REACHING AND DISTINGUISHING STATES OF DISTRIBUTED SYSTEMS

ROBERT M. HIERONS*

Abstract. Some systems interact with their environment at physically distributed interfaces, called ports, and in testing such a system it is normal to place a tester at each port. Each tester observes only the events at its port and it is known that this limited observational power introduces additional controllability and observability problems into testing. Given a multi-port finite state machine (FSM) M, we consider the problems of defining strategies for the testers to either reach a given state of M or to distinguish two states of M. These are important problems since most techniques for testing from a single-port FSM use sequences that reach and distinguish states. Both problems can be solved in low-order polynomial time for single-port FSMs but we prove that the corresponding decision problems are undecidable for multi-port FSMs. However, we also show that they can be solved in low order polynomial times for deterministic FSMs if we restrict attention to controllable tests. These results have important ramifications for testing from a multi-port FSM since they suggest that methods for testing from a single-port FSMs cannot be easily adapted. In addition, two FSMs can be distinguished if and only if their initial states can be distinguished and so the results suggest that, in contrast to single-port FSMs, we cannot expect to produce general complete test generation methods for multi-port FSMs.

Key words. finite state machine, testing, distributed systems, multi-port systems.

AMS subject classifications. 68Q45 Formal languages and automata; 68N30 Mathematical aspects of software engineering; 68Q85 Models and methods for concurrent and distributed computing; 68Q17 Computational difficulty of problems; 68Q60 Specification and verification

1. Introduction.

Many systems have physically distributed interfaces, called ports, at which they interact with their environment. Examples include communications protocols but also web-based systems and cloud computing. In testing such a system we place a tester at each port and the tester at a port p only observes the events that occur at p. It has been known for some time that this inability of the tester to observe events at other ports introduces additional controllability and observability problems into testing (see, for example, [5, 6, 8, 11, 12, 21, 25, 34, 35]). This has led to interest in algorithms that generate *test sequences* (input sequences) that have no such controllability and observability problems but it is known that such test sequences do not always exist.

In this paper we consider the black-box testing of a system under test (SUT) that has multiple interfaces. Since distributed systems are often state-based, we assume that the model of the SUT is state-based and in particular that it is a finite state machine (FSM). We consider FSMs since many state-based formalisms, such as statecharts and SDL, can be seen as FSMs, possibly with data added. Many model-based testing techniques also use FSMs or similar formalisms (see, for example, [4, 13, 15]). In addition, FSMs are a syntactically simple formalism that allow us to investigate foundational issues. In particular, in this paper we prove negative decidability results for FSMs and, having proved this for FSMs, we know that these results must also hold for all formalisms that are more expressive.

It is known that the use of physically distributed ports introduces controllability problems [11]. Let us suppose that we wish to apply the test sequence x_1x_2 to a system specified by the FSM M, x_1 is input at port 1, and x_2 is input at 2. If M only sends output to port 1 in response to x_1 then the tester at port 2 cannot know when

*School of Information Systems, Computing and Mathematics, Brunel University, Uxbridge, UB8 3PH, UK

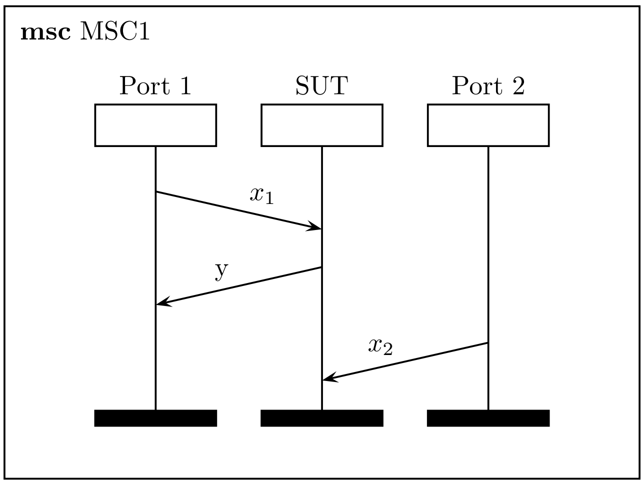

FIG. 1.1. *A controllability problem*

to send x_2. This creates a controllability problem as illustrated in MSC1 in Figure 1.1 in which each vertical line represents a port with time progressing as we move down a line. We have a controllability problem if a tester is required to send an input but does not know when to send this input. There may be no sequence that satisfies a test objective and that has no controllability problems [34]. Thus, if we wish to achieve a test objective, such as testing the response of the SUT to an input when in a particular state, we may have to use tests that contain controllability problems. Despite this, almost all work in this area has restricted testing to the use of test sequences that cause no controllability problems. This motivates the work described in this paper: if we wish to be able to have tests that include controllability problems and yet achieve certain test objectives then we need to know how, and when, we can produce such tests.

When testing from a single-port FSM it is normal to produce a set of test sequences. In contrast, we show that we need the testers at each port to correspond to adaptive strategies when testing from a multi-port FSM even when the FSM is deterministic. The behaviour of the tester at a port p can then adapt to the observations it makes. The use of such strategies is particularly important when we allow testing to include controllability problems since the order in which inputs are received by the SUT is not predictable and this introduces additional nondeterminism into testing. Previous work, in testing from multi-port FSMs, appears not to have considered the use of such strategies and has restricted attention to test sequences that do not cause controllability problems.

In this paper we consider the problem of defining strategies to achieve two goals: reaching a state of an FSM and distinguishing two states of an FSM. One of the main motivations for considering these problems is that most algorithms for generating tests from single-port FSMs use input sequences or adaptive processes to reach and distinguish states (see, for example, [1, 2, 9, 17, 22, 26, 32]). If we can produce strategies to reach and distinguish states of a multi-port FSM then we may be able to adapt techniques for testing from a single-port FSM. We consider several situations. First, we consider the case in which we are testing from a deterministic FSM (DFSM) and we restrict testing to controllable tests. It transpires that in this situation the problems of reaching and distinguishing states can be solved in low-order polynomial time. We then show that the problems, of reaching and distinguishing states, are undecidable for nondeterministic FSMs even if we restrict attention to strategies that cause no controllability problems. Finally, we prove that the general problems are also undecidable for DFSMs.

There is an additional consequence of these results, which is that since the problem of distinguishing two FSMs in testing corresponds to distinguishing their initial states, it is undecidable whether two FSMs can be distinguished. This contrasts with the case for single-port FSMs, the decidability result for single-port FSMs showing that it is possible to find a finite set of test sequences that distinguishes between a single-port FSM M and a finite set of faulty machines [27]. Given a single-port FSM M and integer r, there are algorithms that return sets of test sequences that distinguish between M and all faulty FSMs with at most r states (see, for example, [9, 14, 17, 19, 22, 32, 31]) and such sets are said to be r-complete. The results in this paper suggest that we cannot expect to produce general algorithms that take a multi-port FSM and return an r-complete set of test sequences.

The paper is structured as follows. First, in Section 2, we describe multi-port FSMs and controllability problems. In Section 3 we then consider the problems of reaching and distinguishing states of a DFSM when we restrict attention to sequences that cause no controllability problems. In Section 4 we explain why strategies are required in order to reach and distinguish states of a multi-port FSM when we do not restrict attention to tests that cause no controllability problems and we explore such strategies. Section 5 describes multi-player team games of incomplete information. In Section 6 we use results from multi-player games to prove that it is undecidable where there is a strategy that reaches a given state of a multi-port nondeterministic FSM or that distinguishes two states of such an FSM. In Section 7 we then prove that these problems are generally undecidable for DFSMs. Finally, conclusions are drawn and future work discussed in Section 8.

2. Multi-port finite state machines and controllability problems.

In this paper we assume that the system being considered has m physically distributed interfaces, called ports, and we let $\mathcal{P} = \{1, \ldots, m\}$ be the set of names of the ports. A multi-port finite state machine M with m ports is defined by a tuple (S, s_0, X, Y, h) in which $S = \{s_1, \ldots, s_n\}$ is a finite set of states; $s_0 \in S$ is the initial state; X is the finite input alphabet; Y is the finite output alphabet; and h is the transition relation of type $S \times X \leftrightarrow S \times Y$. For all $p \in \mathcal{P}$, X_p is the set of inputs that can be received at p and so $X = X_1 \cup \ldots \cup X_m$. Similarly, $Y = (Y_1 \cup \{-\}) \times \ldots \times (Y_m \cup \{-\})$ where for all $p \in \mathcal{P}$, Y_p denotes the outputs the SUT can send to port p and $-$ denotes no output being sent to a port. $(y_1, \ldots, y_m) \in Y$ denotes the value $y_p \in Y_p \cup \{-\}$ being sent to port p (or no output being sent to p if $y_p = -$). We assume that the sets of X_p and Y_p are pairwise disjoint: for all $p, q \in \mathcal{P}$ with $p \neq q$ we have that $X_p \cap X_q = \emptyset$ and $Y_p \cap Y_q = \emptyset$. We can ensure that this is the case by adding port labels to inputs and outputs if necessary.

If $(s', y) \in h(s, x)$ and M receives input x when in state s then it can produce output y and move to s'. This defines a *transition* $(s, s', x/y)$. This type of model corresponds to a reactive system where an input or event triggers an operation and this operation can change the state and produce output. Most work on testing state based systems with multiple ports has used such formalisms but an alternative is to allow transitions to be triggered by multiple inputs [16]. M is a *deterministic FSM (DFSM)* if for all $s \in S$ and $x \in X$, $|h(s, x)| \leq 1$. M is *completely specified* if for

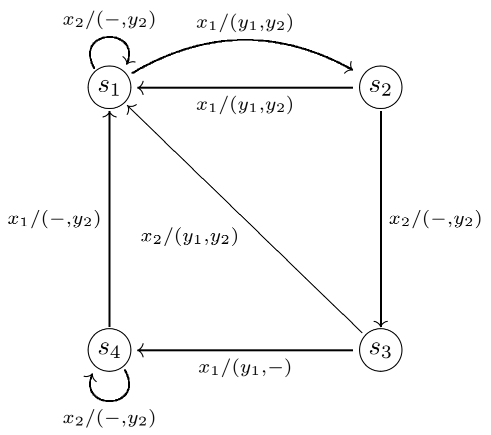

FIG. 2.1. *Finite State Machine M_0*

all $s \in S$ and $x \in X$, $|h(s,x)| \geq 1$. This paper only considers completely specified FSMs and throughout the paper M will refer to such an FSM. Since we are interested in systems with multiple ports, a multi-port finite state machine will be called a *finite state machine (FSM)*; we will refer to FSMs with one port as *single-port FSMs*. Single-port FSMs are sometimes called Mealy machines or transducers.

A DFSM, called M_0 throughout the paper, is shown in Figure 2.1. In M_0, the initial state is s_1, input x_1 is received at port 1 and x_2 is received at port 2. Here, for example, if M_0 receives input x_2 at port 2 when in state s_2 then it moves to states s_3 and sends output y_2 to port 2 while if it receives input x_2 at port 2 when in state s_3 then it moves to state s_1 and sends output y_1 to port 1 and y_2 to port 2. Thus, $h(s_2, x_2) = \{(s_3, (-, y_2))\}$ and $h(s_3, x_2) = \{(s_1, (y_1, y_2))\}$.

An FSM M interacts with its environment through a sequence of steps where each step involves M receiving an input and producing an output and so each such step corresponds to a transition of M. M thus interacts with its environment through a sequence of consecutive transitions. Such a sequence $\rho = t_1 \dots t_k$, $t_i = (s_i, s_{i+1}, x_i/y_i)$, is a *walk* with *label* $x_1/y_1, \dots, x_k/y_k$, *starting state* s_1, and *ending state* s_{k+1}. Here $x_1/y_1, \dots, x_k/y_k$ is an *input/output sequence*, also called a *global trace*, and $x_1, \dots, x_k$ is the *input portion* of this global trace. For example, M_0 has a walk $(s_2, s_3, x_2/(-, y_2))$ $(s_3, s_1, x_2/(y_1, y_2))$ and this has starting state s_2, ending state s_1, and label $x_2/(-, y_2)$ $x_2/(y_1, y_2)$ with input portion $x_2 x_2$.

FSM M defines the regular language $L(M)$ of labels of walks with starting state s_0. Similarly, we let $L_M(s)$ denote the set of labels of walks of M with starting state s. Two states s and s' of M are said to be *equivalent* if they define the same language: $L_M(s) = L_M(s')$. Similarly, two FSMs M and N are *equivalent* if $L(M) = L(N)$.

We assume that the ports of M are physically distributed, that separate agents interact with M at these ports and there is no global clock. As a result, no agent (tester) observes a global trace that occurs in testing. Instead, the agent at port p observes a *local trace*: the sequence of inputs and outputs that occur at p. Given a global trace σ and port p we will let $\pi_p(\sigma)$ denote the local trace at p, which is the projection of σ at p. Here π_p is defined by the following in which ϵ denotes the empty sequence (see, for example, [23]).

$$\begin{aligned}
\pi_p(\epsilon) &= \epsilon \\
\pi_p((x/(y_1,\ldots,y_m))\sigma) &= \pi_p(\sigma) \text{ if } x \notin X_p \wedge y_p = - \\
\pi_p((x/(y_1,\ldots,y_m))\sigma) &= x\pi_p(\sigma) \text{ if } x \in X_p \wedge y_p = - \\
\pi_p((x/(y_1,\ldots,y_m))\sigma) &= y_p\pi_p(\sigma) \text{ if } x \notin X_p \wedge y_p \neq - \\
\pi_p((x/(y_1,\ldots,y_m))\sigma) &= xy_p\pi_p(\sigma) \text{ if } x \in X_p \wedge y_p \neq -
\end{aligned}$$

Two global traces are indistinguishable if their projections are identical at each port. More formally, global traces σ_1 and σ_2 are *indistinguishable*, written $\sigma_1 \sim \sigma_2$, if for all $p \in \mathcal{P}$ we have that $\pi_p(\sigma_1) = \pi_p(\sigma_2)$. This corresponds to the assumption that the agent at each port only observes the sequence of events (local trace) at its port and so the most we can do is to later bring together these local traces. For example, if we consider global traces $\sigma_1 = x_1/(y_1, y_2)x_1/(y_1, -)$ and $\sigma_2 = x_1/(y_1, -)x_1/(y_1, y_2)$ we find that $\pi_1(\sigma_1) = x_1y_1x_1y_1$, $\pi_1(\sigma_2) = x_1y_1x_1y_1$, $\pi_2(\sigma_1) = y_2$, and $\pi_2(\sigma_2) = y_2$ and so $\sigma_1 \sim \sigma_2$. Clearly $\sim$ is an equivalence relation.

If we are testing from a DFSM M and wish to trigger a walk whose label is the global trace $\sigma \in L(M)$ then the tester at port $p \in \mathcal{P}$ has to 'implement' the projection $\pi_p(\sigma)$ of σ. Here, the tester at port p has to decide when to apply input on the basis of the observations it makes, in contrast to the situation where we wish to apply an input sequence to a single-port FSM. Let us suppose, for example, that we wish to apply the test sequence $x_1x_2x_2$ in testing from DFSM M_0 and thus to execute a walk with label $\sigma = x_1/(y_1, y_2)x_2/(-, y_2)x_2/(y_1, y_2) \in L(M_0)$. Then the tester at port 1 simply has to apply x_1 and then observe output while the tester at port 2 has to wait for y_2 before it applies x_2 twice. In Section 4 we discuss the use of strategies, which are adaptive processes, in testing.

In testing a multi-port system, we place a tester at each port and the tester at a port p only observes the interactions that occur at p. Thus, the tester at p can only know when to send an input to the SUT through the observations it makes. It is known that this restriction can lead to *controllability problems* in testing (see, for example, [5, 6, 8, 11, 12, 21, 25, 34, 35]). A controllability problem occurs when the tester at a port $p \in \mathcal{P}$ should apply an input x but cannot know when to do this based on the observations it makes. For DFSMs, this has been characterised in terms of the corresponding global trace being synchronisable.

DEFINITION 2.1. *Walk $\rho = t_1 \ldots t_k$, $t_i = (s_i, s_{i+1}, x_i/y_i)$, is* synchronisable *if for all $1 < i \leq k$ we have that the port $p \in \mathcal{P}$ at which x_i is applied satisfies: $\pi_p(x_{i-1}/y_{i-1}) \neq \epsilon$. We also say that the label of ρ is* synchronisable *and that $x_1, \ldots, x_k$ is* synchronisable from state *s_1. Given a DFSM M, test sequence $w = x_1, \ldots, x_k$ is* controllable for *M if the global trace $\sigma \in L(M)$ with input portion w is synchronisable and we simply say that w is* controllable *when M is clear.*

We use the terms synchronisable and controllable to be consistent with the literature. The term synchronisable will be used with walks and their labels while the term controllable will be used for test sequences and test strategies.

If we consider M_0, we find that the walk $(s_2, s_3, x_2/(-, y_2))(s_3, s_1, x_2/(y_1, y_2))$ is synchronisable since the second transition has input at port 2 and the first transition has label $x_2/(-, y_2)$ with $\pi_2(x_2/(-, y_2)) = x_2y_2 \neq \epsilon$. In practice, this means that the tester at port 2 knows when to apply x_2 to trigger the second transition: either after sending x_2 or receiving y_2 in response. In contrast, the walk $(s_2, s_3, x_2/(-, y_2))(s_3, s_4, x_1/(y_1, -))$ is not synchronisable since the input of the second transition is at port 1 but we have that $\pi_1(x_2/(-, y_2)) = \epsilon$. The problem here is that the tester at port 1 does not observe either the input or output of the first transition in this walk and so cannot know when to apply its input.

Essentially, if we apply a test sequence that is not controllable then the order in which the inputs are received is not predictable. As a result, there may be several different walks of the FSM that are consistent with the attempts of the local testers to apply this test sequence. Thus, even if the SUT and the testers are deterministic, the use of a test sequence that is not controllable can introduce non-determinism into testing since the composition of the SUT and testers may be non-deterministic. As a result there has been interest in producing test sequences that are controllable. However, it is known that in general we may have no controllable test sequence that achieves an objective such as testing a particular transition. For example, in M_0 we can observe that no synchronisable walk with starting state s_1 includes the transition $(s_3, s_4, x_1/(y_1, -))$ since the only transition with ending state s_3 has input at port 2 and produces no output at port 1. This is part of the motivation for the work in this paper: if we cannot achieve a test objective, such as testing a particular transition, using controllable test sequences then we need to use tests that have controllability problems and we would like to know how and when we can generate these.

This notion of controllability has been generalised to non-deterministic FSMs. It might appear that we can simply require that all possible global traces, produced in response to a test sequence, are synchronisable. However, it has been shown that this is not sufficient since we can have situations such as the following in which we are testing with test sequence $x_1x_1x_2$ and there are two possible global traces [18]:

1. a synchronisable global trace $x_1/(y_1, -)x_1/(-, y_2)x_2/(-, y_2)$; and
2. a synchronisable global trace $x_1/(y_1, y_2)x_1/(-, y_2)x_2/(y_1, -)$;

The problem is that in the first global trace the tester at port 2 should apply x_2 after receiving y_2 while in the second it should apply x_2 after receiving y_2y_2. As a result, if the tester at port 2 observes y_2 then it does not know whether to send x_2 or wait for another y_2 since it does not observe behaviour at port 1. These two scenarios are illustrated in Figures 2.2 and 2.3.

This problem only occurs if there is a port p and prefixes σ_1 and σ_2 of global traces in response to the test sequence being used such that σ_1 and σ_2 have identical projections at port p, and so cannot be distinguished by p, and yet the actions at p must differ after σ_1 and σ_2. This has been formalised in the following way [18].

Definition 2.2. *Given FSM M, a test sequence w is* controllable *for M if there do not exist $\sigma_1, \sigma_2 \in L(M)$ whose input portions are prefixes of w such that $|\sigma_1| \neq |\sigma_2|$ and the next input to be applied after σ_1 is to be applied at a port p such that $\pi_p(\sigma_1) = \pi_p(\sigma_2)$.*

3. Controllable testing from a DFSM.

In this section we adapt results in the literature to show that the problems of reaching states and distinguishing states can be solved in low-order polynomial time if we are testing from a DFSM and restricting attention to controllable test sequences [20, 23]. First we consider the problem of reaching a state of a DFSM using a controllable test sequence. The method developed in this paper is based on work in [20], which adapts an approach given in [21]. The basic idea is that we produce a DFSM $\chi_{min}(M)$ from M whose walks are exactly the synchronisable walks of M and we can then determine which states of M have corresponding reachable states in $\chi_{min}(M)$. Note that [20] defines $\chi_{min}(M)$ but does not consider its use to determine reachability.

The construction of the DFSM $\chi_{min}(M)$ is based on the following sets.

1. For state $s_i \in S$ and port $p \in \mathcal{P}$, $Depart^p(s_i) = \{(s_i, s_j, x/y) | (s_j, y) \in$

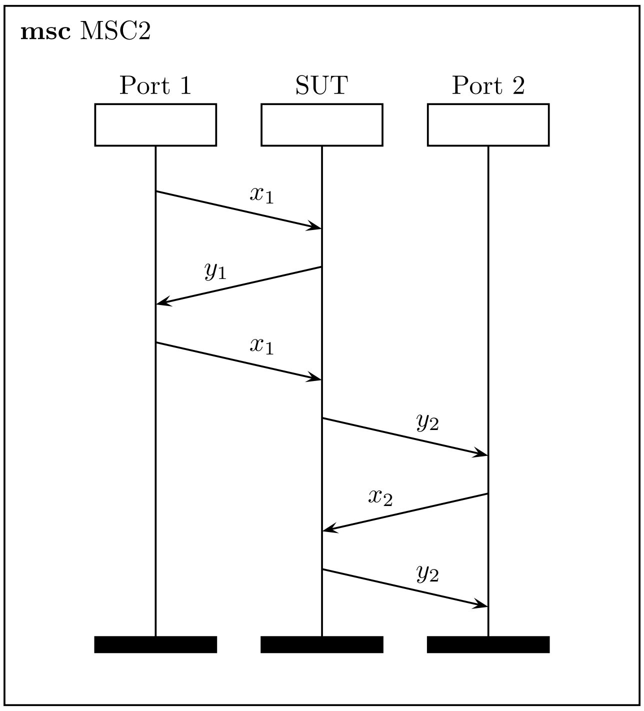

Fig. 2.2. *A first global trace involved in a controllability problem*

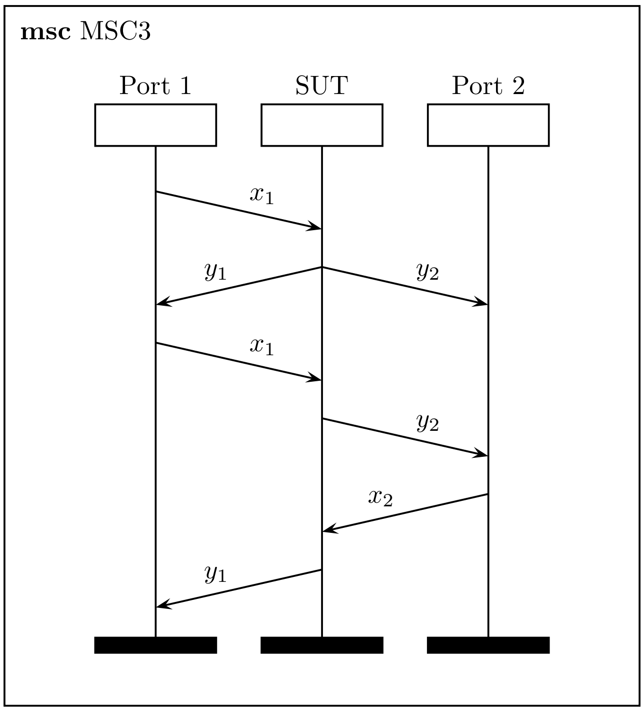

Fig. 2.3. *A second global trace involved in a controllability problem*

$h(s_i, x) \wedge x \in X_p\}$. This corresponds to the set of transitions with starting state s_i and input at p.

2. For state s_i and port set $\mathcal{P}' \subseteq \mathcal{P}$, $Arrive^{\mathcal{P}'}(s_i) = \{(s_j, s_i, x/y) \in T | \mathcal{P}' = \{p' \in \mathcal{P} | \pi_{p'}(x/y) \neq \epsilon\}\}$. This is the set of transitions of M with ending state s_i that involve the set $\mathcal{P}'$ of ports.

In a synchronisable walk, if there is a transition $t \in Arrive^{\mathcal{P}'}(s_i)$ then this can only be followed by a transition t' if $t' \in Depart^p(s_i)$ for some $p \in \mathcal{P}'$.

Consider, for example, the DFSM M_0 given in Figure 2.1. If we use the label t_{ij} to denote the transition with starting state s_i and input x_j then we have that $Depart^j(s_i) = \{t_{ij}\}$ and we also obtain the following sets.

$Arrive^{\{1,2\}}(s_1)$	$\{t_{21}, t_{32}, t_{41}\}$
$Arrive^{\{2\}}(s_1)$	$\{t_{12}\}$
$Arrive^{\{1,2\}}(s_2)$	$\{t_{11}\}$
$Arrive^{\{2\}}(s_3)$	$\{t_{22}\}$
$Arrive^{\{1\}}(s_4)$	$\{t_{31}\}$
$Arrive^{\{2\}}(s_4)$	$\{t_{42}\}$

We can now define $\chi_{min}(M) = (S', s'_0, X, Y, h')$. First, the following define the state set S' (recall that M has states set $S = \{s_1, \ldots, s_n\}$ and initial state s_0).

1. For all $1 \leq i \leq n$ and $\mathcal{P}' \subseteq \mathcal{P}$ we include $s_i^{\mathcal{P}'}$ in S' if $Arrive^{\mathcal{P}'}(s_i) \neq \emptyset$.
2. If the initial state s_0 of M corresponds to state $s_i \in S$ then state $s_i^{\mathcal{P}}$ is in S', we call this $s_0^{\mathcal{P}}$ and we set the initial state s'_0 of $\chi_{min}M$ to be $s_0^{\mathcal{P}}$.

We require $s_0^{\mathcal{P}}$ to be in S' since this represents the DFSM being in the initial state and testing having not started: the next (first) input can then be at any port. We can define h' by the following:

1. $h'(s_i^{\mathcal{P}'}, x) = \emptyset$ if $x \in X_p$ and $p \notin \mathcal{P}'$
2. $h'(s_i^{\mathcal{P}'}, x) = \{(s_j^{\mathcal{P}''}, y)\}$ if $x \in X_p$, $p \in \mathcal{P}'$, $h(s_i, x) = (s_j, y)$ and $\mathcal{P}'' = \{p \in \mathcal{P} | \pi_p(x/y) \neq \epsilon\}$.

The DFSM $\chi_{min}(M_0)$ is shown in Figure 3.1.

The DFSM $\chi_{min}(M)$ is constructed in a manner that ensures that all walks of $\chi_{min}(M)$ are synchronisable. The following results have been proved [20].

Proposition 3.1. *For each synchronisable walk ρ in M that starts at s_0, there is a unique synchronisable walk ρ' in $\chi_{min}(M)$ that starts at $s_0^{\mathcal{P}}$ such that $label(\rho) = label(\rho')$.*

Proposition 3.2. *For each walk ρ' in $\chi_{min}(M)$ that starts at $s_0^{\mathcal{P}}$, there is a unique synchronisable walk ρ in M that starts at s_0 such that $label(\rho) = label(\rho')$.*

We can now bring these together to prove the following result.

Theorem 3.3. *Given a DFSM M, there exists a test sequence w that reaches state s_i of M and that causes no controllability problems if and only if there is a set $\mathcal{P}' \subseteq \mathcal{P}$ such that the state $s_i^{\mathcal{P}'}$ can be reached from the initial state of $\chi_{min}(M)$. In addition, if M has n states and p inputs then we can decide whether such an input sequence w exists in $O(np^2)$ time.*

Proof. The first part follows from Propositions 3.1 and 3.2.

For complexity, first note that if M has T transitions then $\chi_{min}(M)$ has at most $T+1$ states and so at most $p(T+1)$ transitions. The problem has therefore been reduced to deciding reachability in a graph with at most $T+1$ vertices and $p(T+1)$ edges and this can be solved in $O((T+1) + p(T+1)) = O(p(T+1))$ time using depth-first search [36]. Further, $T = np$ and so we obtain the result. $\square$

Now, we consider the problem of distinguishing two states of a DFSM. The following definition of what it means to distinguish two states of a DFSM, while avoiding

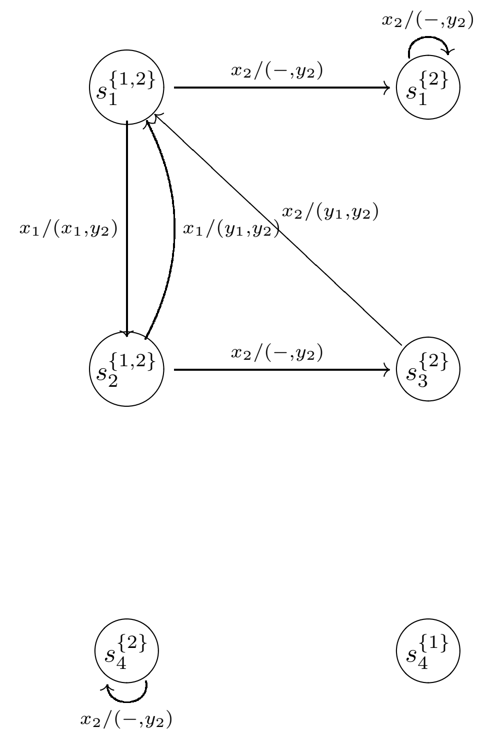

FIG. 3.1. *Finite State Machine* $\chi_{min}(M_0)$

controllability problems, is equivalent to one in [23].

DEFINITION 3.4. *Let us suppose that M is a DFSM, input sequence w is controllable from states s_1 and s_2 of M and that σ_1 and σ_2 label walks with starting states s_1 and s_2 respectively and have input portion w. We say that w* locally s-distinguishes *states s_1 and s_2 of* at port $p \in \mathcal{P}$ *if $\pi_p(\sigma_1) \neq \pi_p(\sigma_2)$. Further, w* locally s-distinguishes *states s_1 and s_2 of M if there exists a port $p \in \mathcal{P}$ such that x locally s-distinguishes s_1 and s_2 at p_i.*

This says that w must lead to no controllability problems when applied in states s_1 and s_2 and the application of w in these states must lead to a different observation at some port p.

The following results have been proved [23].

THEOREM 3.5. *Let M denote a DFSM with n states and m ports. Given states s_1, s_2 of M and port $p \in \mathcal{P}$, if s_1 and s_2 are locally s-distinguished by an input sequence starting with an element of X_p then they are locally s-distinguished by an input sequence of length at most $m(n-1)$ that starts with an element of X_p.*

THEOREM 3.6. *Given integers $n > 1$ and $m > 1$ there exists a DFSM M with n states and m ports that has locally s-distinguishable states s_1 and s_2 that cannot be locally s-distinguished by any input sequence of length less than $m(n-2)+1$.*

In addition, for a DFSM with n states and p inputs, [23] gives an $O(pn^2)$ algorithm that determines whether two states of a DFSM are locally s-distinguishable and, if they are, returns a shortest sequence that locally s-distinguishes them. Thus, we

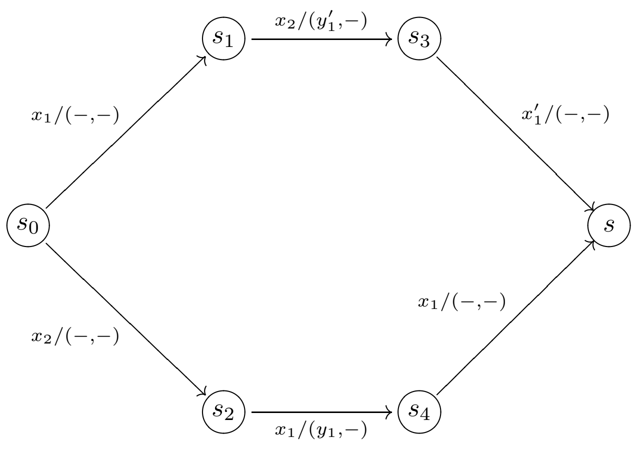

FIG. 4.1. *Reaching s requires adaptive strategies*

obtain the following result.

THEOREM 3.7. *For a DFSM M with n states and p inputs, we can decide in $O(pn^2)$ time whether there are input sequences that distinguish states s_1 and s_2 of a DFSM M and that are synchronisable from s_1 and s_2.*

4. Using strategies for testing.

Most work on testing from single-port FSMs assumes that a preset input sequence is applied. For deterministic single-port systems, we gain nothing in potential test effectiveness by using an adaptive process since as soon as the behaviour of the SUT diverges from that of M we know there has been a failure: there is no need to adapt to such situations even if this can make testing more efficient. In the context of testing from a nondeterministic FSM there has been some interest in adaptive testing (see, for example, [3, 19, 24, 37]).

The situation is different when testing a system with multiple ports. Consider, for example, the fragment of a DFSM shown in Figure 4.1 in which x_1 and x'_1 are input at port 1 and x_2 is input at port 2. Assume also that no other transitions of the DFSM can contribute to reaching state s. If we wish to reach state s then we require an adaptive process. This is because initially x_1 must be received at port 1 and x_2 must be received at port 2. The order in which x_1 and x_2 are received is unpredictable, as a result of controllability problems. However, the tester at port 1 knows that if they receive output y'_1 then they must send input x'_1 and if they receive output y_1 then they must send input x_1 and so the tester at port 1 must adapt its behaviour to the observations made. Thus, when we have multiple-ports, even when testing from a DFSM we may want testing to be adaptive.

Note that if we are testing from a DFSM M using test sequences that lead to no controllability problems then we require only a limited amount of adaptivity. This is because the test sequence w defines only one walk ρ from the initial state M and we simply take the projection of the label σ of ρ at port p in order to define the tester at p. While the tester at a port p may have to observe events at its interface in order to determine when to apply input, the actual sequence of inputs and outputs at port p is fixed. This is why, in Section 3, when we investigated controllable testing from a DFSM M we only considered single walks through M.

We will assume that each tester applies a strategy and so the *local strategy* μ_p for the tester at a port p is a mapping from its observations to the next input it should apply. It might appear that a local strategy should be a mapping from the observed sequence of outputs, at that port, to the next input. However, since the tester at p does not supply all of the inputs, there can be times in testing where the observations it has made should not be followed by input at p: the tester at p should wait for additional outputs or possibly it has finished. In addition, the tester at p might wish to send an input x' after an input x without receiving output between these values. Thus, a local strategy for the local tester at p is a partial function μ_p of type $(X_p \cup Y_p)^* \to X_p$. A *global strategy* μ is a tuple $(\mu_1, \ldots, \mu_m)$ such that for all $p \in \mathcal{P}$ we have that μ_p is a local strategy for the tester at port p.

Let us suppose that we are testing the SUT against FSM M using global strategy $\mu = (\mu_1, \ldots, \mu_m)$. Each local tester uses its local strategy in order to interact with the SUT in the following way: if the sequence of observations at p is σ_p and $\mu_p(\sigma_p)$ is defined and equals x then the local tester at p sends x to the SUT and otherwise it does nothing. The SUT receives the input of x at some later point and this triggers a transition. We assume that if the tester at port p sends inputs x and then x', the SUT must receive x before x'[1]. We can define the set of possible sequences of interactions the local tester at p can have with any SUT given local strategy μ_p.

Definition 4.1. *A sequence $\sigma \in (X_p \cup Y_p)^*$ is* an evolution *of local strategy μ_p if the following properties hold:*

1. *If $\sigma_1 x$ is a prefix of σ and $x \in X_p$ then $\mu_p(\sigma_1) = x$. This says that the local tester at p* can only *send an input x to the SUT when this is specified by the strategy.*
2. *If σ_1 is a prefix of σ and $\mu_p(\sigma_1) = x$ then $\sigma_1 x$ is prefix of σ. This says that the local tester at p* must send *an input to the SUT whenever this is specified by the strategy.*

We let $Ev(\mu_p)$ denote the set of evolutions of μ_p.

The set of possible evolutions of a local strategy μ_p defines the sequences of events that can occur at port p under μ_p. This may contain many irrelevant sequences since we can extend a sequence with arbitrary outputs, and continue doing so while the sequence is not in the domain of μ_p. However, we will not have to consider such sequences since in testing we will only observe behaviours that are consistent with both μ_p and the SUT.

We can now define the set of possible global traces that can occur when using a global strategy $\mu = (\mu_1, \ldots, \mu)$ when testing an FSM implementation N: each such global trace must be in $L(N)$ and consistent with the possible evolutions of the local strategies.

Definition 4.2. *Let us suppose that M is an FSM, N is an FSM with the same sets of ports, inputs and outputs, and $\mu = (\mu_1, \ldots, \mu_m)$ is a global strategy produced from M. Then the set of possible global traces produced in testing N with μ is denoted $\mathcal{T}r(N, \mu)$, which is defined by the following: $\mathcal{T}r(N, \mu) = \{\sigma \in L(N) | \forall p \in \mathcal{P}.\pi_p(\sigma) \in Ev(\mu_p)\}$. For state s of N we similarly define $\mathcal{T}r(N, \mu, s) = \{\sigma \in L_N(s) | \forall p \in \mathcal{P}.\pi_p(\sigma) \in Ev(\mu_p)\}$.*

This definition says that the local trace at port p must be one that could result from applying μ_p.

[1]This assumption holds if either the communications channels are first-in first-out (FIFO) or if sufficient delays are introduced between inputs at a port in testing.

If $\sigma \in \mathcal{T}r(N, \mu)$ then we must have that the tester at p should not supply any further input after observing σ.

PROPOSITION 4.3. *Let us suppose that $\mu = (\mu_1, \ldots, \mu_m)$ is a global strategy and $\sigma \in \mathcal{T}r(N, \mu)$. Then for all $p \in \mathcal{P}$ we have that μ_p is not defined on $\pi_p(\sigma)$.*

Proof. Proof by contradiction: assume that there is a port $p \in \mathcal{P}$ such that μ_p is defined on $\pi_p(\sigma)$. By the definition of $Ev(\mu_p)$, if σ_1 is a prefix of σ and $\mu_p(\sigma_1) = x$ then $\sigma_1 x$ is prefix of σ. However, σ is a prefix of σ and so if $\mu_p(\sigma) = x$ then σx is a prefix of σ. This provides a contradiction as required. $\Box$

This says that if $\sigma \in \mathcal{T}r(N, \mu)$ then after σ none of the local strategies is able to supply additional input. In addition, since $\mathcal{T}r(N, \mu) \subseteq L(N)$, N cannot produce additional output without receiving input. Thus, if $\sigma \in \mathcal{T}r(N, \mu)$ then testing stops if the global trace σ occurs when testing N with μ.

Observe, however, that even if $\sigma \in \mathcal{T}r(N, \mu)$, μ_p may be defined on a local trace σ' such that $\pi_p(\sigma)$ is a (proper) prefix of σ'.

Now consider the problem of reaching the state s in the fragment of the FSM shown in Figure 4.1. We could choose the following local strategies μ_1 and μ_2 for the testers at ports 1 and 2 respectively:

1. Under μ_1 initially the input x_1 is applied. If y_1' is observed then the next input is x_1' and if y_1 is observed then the next input is x_1. Thus, μ_1 maps ϵ to x_1, $x_1 y_1'$ to x_1', and $x_1 y_1$ to x_1.
2. Under μ_1 initially the input x_2 is applied. Thus, μ_2 maps ϵ to x_2.

Thus, the global strategy (μ_1, μ_2) leads to the FSM moving to state s.

We now formally define what it means for a global strategy to reach a state and then for it to distinguish two states.

DEFINITION 4.4. *Global strategy μ reaches the state s of M if for all $\sigma \in \mathcal{T}r(M, \mu)$ we have that every walk of M from state s_0 with label σ has ending state s.*

DEFINITION 4.5. *A global strategy μ* distinguishes *states s and s' of FSM M if for all $\sigma \in \mathcal{T}r(M, \mu, s)$ and $\sigma' \in \mathcal{T}r(M, \mu, s')$ we have that $\sigma \not\sim \sigma'$. Similarly, global strategy μ* distinguishes *FSMs M_1 and M_2 if for all $\sigma \in \mathcal{T}r(M_1, \mu)$ and $\sigma' \in \mathcal{T}r(M_2, \mu)$ we have that $\sigma \not\sim \sigma'$.*

This says that when using global strategy μ, if σ is a possible global trace produced from state s then it is distinguishable from every global trace that can be produced from using μ in state s'.

5. Multiplayer team games with incomplete information.

We are interested in problems in which we have a model of the expected behaviour of the SUT and wish to devise strategies for local testers that interact with the SUT. We focus on two objectives: reaching states and distinguishing states. These problems can be seen in terms of games in which one player is the SUT and the other players are the testers. We therefore have two teams: the player representing the SUT forms one team and the other players (testers) are in the other team. The team that contains the testers wins if the objective is achieved and otherwise the other team, formed by the SUT, wins.

Alur et al. [3] express the problems of reaching and distinguishing states of a single-port non-deterministic FSM in terms of two player games. They then use the result that the problem of deciding whether there is a winning strategy for a two player game with incomplete information is EXPTIME-complete [33]. In two player games we can have the players take turns to make moves and this has led to the notion of alternation and corresponding computational models (see, for example, [7, 28, 29, 30]). The game ends if, after a sequence of moves, one player has achieved a winning position. It is also normal for one player to be designated the $\forall$ player and the other player to be the $\exists$ player. The basic idea is that we are interested in whether the $\exists$ player has a winning strategy: one under which it is guaranteed to win irrespective of the moves of the $\forall$ player. Thus, at each point in the game, if it is the turn of the $\exists$ player then we consider one move (the one it chooses) while if it is the turn of the $\forall$ player then we need to consider all allowed moves. Such games are called $\exists\forall$ games and they nicely fit the situation in which we are, for example, trying to force a single-port SUT into a particular state since the SUT is the $\forall$ player and the tester is the $\exists$ player.

As noted above, in contrast to single-port FSMs, the problems of reaching and distinguishing states of an FSM that has multiple ports are similar to multiplayer games. In this section we describe *multiplayer team games* and the result that we will use. In a multiplayer team game there are two teams, which we call Black and White in order to be consistent with [10]. A team wins if one of its players wins and a team loses if one of the players from the other team wins.

Much of the work in game theory uses an approach in which moves synchronise: the players make moves simultaneously. However, in our context a tester applies an input and the SUT responds by (possibly) changing state and can also produce output. Thus, it is more natural to consider games in which turns alternate in some manner. In two player games it is natural to require the players to take turns but it is less obvious how the next player should be determined in multiplayer games. One possibility is to include a turn variable in a game, the value of this variable identifying the player whose turn it now is [28, 29, 30]. An alternative described by Demaine and Hearn [10] is to order the players and require that the turns cycle through the players: if there are i players then we start with player 1, then player 2 etc. and once player i has made a move we repeat this process/order. Since we use a result due to Demaine and Hearn [10] we also require that moves cycle in this manner. We could also consider games in which the next player to move is chosen randomly. However, we would require a little more structure since the tester to next apply an input might be chosen randomly but the SUT should have the next move.

Given a multiplayer game, there is the problem of deciding whether a particular team has a winning strategy: one that is guaranteed to lead to it reaching a winning state. It has been stated that for a particular multiplayer game, called TEAM-PRIVATE-PEEK, this problem is undecidable [30]. More recently, however, Demaine and Hearn argue that TEAM-PRIVATE-PEEK is decidable but that the general problem of deciding whether a team has a winning strategy is undecidable [10]. They achieve this by adapting the approach of [30] and using a type of game that they call *Constraint Logic*, which we now describe.

The games described by Demaine and Hearn [10] use what they call constraint graphs. A constraint graph is based on a graph $G = (V, E)$ in which V is a finite set of vertices and E is a finite set of (undirected) edges. Each vertex and edge is given a weight 1 (red) or 2 (blue). At any point in the game every edge has a current direction and the inflow of a vertex is the sum of the weights of the edges directed to that vertex. A configuration defines a direction for each edge and is a *legal configuration* if for every vertex v the inflow of v is at least the weight of v. An AND/OR constraint graph is a constraint graph in which every vertex has one of the following forms

1. It is an AND vertex: it has minimum inflow 2 and is incident with three edges with weights 1, 1, 2.

2. It is an OR vertex: it has minimum inflow 2 and is incident with three edges all with weight 2.

It is argued that AND vertices and OR vertices are similar to AND gates and OR gates [10] but this is not important here. For us, what is important is that an AND/OR constraint graph has a finite set of legal configurations. In addition, the following problem has been defined [10].

Definition 5.1 (Team Private Constraint Logic (TPCL)). *Let us suppose that $G = (V, E)$ denotes an AND/OR constraint graph, N and k are integers, we have a partition of $1, \ldots, N$ into nonempty sets W and B and for all $1 \leq i \leq N$ we have sets $E_i \subset E'_i \subset E$ and edge e_i. The TPCL problem is: Does White have a winning strategy for the following game?*

Players $1, \ldots, N$ take turns in the given order. For $1 \leq i \leq N$, player i only sees the direction of edges in E'_i and can change the direction of at most k edges from E_i. A move by i must be known to be legal based on the knowledge of the directions of the edges in E'_i. Player i wins if it reverses the direction of edge e_i and White wins if a player from W wins.

The following results is proved in [10] (Theorem 9).

Theorem 5.2. *It is undecidable whether White has a winning strategy for TPCL even with $N = 3$ players.*

The proof of this result is based on a result that shows that if a particular type of 3 player game is decidable then it is also decidable whether a Turing machine M_T terminates in an accepting state when starting with the empty tape. In the game, each White player W_i ($i \in \{1, 2\}$) is required to provide two sequences U_i and V_i of configurations[2] of M_T and does so one value at a time, with the subsequences representing the configurations being separated by a special symbol #. W wins if a sequence of configurations provided reaches a terminating configuration in which M_T is in an accepting state and B fails to demonstrate one of the following: one of the four sequences of configurations provided is not a legal computation of M_T starting with an empty tape or $U_1 \neq U_2$ or $V_1 \neq V_2$. At each point in the game, B chooses whether the next value provided by W_1 is from U_1 or V_1 and also whether the next value provided by W_2 is from U_2 or V_2. Player B can nondeterministically choose to check any one of the conditions under which it wins. For example, it can check that parts of U_1 and U_2 are consistent or that parts of V_1 and V_2 are consistent by simply receiving these parts in the same turn and comparing them. It can also, for example, let U_2 run one configuration ahead of U_1 and check that the configuration in U_2 is consistent with having taken one move of M_T from the configuration of U_1. Importantly, each check can be made in constant space and in order for W to have a winning strategy, under that strategy it cannot fail any of these checks since it does not know which are being applied. As a result, W has a winning strategy for this game if and only if it is possible to provide sequences U and V of computations of M_T in which M_T starts with an empty tape and terminates in an accepting state. Thus, W has a winning strategy if and only if M_T terminates in an accepting state when starting with the empty tape.

We use the following special case.

Definition 5.3 (3 Player Team Private Constraint Logic (3TPCL)). *Let us suppose that $G = (V, E)$ denotes an AND/OR constraint graph, k is an integer, we have sets $W = \{W_1, W_2\}$ and $B = \{B_3\}$ in which W_i is player i ($i \in \{1, 2\}$) and B_3*

[2]A configuration is a sequence of values representing the current state of M_T and the non-empty part of the tape of M_T.

is player 3 and for all $1 \leq i \leq 3$ we have sets $E_i \subset E'_i \subset E$ and edge e_i. The 3TPCL problem is: Does White have a winning strategy for the following game?

Players 1, 2, and 3 take turns in the given order. For $1 \leq i \leq 3$, player i only sees the direction of edges in E'_i and can change the direction of at most k edges from E_i. A move by i must be known to be legal based on the knowledge of the directions of the edges in E'_i. Player i wins if it reverses the direction of edge e_i and White wins if a player from W wins.

We will show that if the 3TPCL problem can be solved then the TPCL problem can also be solved.

THEOREM 5.4. *It is undecidable whether White has a winning strategy for 3TPCL.*

Proof. Proof by contradiction: assume that it is decidable whether White has a winning strategy for 3TPCL. We now prove that this must mean that it is decidable whether White has a winning strategy for TPCL with three players.

Assume we have an instance of the TPCL problem with three players defined by: AND/OR constraint graph $G = (V, E)$, integer k, a partition of $\{1, 2, 3\}$ into nonempty sets W and B, and for all $1 \leq i \leq 3$ sets $E_i \subset E'_i \subset E$ and edge e_i. As has been noted elsewhere [28, 29, 30], problems in which White has only one player can be represented as two player problems, for which the outcome problem is decidable, and so we can assume that White has two players and Black has one player.

We have three cases.

1. Players 1 and 2 are in W and so this is an instance of the 3TPCL and so we can decide whether White has a winning strategy.
2. Player 1 is in B. For each allowed move of player 1, we obtain an instance of the 3TPCL and White has a winning strategy if and only if it has a winning strategy for all of these instances of the 3TPCL. Thus, since it is decidable whether White has a winning strategy for 3TPCL and player 1 has a finite set of moves, we can decide whether White has a winning strategy for this game.
3. Player 2 is in B. For each allowed move of player 1 we obtain an instance of the TPCL in which Black moves first. White has a winning strategy if and only if it has a winning strategy for one of these instances of the TPCL and we have already seen that we can decide this. Thus, since player 1 has a finite set of moves, it is decidable whether White has a winning strategy.

In each case, it is decidable whether White has a winning strategy but this contradicts Theorem 5.2. $\square$

6. Nondeterministic FSMs.

Alur et al. [3] showed that the problem of deciding whether there is a strategy that reaches a given state of a nondeterministic single-port FSM is EXPTIME-complete. They achieve this by showing that the outcome problem for a $\exists\forall$ game can be converted into a problem of finding a strategy to reach a state of a nondeterministic single-port FSM. Instead, we show that we can represent an instance of the 3TPCL problem in terms of reaching a state of a multi-port FSM.

THEOREM 6.1. *The following problem is undecidable: Given a multi-port FSM M and a state s_r of M, is there a global strategy that reaches s_r.*

Proof. We assume that an instance of the 3TPCL has been given and show that if we can decide reachability for multi-port FSMs then we can also solve this instance of the 3TPCL.

We will define a multi-port FSM M with $m = 2$ ports. Each state of M will be a legal configuration of the 3TPCL combined with a value for a variable t whose value can be 1 or 2. Here the value 1 for t will denote the situation in which W_1 is allowed to move and port 1 can apply the next input, while 2 will denote the situation in which W_2 is allowed to move and port 2 can supply the next input. We will have two additional states: s_r is the state that will represent W winning while s_e will denote an 'error state' from which we cannot reach s_r. In states s_r and s_e, the response of M to input is to stay in the same state and produce no output. For W_1 and W_2, we will construct M so that if input is received at port $i \in \{1, 2\}$ when $t \neq i$ then W cannot win (s_r cannot be reached) and this is achieved by moving to state s_e. Thus, in order to reach s_r a tester corresponding to W_1 or W_2 can only apply input when it is the turn of the corresponding player. Since the 3TPCL has a finite set of configurations, M has a finite set of states.

We now define the response of M to inputs when not in states s_r or s_e. If input is received at port 1 when t is not 1 or input is received at port 2 when t is not 2 then M moves to state s_e, producing no output. From here we cannot reach s_r and so if we wish to reach s_r we must avoid such situations.

Each potential move for W_i ($i \in \{1, 2\}$) is represented by an input at i. There are a finite number of possible moves and so the input sets are finite. If $t = i$ and an input is received at i then there are four possibilities:

1. The input represents an allowed winning move for player i in the current state of the game, as represented by the state of M. Then M changes its state to s_r and sends unique output win to ports 1 and 2 to indicate that W has won.
2. The input represents an illegal move for player i in the current configuration of the game, as represented by the state of M. Then M moves to state s_e and produces no output.
3. The input represents a legal non-winning move for player 1 in the current configuration of the game (so $t = 1$). Then M changes its state to represent the configuration of the game after the move and the value of t is increased to 2. M sends the following output to each port/player j, $j \in \{1, 2\}$: the current state of the part of the constraint graph that corresponds to the set E'_j visible to W_j and the new value of t.
4. The input represents a legal non-winning move for player 2 in the current configuration of the game (so $t = 2$). If B_3 has a winning move after this then M moves to state s_e and produces no output. Otherwise M non-deterministically changes its state to represent the configuration of the game after a sequence of two moves: the move represented by the input followed by a legal move for B_3. In addition, the value of t is changed to 1. M sends the following output to each port/player j, $j \in \{1, 2\}$: the current state of the part of the constraint graph that corresponds to the set E'_j visible to W_j and the new value of t.

These transitions allow us to model the moves of all of the players: input at 1 models a move of W_1 and input at 2 models a move of W_2 and, if this is not winning, a following move by B_3.

The above rules define an FSM M. Now observe that a sequence of inputs takes M to state s_r if and only if the corresponding sequence of moves leads to W winning. Thus, there is a strategy that takes M to state s_r if and only if W has a winning strategy for this game and this can be done for any instance of the 3TPCL. The result thus follows from Theorem 5.4. □

We now prove that if we can decide whether there are strategies to distinguish states then we can also decide whether there are strategies to reach states and thus the second problem, of distinguishing states, is also undecidable.

THEOREM 6.2. *The following problems are undecidable:*

1. *Given FSMs M' and M'', is there a global strategy that distinguishes between M' and M''.*
2. *Given an FSM M and two states s_1 and s_2 of M, is there a global strategy that distinguishes between s_1 and s_2.*

Proof. We prove the first result by reducing the problem of reaching states to this, using Theorem 6.1 that tells us that it is undecidable whether there is a strategy to reach a given state of an FSM. Let us suppose that we have a state s_r of FSM M with m ports and state set $S = \{s_1, \dots, s_n\}$ and initial state $s_0 \in S$. Then for each port $p \in \mathcal{P}$ we introduce a new input $x'_p \notin X_p$ and a new output $y'_p \notin (Y_p \cup \{-\})$ and define FSMs M' and M'':

1. M' is identical to M except that for every state s of M and port $p \in \mathcal{P}$ we add a transition from s to s with input x'_p at p and no output.
2. M'' is formed from two copies M_1 and M_2 of M'. We let $S = \{s_1, \dots, s_n\}$ be the set of states of M_1 and $S' = \{s'_1, \dots, s'_n\}$ be the set of states of M_2. M_2 is isomorphic to M' and thus a state s'_i of M_2 cannot be distinguished from the corresponding state s_i of M'. M_1 is identical to M' except for the transitions with input $x'_1, \dots, x'_m$: if M'' receives x'_p at p when in state s_r then it produces output $(y'_1, \dots, y'_m)$ and from every other state $s_i \neq s_r$ of M_1 the transition with input x_p takes M'' to state s'_i of M_2 and produces no output. The initial state of M'' is s_0.

A strategy can only distinguish between M' and M'' if it leads to the input of some x'_p when M'' *must* be in state s_r since: M' and M'' produce identical behaviour unless an x'_p is input and if an x'_p is input when M'' is in a state other than s_r then M' and M'' produce identical behaviour in response to x'_p and then are in equivalent states and so cannot be distinguished. Thus, if $\mu = (\mu_1, \dots, \mu_m)$ is a global strategy that distinguishes M' and M'' then we can produce a global strategy $\mu' = (\mu'_1, \dots, \mu'_m)$ from μ by: μ'_p is μ_p restricted to sequences from $(X_p \cup Y_p)^*$. Since μ distinguishes between the initial states of M' and M'', the application of μ' to M must be guaranteed to lead to M being in state s_r and so μ' must reach s_r in M. Thus, the problem of deciding whether there is a strategy that reaches a given state s_r can be reduced to the problem of deciding whether there is a strategy that distinguishes two states and so the result follows.

The second result follows from the observation that the problem of distinguishing between two FSMs can be expressed as a problem of distinguishing between the initial states of these FSMs. $\square$

Note that in the FSM constructed in the proof of Theorem 6.1, in any strategy that reaches state s_r we have that each port receives output in response to each input provided. Thus, there are no controllability problems. As a result, in contrast to the situation with DFSMs, we know that it is generally undecidable whether there is a strategy to reach a state of an FSM or to distinguish two states of an FSM even if we restrict testing to strategies that have no controllability problems.

7. Deterministic FSMs. In this section we investigate the problems of reaching and distinguishing states of a DFSM. Initially it might appear that results regarding multiplayer games cannot be applied since they require us to consider all possible moves for the players in B and in order to represent this it seems that we need the FSM to be nondeterministic. However, we have seen that when there are controllability problems the combination of testers with a DFSM can be nondeterministic and we will use this to simulate non-determinism and thus the possible moves for players in B.

We now investigate the problem of deciding whether a global strategy exists to reach a given state s of M.

THEOREM 7.1. *The following problem is undecidable: Given a multi-port DFSM M and a state s_r of M, is there a global strategy that reaches s_r.*

Proof. We assume that an instance of the 3TPCL has been given and show that if we can decide reachability for multi-port FSMs then we can also solve this instance of the 3TPCL.

We will define a multi-port FSM M with $m \geq 2$ ports. Each state of M will represent a legal configuration of the 3TPCL combined with a value for a variable t whose value can be 1, 2, or 3. Here the value 1 will denote the situation in which W_1 is allowed to move and port 1 can apply the next input, while 2 will denote the situation in which W_2 is allowed to move and port 2 can supply the next input. If t has the value 3 then we will be simulating a move of B_3. We will have two additional states: s_r is the state that will represent W winning while s_e will denote an 'error state' from which we cannot reach s_r. In states s_r and s_e, the response of M to input is to stay in the same state and produce no output. For W_1 and W_2, we will construct M so that if input is received at port $i \in \{1, 2\}$ when $t \neq i$ then W cannot win (s_r cannot be reached) and this is achieved by moving to state s_e. Thus, in order to reach s_r the testers corresponding to W_1 and W_2 can only apply input when it is the turn of the corresponding player.

We now define the response of M to inputs when not in states s_r or s_e. If input is received at port 1 when t is not 1 or input is received at port 2 when t is not 2 then M moves to state s_e, producing no output.

Each potential move for W_i ($i \in \{1, 2\}$) is represented by an input at i. If $t = i$ with $i \in \{1, 2\}$ and an input is received at i then there are three possibilities:

1. The input represents an allowed winning move for player i in the configuration of the game, as represented by the state of M. Then M changes its state to s_r and sends unique output win to every port to indicate that W has won.
2. The input represents a legal non-winning move for player i in the current configuration of the game, as represented by the state of M. If $t = 2$ and in the resultant configuration B_3 has a winning move then M moves to state s_e and produces no output. Otherwise, M changes its state to represent the configuration of the game after the move and the value of t is increased by 1. M sends the following output to each port/player j, $j \in \{1, 2\}$: the current part of the constraint graph that corresponds to the set E'_j that is visible to W_j and the new value of t. If the new value of t is 3 then M also sends the value 3 to the ports $3, \ldots, m$.
3. The input represents an illegal move for player i in the current configuration of the game, as represented by the state of M. Then M moves to state s_e and produces no output.

These transitions allow us to model the moves of W_1 and W_2 and for the ports 1 and 2, at which these moves are input, to know when to make moves. It remains to represent the moves of player B_3. We cannot simply include these moves within the move of W_2 since there may be many possible moves for B_3 but M is deterministic. Instead, when $t = 3$ we require input of a (fixed) value at each one of the ports $3, \ldots, m$. In a given state, we map each order in which inputs can be received from the ports, which represents a permutation of the values $3,\dots,m$, to an allowed move of B_3 in the current configuration and we ensure that all allowed moves of B_3 are included. We can represent all allowed moves of B_3 in this way, by including a walk of M for each order, as long as we choose a sufficiently large (but finite) value of m. Specifically, it is sufficient to choose a value of m such that $(m-2)!$ is at least the number of configurations of the game. Once M has received all of these inputs, one from each port $3,\dots,m$, it changes the state to the one that corresponds to the new game configuration and sets $t=1$. Again, M also sends the following output to port j, $j \in \{1,2\}$: the current state of the part of the constraint graph that corresponds to E'_j and the new value of t (which is 1). The last transition, of a walk representing a move of B_3, is responsible for changing the value of t and sending the output. If M receives more than one input at a port in $3,\dots,m$ before receiving one from each port or it receives input from one of these ports when $t \neq 3$ then it moves to the state s_e. Thus, in order to change the state of M when $t=3$ it is necessary for each port $3,\dots,m$ of M to receive exactly one input and the order in which these inputs are received defines the (allowed) move of B_3, which is then simulated by M.

The above rules define an FSM M. Now observe that a sequence of inputs takes M to state s_r if and only if the corresponding sequence of moves, with the inputs at ports $3,\dots,m$ removed, leads to W winning. Thus, there is a strategy that takes M to state s_r if and only if W has a winning strategy for this game and this can be done for any instance of the 3TPCL. The result thus follows from Theorem 5.4. $\square$

The proof of the following is the same as that of Theorem 6.2 except that is builds on Theorem 7.1 rather than Theorem 6.1.

THEOREM 7.2. *The following problems are undecidable:*

1. *Given DFSMs M' and M'', is there a global strategy that distinguishes between M' and M''.*
2. *Given DFSM M and two states s_1 and s_2 of M, is there a global strategy that distinguishes between s_1 and s_2.*

8. Conclusions. Some systems have physically distributed interfaces, called ports, at which they interact with their environment. In testing we place a tester at each port and a tester at a port p observes only the events that occur at p. It is known that this can introduce controllability and observability problems into testing and there is the challenge of producing general test generation methods for testing from multi-port finite state machines (FSMs).

Techniques for testing from a single-port FSM typically use sequences that reach and distinguish the states of the FSM and such sequences can be produced in low-order polynomial time. This suggests that if we wish to develop methods for testing from an FSM M that has multiple ports then we might use sequences that achieve this for M. In this paper we have shown that in order to reach and distinguish states of a multi-port FSM M we may need to apply an (adaptive) strategy, rather than a preset test sequence, at each port and have considered the problem of deciding whether such strategies exist.

The main results of the paper are that the problems of deciding whether there is an adaptive strategy that reaches a given state or that distinguishes two particular states are undecidable. This is true for both nondeterministic and deterministic FSMs. These results have important practical ramifications. First, if we produce a test generation method that requires strategies to reach and distinguish states of a multi-port FSM then we know that it cannot be a general test generation method. In addition, the problem of distinguishing two multi-port FSMs in testing corresponds to distinguishing their initial states. As a result, it is undecidable whether two FSMs can be distinguished and this shows that we cannot expect to be able to find general test generation methods that produce test sequences that distinguish between a multi-port FSM M and a finite set of faulty machines. This contrasts with the situation for single-port FSMs, where it is possible to produce test sequences that distinguish between a single-port FSM specification M and a faulty FSM M' and so, given such a single-port FSM M and integer r, it is possible to produce a set of test sequences that distinguish between M and all faulty FSMs with at most r states (see, for example, [9, 14, 17, 19, 22, 32, 31]). Such sets are said to be r-complete and the results in this paper suggest that we cannot expect to be able to produce general algorithms that take a multi-port FSM M and return an r-complete set of test sequences. However, we have also proved that the problems of reaching and distinguishing states can be solved in low order polynomial time when testing from a deterministic FSM and restricting attention to test sequences that cause no controllability problems. It may therefore be possible to develop test generation algorithms that return r-complete sets of test sequences.

There are several possible directions for future work. First, there may be significant classes of systems for which the problems of reaching and distinguishing states can be solved and it would be useful to identify such classes and develop corresponding algorithms. In addition, there is the challenge of devising test generation techniques for multi-port FSMs that do not depend on strategies to reach and distinguish states. There has been recent work that considers models in which a transition can be triggered by multiple inputs [16]. Naturally, since this is a richer formalism than multi-port FSMs, the negative decidability results developed in this paper also hold for the richer class of models. However, there would be value in investigating the problem of generating tests for such models since it is likely to be quite different from that of generating tests for multi-port FSMs.

REFERENCES

[1] H. AboElFotoh, O. Abou-Rabia, and H. Ural, *A test generation algorithm for protocols modeled as non-deterministic FSMs*, The Software Engineering Journal, 8 (1993), pp. 184–188.

[2] A. V. Aho, A. T. Dahbura, D. Lee, and M. U. Uyar, *An optimization technique for protocol conformance test generation based on UIO sequences and Rural Chinese Postman Tours*, in Protocol Specification, Testing, and Verification VIII, Atlantic City, 1988, Elsevier (North-Holland), pp. 75–86.

[3] R. Alur, C. Courcoubetis, and M. Yannakakis, *Distinguishing tests for nondeterministic and probabilistic machines*, in 27th ACM Symposium on Theory of Computing, 1995, pp. 363–372.

[4] M. Barnett, W. Grieskamp, L. Nachmanson, W. Schulte, N. Tillmann, and M. Veanes, *Towards a tool environment for model-based testing with AsmL*, in Formal Approaches to Testing, vol. 2931 of Lecture Notes in Computer Science, Montreal, Canada, 2003, Springer-Verlag, pp. 252–266.

[5] S. Boyd and H. Ural, *The synchronization problem in protocol testing and its complexity*, Information Processing Letters, 40 (1991), pp. 131–136.

[6] L. Cacciari and O. Rafiq, *Controllability and observability in distributed testing*, Information and Software Technology, 41 (1999), pp. 767–780.

[7] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer, *Alternation*, Journal of the ACM, 28 (1981), pp. 114–133.

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

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

[10] Erik D. Demaine and Robert A. Hearn, *Constraint logic: A uniform framework for modeling computation as games*, in 23rd Annual IEEE Conference on Computational Complexity (CCC 2008), 2008, pp. 149–162.

[11] 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.

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

[13] E. Farchi, A. Hartman, and S. Pinter, *Using a model-based test generator to test for standard conformance*, IBM systems journal, 41 (2002), pp. 89–110.

[14] G. Gonenc, *A method for the design of fault detection experiments*, IEEE Transactions on Computers, 19 (1970), pp. 551–558.

[15] W. Grieskamp, Y. Gurevich, W. Schulte, and M. Veanes, *Generating finite state machines from abstract state machines*, in Proceedings of the ACM SIGSOFT Symposium on Software Testing and Analysis, 2002, pp. 112–122.

[16] Stefan Haar, Claude Jard, and Guy-Vincent Jourdan, *Testing input/output partial order automata*, in (TestCom/FATES 2007), vol. 4581 of Lecture Notes in Computer Science, Springer, 2007, pp. 171–185.

[17] 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, November 1964, pp. 95–110.

[18] R. M. Hierons, *Controllable testing from nondeterministic finite state machines with multiple ports*, submitted.

[19] ———, *Testing from a non-deterministic finite state machine using adaptive state counting*, IEEE Transactions on Computers, 53 (2004), pp. 1330–1342.

[20] Robert M. Hierons, *Canonical finite state machines for distributed systems*, Theoretical Computer Science, 411 (2010), pp. 566–580.

[21] R. M. Hierons and H. Ural, *Synchronized checking sequences based on UIO sequences*, Information and Software Technology, 45 (2003), pp. 793–803.

[22] Robert M. Hierons and Hasan Ural, *Optimizing the length of checking sequences*, IEEE Transactions on Computers, 55 (2006), pp. 618–629.

[23] ———, *The effect of the distributed test architecture on the power of testing*, The Computer Journal, 51 (2008), pp. 497–510.

[24] D. Lee and M. Yannakakis, *Principles and methods of testing finite-state machines - a survey*, Proceedings of the IEEE, 84 (1996), pp. 1089–1123.

[25] 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.

[26] G. L. Luo, G. v. Bochmann, and A. Petrenko, *Test selection based on communicating nondeterministic finite-state machines using a generalized Wp-method*, IEEE Transactions on Software Engineering, 20 (1994), pp. 149–161.

[27] E. P. Moore, *Gedanken-experiments*, in Automata Studies, C. Shannon and J. McCarthy, eds., Princeton University Press, 1956.

[28] G. Peterson, J. Reif, and S. Azhar, *Lower bounds for multiplayer noncooperative games of incomplete information*, Computers and Mathematics with Applications, 41 (2001), pp. 957–992.

[29] ———, *Decision algorithms for multiplayer noncooperative games of incomplete information*, Computers and Mathematics with Applications, 43 (2002), pp. 179–206.

[30] Gary L. Peterson and John H. Reif, *Multiple-person alternation*, in 20th Annual Symposium on Foundations of Computer Science (FOCS 79), IEEE, 1979, pp. 348–363.

[31] Alexandre Petrenko and Nina Yevtushenko, *Testing from partial deterministic FSM specifications*, IEEE Transactions on Computers, 54 (2005), pp. 1154–1165.

[32] 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, 9–11 September 1996, Chapman and Hall, pp. 125–141.

[33] John H. Reif, *The complexity of two-player games of incomplete information*, Journal of Computer and System Sciences, 29 (1984), pp. 274–301.

[34] B. Sarikaya and G. v. Bochmann, *Synchronization and specification issues in protocol testing*, IEEE Transactions on Communications, 32 (1984), pp. 389–395.

[35] K.-C. Tai and Y.-C. Young, *Synchronizable test sequences of finite state machines*, Computer Networks and ISDN Systems, 30 (1998), pp. 1111–1134.

[36] R. E. Tarjan, *Depth-first search and linear graph algorithms*, SIAM Journal of Computing, 1 (1972).

[37] P. Tripathy and K. Naik, *Generation of adaptive test cases from non-deterministic finite state models*, in Proceedings of the 5th International Workshop on Protocol Test Systems, Montreal, September 1992, pp. 309–320.