0^N, \bar{x}') = \lambda_M(s_0^M, \bar{x}')$). Thus, x can be applied after $\bar{x}'$ in N without causing a controllability problem if and only if x can be applied after $\bar{x}'$ in M without causing a controllability problem. We therefore know that $\bar{x} \in \Omega(\chi_{min}(N)) \cap \Omega(\chi_{min}(N))$ and so, by the definition of $\bar{x}$, $\lambda_N(s_0^N, \bar{x}) \neq \lambda_M(s_0^M, \bar{x})$. This contradicts N local synch-conforming to M as required.

Now assume that $\chi_{min}(N)$ and $\chi_{min}(M)$ are equivalent and we need to prove that N locally synch-conforms to M. Again we will use proof by contradiction, assuming that $\chi_{min}(N)$ and $\chi_{min}(M)$ are equivalent and that N does not locally synch-conform to M. Since N does not locally synch-conform to M and $C(M) = C(N)$ (since $\chi_{min}(N)$ and $\chi_{min}(M)$ are equivalent) there is an input sequence $\bar{x}$ that is controllable from the initial states of N and M and that leads to different output sequences when applied in s_0^M and s_0^N. Let $\bar{y}_M = \lambda_M(s_0^M, \bar{x})$ and $\bar{y}_N = \lambda_N(s_0^N, \bar{x})$. Since $\bar{x}$ is controllable from the initial states of N and M, it is the input portion of labels of

1. Since we are using controllable test cases the local testers do not need to synchronise their local clocks.

paths from the initial states of $\chi_{min}(N)$ and $\chi_{min}(M)$ and these paths have labels $\bar{x}/\bar{y}_N$ and $\bar{x}/\bar{y}_M$ respectively. Since $\bar{y}_N \neq \bar{y}_M$, $\bar{x}$ distinguishes the initial states of $\chi_{min}(N)$ and $\chi_{min}(M)$. This contradicts $\chi_{min}(N)$ and $\chi_{min}(M)$ being equivalent as required. □

We thus know that local synch-conformance for FSMs can be expressed in terms of equivalence of partial FSMs. Observe also that all test cases generated from $\chi_{min}(M)$ are controllable and that by using prefixes of test cases we avoid observability problems. Thus, we can treat $\chi_{min}(M)$ as a single-port FSM and use *any* method for testing from a partial single-port FSM to generate c_m-complete test suites. This is the key result of the paper; the rest of the paper adapts such a method for testing from a partial single-port FSM.

We will find that $\chi_{min}(M)$ has some properties that need not hold more generally for partial FSMs. As an example, if two states of a partial FSM with a states are distinguishable then there is an input sequence of length at most $a(a-1)/2$ that distinguishes them. We have that $\chi_{min}(M)$ has $O(n|X|)$ states (at most one per transition of M plus the initial state) and so this result suggests that to distinguish states of $\chi_{min}(M)$ we need input sequences of $O(n^2|X|^2)$ length. In contrast, we have an upper bound of $k(n-1)$ for distributed testing from M [6]. The following also shows that there is no need to differentiate between the concepts of FSMs being equivalent and being quasi-equivalent. Note, however, that we will still have to consider quasi-equivalence when reasoning about states of an FSM.

Proposition 6: Given FSMs M and N, $\chi_{min}(M)$ and $\chi_{min}(N)$ are equivalent if and only if $\chi_{min}(N)$ is quasi-equivalent to $\chi_{min}(M)$.

Proof: First, if $\chi_{min}(N)$ and $\chi_{min}(M)$ are equivalent then $\chi_{min}(N)$ being quasi-equivalent to $\chi_{min}(M)$ follows immediately from the definition.

Now let us suppose that $\chi_{min}(N)$ is quasi-equivalent to $\chi_{min}(M)$. By definition, it is sufficient to prove that $\chi_{min}(N)$ and $\chi_{min}(M)$ are defined on the same sets of input sequences ($\Omega(\chi_{min}(M)) = \Omega(\chi_{min}(N))$). Proof by contradiction: assume that $\Omega(\chi_{min}(M)) \neq \Omega(\chi_{min}(N))$ and let $\bar{x} = \bar{x}'x$ be a shortest input sequence that is in one of $\Omega(\chi_{min}(N))$ and $\Omega(\chi_{min}(M))$ but not both ($x \in X$). Since $\chi_{min}(N)$ is quasi-equivalent to $\chi_{min}(M)$ we must have that $\Omega(\chi_{min}(M)) \subseteq \Omega(\chi_{min}(N))$ and so $\bar{x} \in \Omega(\chi_{min}(N)) \setminus \Omega(\chi_{min}(M))$. However, this means that $\bar{x}'$ is a controllable input sequence from the initial states of M and N and $\bar{x}'$ can be followed by x in N but not in M. This implies that $\chi_{min}(M)$ and $\chi_{min}(N)$ produce different output sequences on $\bar{x}'$. This contradicts $\chi_{min}(M)$ and $\chi_{min}(N)$ being quasi-equivalent as required. □

6 Checking states of the SUT

The previous section explored conditions under which two states can be distinguished in controllable testing. In Section 7 we will see that state counting uses sets of states that are pairwise distinguishable and in this section we explore properties of such sets of states.

Let us suppose that input sequences $\bar{v}_1$ and $\bar{v}_2$ are controllable from the initial state of M and reach states s_1 and s_2 of M respectively. In checking states we would like to follow $\bar{v}_1$ and $\bar{v}_2$ by an input sequence $\bar{x}$ that locally synch-distinguishes s_1 and s_2 such that $\bar{x}$ can be applied after $\bar{v}_1$ and $\bar{v}_2$ without causing any controllability problems. This leads to the following.

Definition 8: Given input sequences $\bar{v}_1$, $\bar{v}_2$ and $\bar{x}$, $(\bar{v}_1, \bar{v}_2, \bar{x})$ is a *separating tuple* for M if the following hold.

- Input sequences $\bar{v}_1\bar{x}$ and $\bar{v}_2\bar{x}$ are controllable from the initial state of M ($\bar{v}_1\bar{x}, \bar{v}_2\bar{x} \in C(M)$).
- The states s_1 and s_2 of M reached by $\bar{v}_1$ and $\bar{v}_2$ are locally synch-distinguished by $\bar{x}$.

Clearly, if $(\bar{v}_1, \bar{v}_2, \bar{x})$ is a separating tuple for M then so is $(\bar{v}_2, \bar{v}_1, \bar{x})$. In test generation we will use sets of input sequences that reach pairwise distinguishable states of $\chi_{min}(M)$. We therefore introduce the following.

Definition 9: Given FSM M, the tuple (V, SP) is a *state identification tuple* for M if the following hold.

- The input sequences in V reach distinct states of $\chi_{min}(M)$.
- SP is a set of separating tuples for M.
- For all $\bar{v}_1, \bar{v}_2 \in V$ there is some $\bar{x} \in X^*$ such that $(\bar{v}_1, \bar{v}_2, \bar{x}) \in SP$.

The idea is that to check states of the SUT we follow the input sequences from V by suitable input sequences defined by SP. Given (V, SP) we can produce the following test suite.

$$T(V, SP) = pref(\{\bar{v}_1\bar{x} | \bar{v}_1 \in V \wedge \exists \bar{v}_2 \in V.(\bar{v}_1, \bar{v}_2, \bar{x}) \in SP\})$$

The following shows that if FSM N locally synch-conforms to M on $T(V, SP)$ then (V, SP) is a state identification tuple for N. There are similar results for testing from a single-port FSM; the key point here is that the use of prefixes overcomes observability problems.

Proposition 7: Let us suppose that (V, SP) is a state identification tuple for FSM M and the FSM N locally synch-conforms to M on each test case in $T(V, SP)$. Then (V, SP) is a state identification tuple for N.

Proof: Let us suppose that $M = (P, S_M, s_0^M, X, Y, \delta_M, \lambda_M)$ and $N = (P, S_N, s_0^N, X, Y, \delta_N, \lambda_N)$. It is sufficient to prove that for all $\bar{x}$ with $(\bar{v}_i, \bar{v}_j, \bar{x}) \in SP$ we have that $\bar{x}$ locally synch-distinguishes the states s_i and s_j of N reached by $\bar{v}_i$ and $\bar{v}_j$ respectively; it is then immediate that $\bar{v}_i$ and $\bar{v}_j$ reach different states of N.

Since $(\bar{v}_i, \bar{v}_j, \bar{x})$ is a separating tuple for M, there is a port $p \in P$ such that $\pi_p(\lambda_M(\delta_M(s_0^M, \bar{v}_i), \bar{x})) \neq \pi_p(\lambda_M(\delta_M(s_0^M, \bar{v}_j), \bar{x}))$. Since N locally synch-conforms to M on each test case in $T(V, SP)$ we have that for every port q, $\pi_q(\lambda_N(s_0^N, \bar{v}_i\bar{x})) = \pi_q(\lambda_M(s_0^M, \bar{v}_i\bar{x}))$ and $\pi_q(\lambda_N(\delta_N(s_0^N, \bar{v}_j), \bar{x})) = \pi_q(\lambda_M(\delta_M(s_0^M, \bar{v}_j), \bar{x}))$. Further, this holds for all prefixes of $\bar{x}$ and so we can deduce that $\lambda_N(\delta_N(s_0^N, \bar{v}_i), \bar{x}) = \lambda_M(\delta_M(s_0^M, \bar{v}_i), \bar{x})$ and $\lambda_N(\delta_N(s_0^N, \bar{v}_j), \bar{x}) = \lambda_M(\delta_M(s_0^M, \bar{v}_j), \bar{x})$. In addition, since $\bar{v}_i\bar{x}$ and $\bar{v}_j\bar{x}$ are controllable from the initial state of M and produce the same trace in M as in N we must have that they are also controllable from the initial state of N.

To conclude, we have that $\bar{v}_i\bar{x}$ and $\bar{v}_j\bar{x}$ are controllable for N and there is a port p such that $\pi_p(\lambda_N(\delta_N(s_0^N, \bar{v}_i), \bar{x})) \neq \pi_p(\lambda_N(\delta_N(s_0^N, \bar{v}_j), \bar{x}))$. Thus, $\bar{x}$ locally synch-distinguishes states s_i and s_j of N reached by $\bar{v}_i$ and $\bar{v}_j$ respectively. The result therefore holds. □

This result is important since it will allow us to know that certain prefixes of a test case reach different states of the SUT if the SUT passes given tests (these prefixes followed by sequences that distinguish states).

We will see that state counting, which is used to drive test generation, takes advantage of sets of pairwise distinguishable states and ideally we want maximal such sets. However, we will show that the problem of finding such a (maximal) state identification tuple is NP-complete (Theorem 3 below). Before proving this we define the maximal clique problem.

Definition 10: Given undirected graph $G = (U, E)$ the *maximal clique problem* is to find a largest set U' of vertices of G such that all vertices in U' are connected in G.

The maximal clique problems is NP-complete [48].

Theorem 3: The problem of finding a largest set of pairwise distinguishable states of $\chi_{min}(M)$ is NP-complete.

Proof: First we prove that the problem is in NP. We will initially consider the following problem: given integer ℓ does $\chi_{min}(M)$ have a set of ℓ states that are pairwise locally synch-distinguishable? We will show that there is a non-deterministic Turing Machine that can solve this problem in polynomial time. The non-deterministic Turing Machine initially guessed a set S' that contains ℓ states of $\chi_{min}(M)$. We know that two states of $\chi_{min}(M)$ can be locally synch-distinguishes if and only if they can be locally synch-distinguishes by an input sequence of length at most $k(n-1)$ [6], where n is the number of states of M and k the number of ports. The non-deterministic Turing Machine randomly generates an input sequence of length at most $k(n-1)$ for each pair of states in S'. If input sequence $\bar{x}$ is guessed for states $s_1^{\mathcal{P}_1}$ and $s_2^{\mathcal{P}_2}$ of $\chi_{min}(M)$ then the Turing Machine checks that $\bar{x}$ is in both $\Omega_{\chi_{min}(M)}(s_1^{\mathcal{P}_1})$ and $\Omega_{\chi_{min}(M)}(s_2^{\mathcal{P}_2})$ and that $\bar{x}$ locally synch-distinguishes $s_1^{\mathcal{P}_1}$ and $s_2^{\mathcal{P}_2}$. Since these checks can be performed in polynomial time, this process takes polynomial time. Thus, given M and ℓ a non-deterministic Turing machine can decide in polynomial time whether $\chi_{min}(M)$ has a set of ℓ states that are pairwise locally synch-distinguishable. Thus, a non-deterministic Turing Machine can initially solve this for ℓ being the number of states of $\chi_{min}(M)$, if there is no solution then it reduces ℓ by 1 and iterates until it finds a largest value of ℓ for which there is a corresponding set of pairwise locally synch-distinguishable states of M. Thus, a non-deterministic Turing Machine can solve the problem in polynomial time and so the problem is in NP.

We now show that the problem is NP-hard and will assume that we have been given a graph $G = (U, E)$, $U = \{u_1, \ldots, u_n\}$, and will construct an FSM M. We will let $P = \{0, 1, \ldots, n\}$, set $S = \{s_0, s_1, \ldots, s_n, s_{n+1}\}$ and will construct M such that for $1 \leq i \leq n$ the state s_i will 'correspond' to vertex u_i.

For each $1 \leq i \leq n$ there is an input x_i at port 0 that takes M from s_0 to s_i and this transition has output y_j at port j ($1 \leq j \leq n, j \neq i$) if and only if there is an edge between u_i and u_j in G. The input of x_i in any other state leads to no change in state and no output.

For each port $1 \leq j \leq n$ there is an input x'_j at port j and this leads to the following transitions.

- From s_0 there is a transition to s_0 with no output.
- From s_i, $1 \leq i \leq n$, if $i \neq j$ then there is a transition to s_{n+1} with no output at port 0 and output y_p at each port $p \neq 0$.
- From s_j there is a transition to s_{n+1} with output j at port 0 and output y_p at each port $p \neq 0$.
- From s_{n+1} there is a transition to s_{n+1} with no output.

Input x'_j, $1 \leq j \leq n$, allows one to distinguish any two states $s_a^{\mathcal{P}_a}, s_b^{\mathcal{P}_b}$ with $1 \leq a < b \leq n$ if $a = j$ or $b = j$. However, for each $1 \leq i \leq n$ only one transition reaches a state of the form $s_i^{\mathcal{P}_i}$ and this has input at port i and output at every port j such that $(u_i, u_j) \in E$. Thus, x'_j can only be applied in state $s_i^{\mathcal{P}_i}$ if $i = j$ or there is an edge between u_i and u_j in G. Thus, $s_i^{\mathcal{P}_i}$ and $s_j^{\mathcal{P}_j}$ can be locally synch-distinguished ($1 \leq i < j \leq n$) if and only if $(u_i, u_j) \in E$. Clearly, we can distinguish all pairs of states where one or more is either s_0^P or s_{n+1}^P. Thus, a set S' of states of $\chi_{min}(M)$ is a maximal set of pairwise locally synch-distinguishable states of $\chi_{min}(M)$ if and only if it contains s_0^P and s_{n+1}^P and a subset of $\{s_1^{\mathcal{P}_1}, \ldots, s_n^{\mathcal{P}_n}\}$ such that for all $s_i^{\mathcal{P}_i}, s_j^{\mathcal{P}_j} \in S'$ with $s_i^{\mathcal{P}_i} \neq s_j^{\mathcal{P}_j}$ we have that $(u_i, u_j) \in E$. This is the case if and only if $\{u_i | s_i^{\mathcal{P}_i} \in S' \setminus \{s_0^P, s_n^P\}\}$ is a maximal clique of G. Thus, any algorithm that solves the problem of finding a maximal set of states that are pairwise locally synch-distinguishable can also be used to solve the maximal clique problem. The result now follows from the fact that the construction of M from G can be performed in polynomial time and the maximal clique problem is NP-hard. □

Clearly, this result applies also to partial FSMs.

7 Test suite generation

In this section we develop a method for generating a c_m-complete test suite for an FSM. This will build on the result (Theorem 2) that N locally synch-conforms to M if and only if $\chi_{min}(N)$ is equivalent to $\chi_{min}(M)$. Since state counting has been developed for testing from a partial (single-port) FSM [10], we adapt this approach. State counting utilises test cases that reach states of specification M and test cases that distinguish sets of states of M. For the former we require controllable input sequences that reach states (Section 4) and for the latter we require sets of states that can be distinguished in controllable testing (Section 6).

First we show that there is an algorithm for generating a c_m-complete test suite for use in distributed testing. In this, given FSM M and integer a, we let $C(M, a) = C(M) \cap X^a$ denote the set of input sequences of length a that label controllable paths from the initial state of M.

Theorem 4: Given integer m and FSM M with n states, the set of prefixes of $C(M, k(m+n-1))$ is c_m-complete.

Proof: We require to prove that if N is an FSM with the same input and output alphabets as M and no more than m states and N does not locally synch-conform to M, then there is some prefix of an input sequence in $C(M, k(m+n-1))$ that locally synch-distinguishes N from M. Let $M \oplus N$ denote the disjoint union of M and N, which is formed by taking the disjoint union of the states of N and M and retaining the transitions. Then an input sequence locally synch-distinguishes N from M if and only if it is controllable in M and N and locally synch-distinguishes the initial state of N and M in $M \oplus N$. However, by Theorem 1 and from $M \oplus N$ having at most $m+n$ states, there is such an input sequence if and only if there is such an input sequence of length at most $k(m+n-1)$ and so the result follows. □

We thus know that there are c_m-complete test suites. In contrast, it is undecidable whether an FSM has an m-complete test suit [49]. The problem now is to find methods that can return smaller c_m-complete test suites. We will adapt state counting, which can be explained using the product machine $P(M, N)$ for FSM specification M and (unknown) FSM N that models the SUT.

Definition 11: Given FSMs $M = (S, s_0, X, Y, \delta, \lambda)$ and $N = (S^1, s_0^1, X, Y, \delta^1, \lambda^1)$ the product machine $P(M, N)$ is the FSM $(S \times S^1, (s_0, s_0^1), X, Y \cup \{e\}, \delta'', \lambda'')$ for some $e \notin Y$ where δ'' and λ'' are defined by the following in which $(s_1, s_1^1) \in S \times S^1$ and $x \in X$.

1) If $\lambda(s_1, x) = \lambda^1(s_1^1, x)$ then $\lambda''((s_1, s_1^1), x) = \lambda(s_1, x)$ and $\delta''((s_1, s_1^1), x) = (\delta(s_1, x), \delta^1(s_1^1, x))$.
2) If $\lambda(s_1, x) \neq \lambda^1(s_1^1, x)$ then $\lambda''((s_1, s_1^1), x) = e$ and $\delta''((s_1, s_1^1), x) = (\delta(s_1, x), \delta^1(s_1^1, x))$.

$P(M, N)$ simulates the parallel execution of M and N as long as their outputs agree; if their outputs do not agree, and so there has been a failure, the special output e is produced. Thus, a controllable input sequence leads to a failure if and only if it leads to the product machine producing e. This is captured by the following results.

Proposition 8: Given FSMs M and N with the same set of ports and the same input and output alphabets, if input sequence $\bar{x}$ locally synch-distinguishes N from M then $P(M, N)$ produces an output sequence that contains e when given $\bar{x}$.

Proposition 9: Given FSMs M and N with the same set of ports and the same input and output alphabets, if an input sequence $\bar{x}$ leads to $P(M, N)$ producing output e and no proper prefix of $\bar{x}$ does this then $\bar{x}$ locally synch-distinguishes N from M.

The second result differs slightly from results for testing from single-port FSMs since it requires that no proper prefix of the sequence leads to output e; it does so to avoid the potential for observability problems leading to fault masking. To see this consider the FSM M_0' shown in Figure 7; this is the same as M_0 except that the transition from s_0 to s_1 and the transition from s_1 to s_1 have changed. If we compute $P(M_0, M_0')$ we have that the input of x_1 in the initial state leads

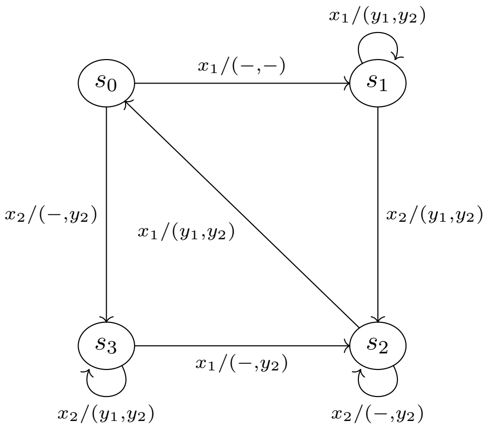

Fig. 7. Finite State Machine M_0'

to different outputs from M_0 and M_0' and so output e. A second input x_1 then leads to output e and so the input sequence x_1x_1 leads to $P(M_0, M_0')$ producing ee. However, the corresponding traces of M_0 and M_0' are $x_1/(-, y_2)x_1/(y_1, -)$ and $x_1/(-, -)x_1/(y_1, y_2)$ respectively and these are observationally equivalent (they have the same sets of projections), despite the last output of $P(M_0, M_0')$ in response to x_1x_1 being e.

We now adapt the approach of Petrenko and Yevtushenko [10]. Previously, state counting was developed for non-deterministic FSMs and the key difference introduced by an FSM being partial is that quasi-equivalence defines a partial order over states (rather than an equivalence relation). Recall that s_i is quasi-equivalent to s_j ($s_j \sqsubseteq s_i$) if $\Omega_M(s_j) \subseteq \Omega_M(s_i)$ and $\lambda(s_i, \bar{x}) = \lambda(s_j, \bar{x})$ for all $\bar{x} \in \Omega_M(s_j)$. State counting is based on reasoning about the states of the product machine and noting that if N does not conform to M then there is some minimal input sequence that demonstrates this. Given an input sequence $\bar{x}$, this reasoning places a lower bound on the number of states that N must have if a particular set of tests sequences lead to no failures and $\bar{x}$ is a (minimal) prefix of an input sequence that leads to a failure. If this lower bound exceeds the upper bound on the number of states of N then there is no need to extend $\bar{x}$ further. The lower-bound will be based on two observations [10].

1) Let us suppose that prefixes $\bar{x}_1$ and $\bar{x}_2$ of $\bar{x}$ reach states s_1 and s_2 respectively of $\chi_{min}(M)$, $\bar{x}_1$ is shorter than $\bar{x}_2$, and $s_2 \sqsubseteq s_1$. If $\bar{x}$ is a minimal prefix of an input sequence $\bar{x}'$ that reaches a failure then $\bar{x}_1$ and $\bar{x}_2$ must reach different states of N. This is based on the observation that (since $s_2 \sqsubseteq s_1$) all behaviours of M from s_2 are also behaviours from s_1 and so, if $\bar{x}_1$ and $\bar{x}_2$ reach the same state of N then we can replace $\bar{x}_2$ by $\bar{x}_1$ in $\bar{x}'$ and still obtain a failure, contradicting the minimality of $\bar{x}'$.
2) Let us suppose that prefixes $\bar{x}_1$ and $\bar{x}_2$ of $\bar{x}$ reach states s_1 and s_2 respectively in $\chi_{min}(M)$ and we can distinguish s_1 and s_2 using input sequence $\bar{w}$.

If the SUT does not fail $\bar{x}_1\bar{w}$ and $\bar{x}_2\bar{w}$ then $\bar{w}$ must also distinguish the states of N reached by $\bar{x}_1$ and $\bar{x}_2$ and so these must be different states of N.

We start by exploring how states of $\chi_{min}(M)$ relate under quasi-equivalence. The first result is immediate from the definitions of $\chi_{min}(M)$ and quasi-equivalence.

Proposition 10: Given state s of FSM M and states $s^{\mathcal{P}_1}$ and $s^{\mathcal{P}_2}$ of $\chi_{min}(M)$, if P' is the set of ports that have non-empty input alphabets then $s^{\mathcal{P}_1} \sqsubseteq s^{\mathcal{P}_2}$ if and only if $\mathcal{P}_1 \cap P' \subseteq \mathcal{P}_2 \cap P'$.

We can extend this to states $s_1^{\mathcal{P}_1}$ and $s_2^{\mathcal{P}_2}$ ($s_1 \neq s_2$) using the following concept.

Definition 12: States s_1 and s_2 of M are *p-equivalent* ($p \in P$) if $C(M(s_1)) \cap X_p(X^*) = C(M(s_2)) \cap X_p(X^*)$ and $\lambda(s_1, \bar{x}) = \lambda(s_2, \bar{x})$ for all $\bar{x} \in C(M(s_1)) \cap X_p(X^*)$.

The first part of the definition ($C(M(s_1)) \cap X_p(X^*) = C(M(s_2)) \cap X_p(X^*)$) requires that the same set of input sequences that start with input at p (sequences in $X_p(X^*)$) are controllable from s_1 and s_2; the second part requires that these lead to the same output sequences. It is straightforward to see that states s and s' of $\chi_{min}(M)$ are p-equivalent if and only if for all $x \in X_p$ we have that $\lambda'(s, x) = \lambda'(s', x)$ and states $\delta'(s, x)$ and $\delta'(s', x)$ are equivalent (recall that δ' and λ' are the state transfer and output functions of $\chi_{min}(M)$). Since state equivalence can be decided in low-order polynomial time for DFSMs we can also decide p-equivalence in polynomial time.

Proposition 11: Let us suppose that s_1 and s_2 are states of FSM M and $s_1^{\mathcal{P}_1}$ and $s_2^{\mathcal{P}_2}$ are states of $\chi_{min}(M)$. If P' is the set of ports that have non-empty input alphabets then we have that $s_1^{\mathcal{P}_1} \sqsubseteq s_2^{\mathcal{P}_2}$ if and only if the following conditions hold.

1) $\mathcal{P}_1 \cap P' \subseteq \mathcal{P}_2 \cap P'$; and
2) For all $p \in \mathcal{P}_1 \cap P'$, $s_1^{\mathcal{P}_1}$ and $s_2^{\mathcal{P}_2}$ are p-equivalent.

Thus, we can decide whether $s_1^{\mathcal{P}_1} \sqsubseteq s_2^{\mathcal{P}_2}$ in low-order polynomial time.

Petrenko and Yevtushenko used the notion of a core cover of a partial FSM.

Definition 13: A set of states S' of $\chi_{min}(M)$ is a *core* of $\chi_{min}(M)$ if S' contains the initial state, for every state s of $\chi_{min}(M)$ there is some $s' \in S'$ such that $s \sqsubseteq s'$ and no proper subset of S' satisfies these conditions. A set K of input sequences is a *core cover* of $\chi_{min}(M)$ if $\epsilon \in K$ and there is a core S' of $\chi_{min}(M)$ such that each state in S' is reached by exactly one sequence in K.

If we consider $\chi_{min}(M_0)$ we find that the states $s_0^{\{1,2\}}$, $s_1^{\{1,2\}}$, $s_2^{\{1,2\}}$, and $s_3^{\{1,2\}}$ are all reachable and form a core of $\chi_{min}(M_0)$. We have that ϵ takes $\chi_{min}(M_0)$ to $s_0^{\{1,2\}}$, x_1 takes $\chi_{min}(M_0)$ to $s_1^{\{1,2\}}$, x_1x_2 takes $\chi_{min}(M_0)$ to $s_2^{\{1,2\}}$, and x_2x_2 takes $\chi_{min}(M_0)$ to $s_3^{\{1,2\}}$. Thus, we have that $\{\epsilon, x_1, x_1x_2, x_2x_2\}$ is a core cover for $\chi_{min}(M_0)$.

Given state s of $\chi_{min}(M)$ and $\bar{x} \in \Omega_{\chi_{min}(M)}(s)$, we can examine the path $\bar{\rho}$ of $\chi_{min}(M)$ with starting state s and a label whose input portion is $\bar{x}$. If t is a state of $\chi_{min}(M)$ then we can look at the *non-empty* prefixes of $\bar{x}$ that reach t or states that are quasi-equivalent to t. This set is denoted $Pref_{s,t}(\bar{x}) = \{\bar{x}' \in pref(\bar{x}) \setminus \{\epsilon\} | t \sqsubseteq \delta(s, \bar{x}')\}$. Partial order $\sqsubseteq_{s,t}$ is defined on $Pref_{s,t}(\bar{x})$ by: $a_i \sqsubseteq_{s,t} a_j$ if $|a_j| \leq |a_i|$ and $a_i \sqsubseteq a_j$. Consider $\chi_{min}(M_0)$ and $s = s_3^{\{1,2\}}$. If we let $\bar{x} = x_1x_2x_2x_2$ then we find that from $s_3^{\{1,2\}}$ the input sequence $\bar{x}$ visits $s_2^{\{1,2\}}$ and then $s_2^{\{2\}}$ three times. If $t = s_2^{\{2\}}$ then $Pref_{s,t}(\bar{x}) = \{x_1, x_1x_2, x_1x_2x_2, x_1x_2x_2x_2\}$ since $s_2^{\{2\}} \sqsubseteq s_2^{\{1,2\}}$. Further, $x_1x_2x_2x_2 \sqsubseteq x_1x_2x_2$, $x_1x_2x_2 \sqsubseteq x_1x_2$, and $x_1x_2 \sqsubseteq x_1$.

Petrenko and Yevtushenko let $\ell(Pref_{s,t}(\bar{x}), \sqsubseteq_{s,t})$ be the length of the longest chain in partially ordered set $(Pref_{s,t}(\bar{x}), \sqsubseteq_{s,t})$. For $\chi_{min}(M_0)$, $\bar{x} = x_1x_2x_2x_2$, $s = s_3^{\{1,2\}}$ and $t = s_2^{\{2\}}$, $\ell(Pref_{s,t}(\bar{x}), \sqsubseteq_{s,t}) = 4$. The key point corresponds to observation 1: given chain $\bar{a}_1, \ldots, \bar{a}_b$ in $(Pref_{s,t}(\bar{x}), \sqsubseteq_{s,t})$, if $1 \leq i < j \leq b$ then all behaviours of $\chi_{min}(M)$ possible in the state reached from s by $\bar{a}_j$ are possible from the state of $\chi_{min}(M)$ reached from s by $\bar{a}_i$. Further, all behaviours possible from t are possible from $\bar{a}_i$ and $\bar{a}_j$. If we apply $\bar{x}$ after initial input sequence $\bar{x}'$, such as one in a core cover K, then we can reason about the states of the SUT met by sequences of the form $\bar{x}'\bar{a}_i$. In particular, if $1 \leq i < j \leq b$ and $\bar{x}'\bar{a}_i$ and $\bar{x}'\bar{a}_j$ reach the same state of the SUT then $\bar{x}'\bar{x}$ cannot be a shortest extension of $\bar{x}'$ that leads to a failure; since $\bar{a}_j \sqsubseteq_{s,t} \bar{a}_i$, we can replace $\bar{a}_j$ by $\bar{a}_i$ without losing any behaviours and so obtain the failure with a shorter sequence.

The following result adapts one previously proved (Lemma 3, [10]) and is based on the above observation.

Lemma 1: Given core cover K of $\chi_{min}(M)$, FSM N with at most m states, and state q of $P(\chi_{min}(M), N)$, there exists $\bar{x}' \in K$ and $\bar{x}'\bar{x} \in \Omega(P(\chi_{min}(M), N))$ that reaches a state q' with $q \sqsubseteq q'$ such that $\ell(Pref_{\delta'(s_0', \bar{x}'),t}(\bar{x}), \sqsubseteq_{\delta'(s_0', \bar{x}'),t}) \leq m - 1$ for every state t of $\chi_{min}(M)$.

State counting also takes advantage of situations in which states can be distinguished (observation 2). By Proposition 7, if prefixes $\bar{x}'\bar{x}_1$ and $\bar{x}'\bar{x}_2$ of $\bar{x}'\bar{x}$ reach states s_1 and s_2 of M that can be distinguished, we follow each by an input sequence $\bar{w}$ that distinguishes s_1 and s_2, and the behaviour is as specified then $\bar{w}$ must distinguish the states of the SUT reached by $\bar{x}'\bar{x}_1$ and $\bar{x}'\bar{x}_2$. Thus, $\bar{x}'\bar{x}_1$ and $\bar{x}'\bar{x}_2$ reach distinct states of the SUT.

Let us suppose that R is a set of pairwise distinguishable states of $\chi_{min}(M)$ and for each pair s_1, s_2 of distinct states in R the input sequence $\gamma(s_1, s_2)$ distinguishes s_1 and s_2. For $t \in R$, $R_t = \{\gamma(s, t) | s \in R \setminus \{t\}\}$ distinguishes t from all other states in R. We will assume that $\gamma(s, t)$ is fixed; we do not use different input sequences to distinguish s and t for different R. This assumption simplifies the exposition and can easily be relaxed. Given R, K_R will be the set of input sequences from the core cover that reach states that are quasi-equivalent to states in R ($K_R = \{\bar{x} \in K | \exists t \in R . t \sqsubseteq \delta'(s_0', \bar{x})\}$). If $\bar{x} \in K_R$ takes $\chi_{min}(M)$ to a state quasi-equivalent to t, then we can follow $\bar{x}$ with elements of R_t. Observe that between them K_R and R_t define a state identification tuple (Section 6).

Let us suppose that we extend $\bar{x}' \in K$ by $\bar{x}$ ($\bar{x} \in \Omega_{\chi_{min}(M)}(\delta'(s_0', \bar{x}'))$). Further, let us suppose that for all $t \in R$ and $\bar{x}'' \in pref(\bar{x})$ such that $t \sqsubseteq \delta'(s_0', \bar{x}'\bar{x}'')$ we have that N conforms to $\chi_{min}(M)$ on all $\bar{x}'\bar{x}''\bar{w}$ such that $\bar{w} \in R_t$. As noted above, for any $t_1, t_2 \in R$ with $t_1 \neq t_2$, $\bar{x}_1 \in pref(\bar{x})$ such that $t_1 \sqsubseteq \delta'(s'_0, \bar{x}'\bar{x}'')$, and $\bar{x}_2 \in pref(\bar{x})$ such that $t_2 \sqsubseteq \delta'(s'_0, \bar{x}'\bar{x}'')$ we must have that $\bar{x}'\bar{x}_1$ and $\bar{x}'\bar{x}_2$ reach different states of the SUT (otherwise one of the tests would fail when followed by $\gamma(t_1, t_2)$).

Consider $\chi_{min}(M_0)$, $\bar{x} = \epsilon$, and $\bar{x}' = x_2x_2x_1x_1$. The states reached by non-empty prefixes of $\bar{x}'$ are $s_3^{\{2\}}$, $s_3^{\{1,2\}}$, $s_2^{\{1,2\}}$ and $s_0^{\{1,2\}}$ respectively. We have that $s_3^{\{2\}} \sqsubseteq s_3^{\{1,2\}}$ and that the states in $R = \{s_0^{\{1,2\}}, s_2^{\{1,2\}}, s_3^{\{1,2\}}\}$ are pairwise distinguishable. We can set $\gamma(s_0^{\{1,2\}}, s_2^{\{1,2\}}) = x_1$, $\gamma(s_0^{\{1,2\}}, s_3^{\{1,2\}}) = x_2$, and $\gamma(s_2^{\{1,2\}}, s_3^{\{1,2\}}) = x_2$. If we follow each non-empty prefix of $\bar{x}'$ that reaches a state in R by the corresponding input sequences in R_t, we obtain: x_2x_2 followed by x_2 ($R_{s_3^{\{1,2\}}} = \{x_2\}$), $x_2x_2x_1$ followed by x_1 and x_2 ($R_{s_2^{\{1,2\}}} = \{x_1, x_2\}$), and $x_2x_2x_1x_1$ followed by x_1 and x_2 ($R_{s_0^{\{1,2\}}} = \{x_1, x_2\}$). If the SUT passes these test cases (and their prefixes) then x_2x_2, $x_2x_2x_1$ and $x_2x_2x_1x_1$ reach different states of the SUT (since the states reached are distinguished in testing).

Now let us suppose that $\bar{x}'\bar{x}$ with $\bar{x}' \in K$ is a prefix of a minimal extension of an element of K that leads to failure. Let us also suppose that the SUT does not fail on tests of the above form ($\bar{x}'\bar{x}''\bar{w}$ where $\bar{x}'\bar{x}''$ reaches a state quasi-equivalent to t and $\bar{w} \in R_t$). By the minimality of $\bar{x}$, the prefixes of $\bar{x}$ in $(Pref_{\delta(s_0,\bar{x}'),t}(\bar{x}), \sqsubseteq_{\delta(s_0,\bar{x}'),t})$ for state $t \in R$ reach different states of the SUT (Lemma 1). Further, if two prefixes of $\bar{x}'$ reach states quasi-equivalent to different states from R when applied from $\delta'(s'_0, \bar{x}')$ then, since the SUT passes these tests, these must also reach different states of the SUT. Further, let us suppose that in testing we follow each sequence in the core cover by the corresponding sequences used to distinguish the states. In M the cover reaches a set of states that are quasi-equivalent to those in R and so the minimality of $\bar{x}'\bar{x}$ implies that no non-empty prefix of $\bar{x}'$ reaches a state of the product machine that is also reached by a corresponding element of the core cover. This leads to the following lower bound on the number of states of the SUT if no failures are observed.

$$lb(\bar{x}', \bar{x}, R) = \sum_{t \in R} \ell(Pref_{\delta'(s'_0,\bar{x}'),t}(\bar{x}), \sqsubseteq_{\delta'(s'_0,\bar{x}'),t}) + |R|$$

Thus, if this value exceeds m then, since the SUT has at most m states, either the SUT fails one or more of these test cases or $\bar{x}'$ is not a prefix of a minimal extension of an element of K that leads to failure. In either case there is no need to extend $\bar{x}\bar{x}'$ further.

There can be alternative sets of pairwise distinguishable states of $\chi_{min}(M)$ and we let $\mathcal{R}$ denote the set of known sets of pairwise distinguishable states. Given a core cover K, input sequence $\bar{x}' \in K$, and set $\mathcal{R}$, we will consider the maximum value over $R \in \mathcal{R}$.

$$lb'(\bar{x}', \bar{x}, \mathcal{R}) = \max_{R \in \mathcal{R}} lb(\bar{x}', \bar{x}, R)$$

Given $\bar{x}' \in K$ and integer m, the following set of input sequences is then defined.

$$N(\bar{x}', \mathcal{R}) = \left\{ \begin{array}{l} \bar{x} \in \Omega_{\chi_{min}(M)}(\delta'(s'_0, \bar{x}'))| \\ \forall \bar{x}'' \in pref(\bar{x}) \setminus \{\bar{x}\}.lb'(\bar{x}', \bar{x}'', \mathcal{R}) \leq m \\ \wedge lb'(\bar{x}', \bar{x}, \mathcal{R}) = m + 1) \end{array} \right\}$$

The essential idea is that for $\bar{x}$ to be in $N(\bar{x}', \mathcal{R})$ we require the following to hold for any SUT that does not fail the test (where we extend a prefix $\bar{x}''$ of $\bar{x}$ by R_t whenever we have that $t \sqsubseteq \delta'(s'_0, \bar{x}'\bar{x}'')$ and $t \in R$):

- No proper prefix $\bar{x}''$ of $\bar{x}$ satisfies the termination criterion: for all $R \in \mathcal{R}$, $lb(\bar{x}', \bar{x}'', R) \leq m$; and
- $\bar{x}$ satisfies the termination criterion that there is some $R \in \mathcal{R}$ such that $lb(\bar{x}', \bar{x}, R) > m$.

In state counting from partial single-port FSMs, for the second condition it is necessary to consider the case where $\bar{x}'\bar{x}$ cannot be extended due to no further inputs being defined [10]. However, this cannot happen here since we require M to be completely-specified.

Proposition 12: If s is a reachable state of $\chi_{min}(M)$ then at least one transition leaves s.

Proof: Consider a path to s whose label $\bar{x}/\bar{y}$ has an input portion that ends in x. Input x can be followed by x without causing a controllability problem. Since we can apply x after $\bar{x}/\bar{y}$ without causing controllability problems, there is a transition from s with input x. □

If a sequence $\bar{x}$ is in $N(\bar{x}', \mathcal{R})$ then we choose a set $R(\bar{x}', \bar{x}) \in \mathcal{R}$ that can be used in determining that the termination criterion holds along with a maximal chain $C(\bar{x}', \bar{x}, t)$ in $(Pref_{\delta'(s'_0,\bar{x}'),t}(\bar{x}), \sqsubseteq_{\delta'(s'_0,\bar{x}'),t})$ ($t \in R$). Note that although the states of $\chi_{min}(M)$ reached by the input sequences in a chain need not be the same, by the definition of $\sqsubseteq_{s,t}$, each at least has the behaviours of t and thus the input sequences from R_t can be used.

The resultant test suite has two parts:

1) For a sequence $\bar{x}$ from the core cover, that reaches state s of $\chi_{min}(M)$, $\bar{x}$ followed by every input sequence in R_s for set R used.
2) The set of prefixes of: $\bar{x}'$ followed by every $\bar{x}_1\bar{w}$ such that $\bar{x}_1$ appears in a maximal chain in some $Pref_{\delta'(s'_0,\bar{x}'),t}(\bar{x})$ and $\bar{w} \in R_t$.

The algorithm is summarised in Algorithm 1. Once $\chi_{min}(M)$ has been constructed there are two loops. The first constructs the $N(\bar{x}', \mathcal{R})$ and the corresponding test cases. The second loop adds in the test cases that result from members of the core cover.

If we do not take prefixes then Algorithm 1 returns a test suite that is m-complete for $\chi_{min}(M)$ [10]. Thus, by Theorem 2, we obtain the following.

Theorem 5: Given an FSM M and integer m, Algorithm 1 returns a c_m-complete test suite for M.

The algorithm thus returns a test suite with guaranteed fault detection ability. Similar to state counting, test suite size depends on several factors. First, the size of the test suite grows exponentially in terms of $m - n$ even for a completely-specified single-port FSM [2], [18]. In state counting the test suite size also depends on the number of states that are in the core and the sizes of the sets of

Algorithm 1 Test suite generation

```
Input FSM $M$ and integer $m$.
Construct $\chi_{min}(M)$ a core cover $K$ for $\chi_{min}(M)$.
Produce a set of input sequences that distinguish states of $\chi_{min}(M)$ and corresponding set $\mathcal{R}$ of sets of pairwise distinguishable states.
Set $\mathcal{T} = \emptyset$.
for all $\bar{x}' \in K$ do
    Find $N(\bar{x}', \mathcal{R})$.
    For each $\bar{x} \in N(\bar{x}', \mathcal{R})$ let $R = R(\bar{x}', \bar{x}) \in \mathcal{R}$ be a set used to demonstrate that we can terminate with $\bar{x}$ and for $t \in R$ let $C(\bar{x}', \bar{x}, t)$ denote some corresponding maximal chain.
    Add to $\mathcal{T}$ the set of $\bar{x}'\bar{x}''\bar{w}$ such that $\bar{x}'' \in C(\bar{x}', \bar{x}, t)$ for some $\bar{x} \in N(\bar{x}', \mathcal{R})$, $t \in R(\bar{x}, \bar{x}')$, and $\bar{w} \in R_t$.
end for
for all $\bar{x}' \in K$ do
    Let $s$ be the state of $\chi_{min}(M)$ reached by $\bar{x}'$.
    Add to $\mathcal{T}$ all sequences of the form $\bar{x}'\bar{w}$ such that there is some $t \sqsubseteq s$ in some $R = R(\bar{x}', \bar{x})$ used in a termination criterion and $\bar{w} \in R_t$.
end for
Return $pref(\mathcal{T})$.
```

pairwise distinguishable states and grows exponentially as the sizes of these two sets reduce. The dependence on the size of the sets of pairwise distinguishable states motivated our interest in finding maximal such sets (Section 6). Thus, this approach will scale best in situations in which the core is relatively large and most states of $\chi_{min}(M)$ are pairwise distinguishable. The tester can apply a cost benefit analysis in choosing a value for m.

8 Conclusions

This paper defined the notion of a c_m-complete test suite: a set of controllable test cases that distinguish FSM specification M from any FSM N that has no more than m states and can be distinguished from M in controllable distributed testing. This was motivated by two factors. First, controllable test cases provide practical advantages (the generation of controllable test cases has been the main focus of work on distributed testing). Second, determining whether N can be distinguished from M in distributed testing is generally undecidable [27] and so there is no general method for producing an m-complete test suite for distributed testing.

We proved that an FSM M can be mapped to a partial FSM $\chi_{min}(M)$ such that a test suite is m-complete for $\chi_{min}(M)$ if and only if it is c_m-complete for M. Thus, methods for generating m-complete test suites from partial single-port FSMs can be adapted for use in distributed testing. Further, $\chi_{min}(M)$ can be constructed in low-order polynomial time. We proved that the problem of finding maximal sets of pairwise distinguishable states is NP-complete for distributed testing and also testing from a partial FSM. This result is relevant since most methods for generating an m-complete test suite take advantage of sets of pairwise distinguishable states: the size of the m-complete test suites depends on the size of the sets of pairwise distinguishable states used. Finally, we showed how the state counting method for partial FSMs can be adapted to distributed testing and explored how the properties of distributed testing affect this method.

There are several lines of future work. The proposed method avoids observability problems by including all prefixes of the test cases but we may not need all such prefixes. Let us suppose, for example, that there is a test cases $x_1x_1x_2$ with expected trace $x_1/(y_1,-)x_1/(y_1,y_2)x_2/(y_1',y_2)$. Further, let us suppose that we have tested with x_1 and observed x_1y_1 at port 1 and ϵ at port 2 and tested with $x_1x_1x_2$ and observed $x_1y_1x_1y_1y_1'$ at port 1 and $y_2x_2y_2$ at port 2. We can deduce that the last two inputs in $x_1x_1x_2$ lead to two outputs at each port and so the response to x_1x_1 must have been $(y_1,-)(y_1,y_2)$. Thus, we do not have to include prefix x_1x_1. The first challenge is that of determining which prefixes are required. A second challenge is to incorporate such minimisation into test suite generation. It would also be good to see research that explores the impact of restricting testing to controllable test cases, ideally investigating a range of classes of systems. Finally, there may be value in devising test cases that are not controllable but where, for example, there are controllable 'parts' that achieve the test objectives.

References

[1] W. Grieskamp, N. Kicillof, K. Stobie, and V. Braberman, "Model-based quality assurance of protocol documentation: tools and methodology," *The Journal of Software Testing, Verification and Reliability*, vol. 21, no. 1, pp. 55–71, 2011.

[2] T. S. Chow, "Testing software design modelled by finite state machines," *IEEE Transactions on Software Engineering*, vol. 4, pp. 178–187, 1978.

[3] R. Dssouli and G. von Bochmann, "Error detection with multiple observers," in *Protocol Specification, Testing and Verification V*. Elsevier Science (North Holland), 1985, pp. 483–494.

[4] ——, "Conformance testing with multiple observers," in *Protocol Specification, Testing and Verification VI*. Elsevier Science (North Holland), 1986, pp. 217–229.

[5] F. C. Hennie, "Fault-detecting experiments for sequential circuits," in *Proceedings of Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, Princeton, New Jersey, 1964, pp. 95–110.

[6] R. M. Hierons and H. Ural, "The effect of the distributed test architecture on the power of testing," *The Computer Journal*, vol. 51, no. 4, pp. 497–510, 2008.

[7] E. F. Moore, "Gedanken-experiments," in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton University Press, 1956.

[8] A. Petrenko, N. Yevtushenko, A. Lebedev, and A. Das, "Nondeterministic state machines in protocol conformance testing," in *Proceedings of Protocol Test Systems, VI (C-19)*. Pau, France: Elsevier Science (North-Holland), 28-30 September 1994, pp. 363–378.

[9] A. Petrenko, N. Yevtushenko, and G. v. Bochmann, "Testing deterministic implementations from nondeterministic FSM specifications," in *Testing of Communicating Systems, IFIP TC6 9th International Workshop on Testing of Communicating Systems*. Darmstadt, Germany: Chapman and Hall, 9–11 September 1996, pp. 125–141.

[10] A. Petrenko and N. Yevtushenko, "Testing from partial deterministic FSM specifications," *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1154–1165, 2005.

[11] B. Sarikaya and G. Bochmann, "Synchronization and specification issues in protocol testing," *IEEE Transactions on Communications*, vol. 32, pp. 389–395, April 1984.

[12] N. V. Yevtushenko, A. V. Lebedev, and A. F. Petrenko, "On checking experiments with nondeterministic automata," *Automatic Control and Computer Sciences*, vol. 6, pp. 81–85, 1991.

[13] E. Brinksma, "A theory for the derivation of tests," in *Proceedings of Protocol Specification, Testing, and Verification VIII*. Atlantic City: North-Holland, 1988, pp. 63–74.

[14] E. Brinksma, L. Heerink, and J. Tretmans, "Factorized test generation for multi-input/output transition systems," in *11th International Workshop on Testing Communicating Systems (IWTCS)*, ser. IFIP Conference Proceedings, vol. 131. Kluwer, 1998, pp. 67–82.

[15] R. M. Hierons, M. G. Merayo, and M. Núñez, "Implementation relations and test generation for systems with distributed interfaces," *Distributed Computing*, vol. 25, no. 1, pp. 35–62, 2012.

[16] J. Tretmans, "Conformance testing with labelled transitions systems: Implementation relations and test generation," *Computer Networks and ISDN Systems*, vol. 29, no. 1, pp. 49–79, 1996.

[17] ——, "Model based testing with labelled transition systems," in *Formal Methods and Testing*, ser. Lecture Notes in Computer Science, vol. 4949. Springer, 2008, pp. 1–38.

[18] M. P. Vasilevskii, "Failure diagnosis of automata," *Cybernetics*, vol. 4, pp. 653–665, 1973.

[19] R. M. Hierons, "Testing from a non-deterministic finite state machine using adaptive state counting," *IEEE Transactions on Computers*, vol. 53, no. 10, pp. 1330–1342, 2004.

[20] ISO/IEC, *Information technology - Opens Systems Interconnection, 9646 Parts 1-7*. ISO/IEC, 1995.

[21] W.-H. Chen and H. Ural, "Synchronizable checking sequences based on multiple UIO sequences," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 152–157, 1995.

[22] R. M. Hierons and H. Ural, "UIO sequence based checking sequences for distributed test architectures," *Information and Software Technology*, vol. 45, no. 12, pp. 793–803, 2003.

[23] ——, "Checking sequences for distributed test architectures," *Distributed Computing*, vol. 21, no. 3, pp. 223–238, 2008.

[24] G. Luo, R. Dssouli, and G. v. Bochmann, "Generating synchronizable test sequences based on finite state machine with distributed ports," in *The 6th IFIP Workshop on Protocol Test Systems*. Elsevier (North-Holland), 1993, pp. 139–153.

[25] K.-C. Tai and Y.-C. Young, "Synchronizable test sequences of finite state machines," *Computer Networks and ISDN Systems*, vol. 30, no. 12, pp. 1111–1134, 1998.

[26] Y. C. Young and K. C. Tai, "Observational inaccuracy in conformance testing with multiple testers," in *IEEE 1st workshop on application-specific software engineering and technology*, 1998, pp. 80–85.

[27] R. M. Hierons, "Reaching and distinguishing states of distributed systems," *SIAM Journal on Computing*, vol. 39, no. 8, pp. 3480–3500, 2010.

[28] S. Guyot and H. Ural, "Synchronizable checking sequences based on UIO sequences," in *Protocol Test Systems, VIII*. Evry, France: Chapman and Hall, September 1995, pp. 385–397.

[29] A. Khoumsi, "A temporal approach for testing distributed systems," *IEEE Transactions on Software Engineering*, vol. 28, no. 11, pp. 1085–1103, 2002.

[30] O. Rafiq and L. Cacciari, "Coordination algorithm for distributed testing," *The Journal of Supercomputing*, vol. 24, no. 2, pp. 203–211, 2003.

[31] C. Gaston, R. M. Hierons, and P. L. Gall, "An implementation relation and test framework for timed distributed systems," in *25th IFIP WG 6.1 International Conference on Testing Software and Systems (ICTSS 2013)*, ser. Lecture Notes in Computer Science, vol. 8254. Springer, 2013, pp. 82–97.

[32] A. Pretschner, W. Prenninger, S. Wagner, C. Kühnel, M. Baumgartner, B. Sostawa, R. Zölch, and T. Stauner, "One evaluation of model-based testing and its automation," in *27th International Conference on Software Engineering (ICSE 2005)*, 2005, pp. 392–401.

[33] A. Y. Duale and M. U. Uyar, "A method enabling feasible conformance test sequence generation for EFSM models," *IEEE Transactions on Computers*, vol. 53, no. 5, pp. 614–627, 2004.

[34] S. Boyd and H. Ural, "The synchronization problem in protocol testing and its complexity," *Information Processing Letters*, vol. 40, no. 3, pp. 131–136, 1991.

[35] R. M. Hierons, "Canonical finite state machines for distributed systems," *Theoretical Computer Science*, vol. 411, no. 2, pp. 566–580, 2010.

[36] E. C. de Almeida, J. E. Marynowski, G. Sunyé, Y. L. Traon, and P. Valduriez, "Efficient distributed test architectures for large-scale systems," in *22nd IFIP WG 6.1 International Conference on Testing Software and Systems (ICTSS 2010)*, ser. Lecture Notes in Computer Science, vol. 6435. Springer, 2010, pp. 174–187.

[37] R. M. Hierons, "Testing a distributed system: generating minimal synchronised test sequences that detect output-shifting faults," *Information and Software Technology*, vol. 43, no. 9, pp. 551–560, 2001.

[38] W.-J. Wu, W.-H. Chen, and C. Y. Tang, "Synchronizable test sequence for multi-party protocol conformance testing," *Computer Communications*, vol. 21, no. 13, pp. 1177–1183, 1998.

[39] G.-V. Jourdan, H. Ural, and H. Yenigün, "Minimizing coordination channels in distributed testing," in *Formal Techniques for Networked and Distributed Systems (FORTE 2006)*, ser. Lecture Notes in Computer Science, vol. 4229. Springer, 2006, pp. 451–466.

[40] R. M. Hierons and H. Ural, "Overcoming controllability problems with fewest channels between testers," *Computer Networks*, vol. 53, no. 5, pp. 680–690, 2009.

[41] S. Haar, C. Jard, and G.-V. Jourdan, "Testing input/output partial order automata," in *19th International Conference on Testing of Software and Communicating Systems*, ser. Lecture Notes in Computer Science, vol. 4581. Springer, 2007, pp. 171–185.

[42] G. von Bochmann, S. Haar, C. Jard, and G.-V. Jourdan, "Testing systems specified as partial order input/output automata," in *20th International Conference on Testing of Software and Communicating Systems (TestCom/FATES)*, ser. Lecture Notes in Computer Science, vol. 5047. Springer, 2008, pp. 169–183.

[43] H. P. de León, S. Haar, and D. Longuet, "Unfolding-based test selection for concurrent conformance," in *25th International Conference on Testing Software and Systems (ICTSS 2013)*, ser. Lecture Notes in Computer Science, vol. 8254. Springer, 2013, pp. 98–113.

[44] R. Alur, K. Etessami, and M. Yannakakis, "Inference of message sequence charts," in *22nd IEEE International Conference on Software Engineering*, Limerick, Ireland, 4-11 June 2000, pp. 304–313.

[45] ——, "Realizability and verification of MSC graphs," *Theoretical Computer Science*, vol. 331, no. 1, pp. 97–114, 2005.

[46] A. Gill, *Introduction to The Theory of Finite State Machines*. McGraw-Hill, New York, 1962.

[47] J. E. Hopcroft, "An n log n algorithm for minimizing the states in a finite automaton," in *The theory of Machines and Computation*, Z. Kohavi, Ed. Academic Press, 1971, pp. 189–196.

[48] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[49] R. M. Hierons, "Verifying and comparing finite state machines for systems that have distributed interfaces," *IEEE Transactions on Computers*, vol. 62, no. 8, pp. 1673–1683, 2013.

Robert M Hierons Rob Hierons received a BA in Mathematics (Trinity College, Cambridge), and a Ph.D. in Computer Science (Brunel University). He then joined the Department of Mathematical and Computing Sciences at Goldsmiths College, University of London, before returning to Brunel University in 2000. He was promoted to full Professor in 2003.