Oracles for Distributed Testing

Robert M. Hierons *Senior Member, IEEE*

R.M. Hierons is with the School of Information Systems, Computing, and Mathematics, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK

Abstract

The problem of deciding whether an observed behaviour is acceptable is the *oracle problem*. When testing from a finite state machine (FSM) it is easy to solve the oracle problem and so it has received relatively little attention for FSMs. However, if the system under test has physically distributed interfaces, called ports, then in distributed testing we observe a local trace at each port and we compare the set of local traces with the set of allowed behaviours (global traces). This paper investigates the oracle problem for deterministic and non-deterministic FSMs and for two alternative definitions of conformance for distributed testing. We show that the oracle problem can be solved in polynomial time for the weaker notion of conformance ($\sqsubseteq_w$) but is NP-hard for the stronger notion of conformance ($\sqsubseteq_s$), even if the FSM is deterministic. However, when testing from a deterministic FSM with controllable input sequences the oracle problem can be solved in polynomial time and similar results hold for nondeterministic FSMs. Thus, in some cases the oracle problem can be efficiently solved when using $\sqsubseteq_s$ and where this is not the case we can use the decision procedure for $\sqsubseteq_w$ as a sound approximation.

Index Terms

D2.4: Software Engineering/Software/Program Verification, D2.5: Software Engineering/Testing and Debugging, H.3.4 [Systems and Software]: Distributed systems, finite state machine, nondeterminism, test oracle, controllability, local observability.

I. INTRODUCTION

There is increasing interest in and use of distributed systems. Some of these systems have physically distributed interfaces, often called ports, and an agent at a port p only observes the sequence of interactions that occur at p, this being called a local trace. Examples of such systems include web services but also cloud computing. As a result of there being physically distributed ports, no individual agent observes the global trace of the system and a set of local traces can be consistent with several global traces. The presence of distributed ports can thus have a significant impact on testing (see, for example, [1], [2], [3], [4], [5], [6], [7], [8]). Typically, systems with distributed ports are state-based and state-based systems are usually specified using languages based on finite state machines (FSMs) [9], [10], [11], [12], [13], [14], [15], [16] or input output transition systems [17]. This has led to interest in testing systems that have distributed interfaces and are specified using FSMs [18], [9], [2], [3], [13], [4], [5], [19], [7], [8] and, more recently, input output transition systems [20], [21].

In this paper we are interested in *black-box testing*, in which only inputs and outputs are observed. When testing a *system under test (SUT)* it is necessary to check that an observed behaviour is consistent with the requirements or specification and this is called the *oracle problem*. Ideally, we have an automated oracle and in many cases it is sufficient to use a model or specification from which the SUT was developed. In this paper we assume that there is an FSM model of the SUT. Normally this makes the oracle problem trivial since we check that an observed trace is a trace of the model and this can be done in low order polynomial time. However, if the SUT has physically distributed ports then we obtain different conformance relations since the observation made is a set of local traces, one at each port, rather than a global trace. As a result, it is no longer sufficient to check that a (global) trace is a trace of the model. Instead, we need to check that the set of observations (local traces) is consistent with the specification.

It has been known for over 20 years that the presence of physically distributed ports introduces additional controllability and observability problems into testing and these can limit the effectiveness of testing [2]. Let us suppose that we intend to apply input sequence x_1x_2 when FSM M is in state s, x_1 is input at port p, and x_2 is input at $q \neq p$. If, when in state s, M does not send output to q in response to x_1 then the tester at q cannot know when to send x_2. This creates a controllability problem as illustrated in MSC1 in Figure 1 in which each vertical line represents a timeline, time progressing as we move down a line. A controllability problem exists when a tester is required to send an input but was not involved in the previous transition and so does not know when to send this input. If a sequence of transitions does not have this problem it is *controllable*. However, there may be no controllable sequence that satisfies a test objective such as executing a particular transition [7].

Now let us suppose that x_1x_2 is to be input when M is in state s and x_1 and x_2 are input at port p. Suppose further that x_1 is expected to lead to output y at port p and y' at port $q \neq p$ and x_2 is expected to lead to output y at p only. Then x_1yx_2y should be observed at port p and y' should be observed at q. These local traces are still observed if y is produced in response to x_1 and y and y' are produced in response to x_2, in which case there is fault masking. These two scenarios are illustrated by MSC2 and MSC3 in Figure 2. These transitions could lead to failures if used within a *different* sequence.

Since the presence of multiple ports affects the ability of both testers and users to observe

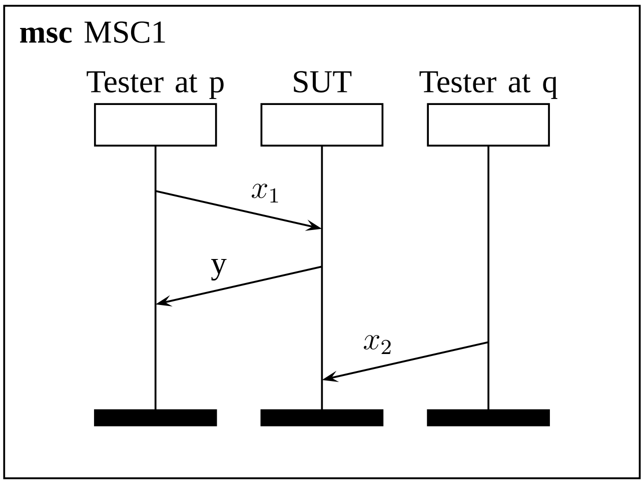

Fig. 1. A controllability problem

system behaviour, we need to define conformance relations for distributed systems: if we test using the wrong conformance relation then we may obtain the wrong verdict (the result of testing is incorrect) or testing may be inefficient. An incorrect verdict may be produced since we might declare a behaviour faulty even when the users cannot distinguish between this and a correct behaviour. Inefficiency might occur through producing tests to find 'faulty' behaviours that are indistinguishable from correct behaviours and so do not actually represent failures. Most previous work has used traditional conformance relations designed for systems that have a single interface and has attempted to produce input sequences that do not have controllability or observability problems. The resultant test generation algorithms lack generality, since these problems cannot always be overcome. Even worse, since the wrong conformance relation is used, the system under test may fail such a test *even though it cannot be distinguished from a correct system in use*.

Recent work has defined what it means for an input sequence to distinguish two states or deterministic FSMs (DFSMs) when restricting testing to input sequences that cause no controllability problems and has defined a corresponding conformance relation [4]. This has been extended to more general conformance relations, that are used in this paper, for both DFSMs and nondeterministic FSMs (NFSMs) [22]. This has also been extended to input output transition systems [20]. These conformance relations reflect the inability of a tester or user to

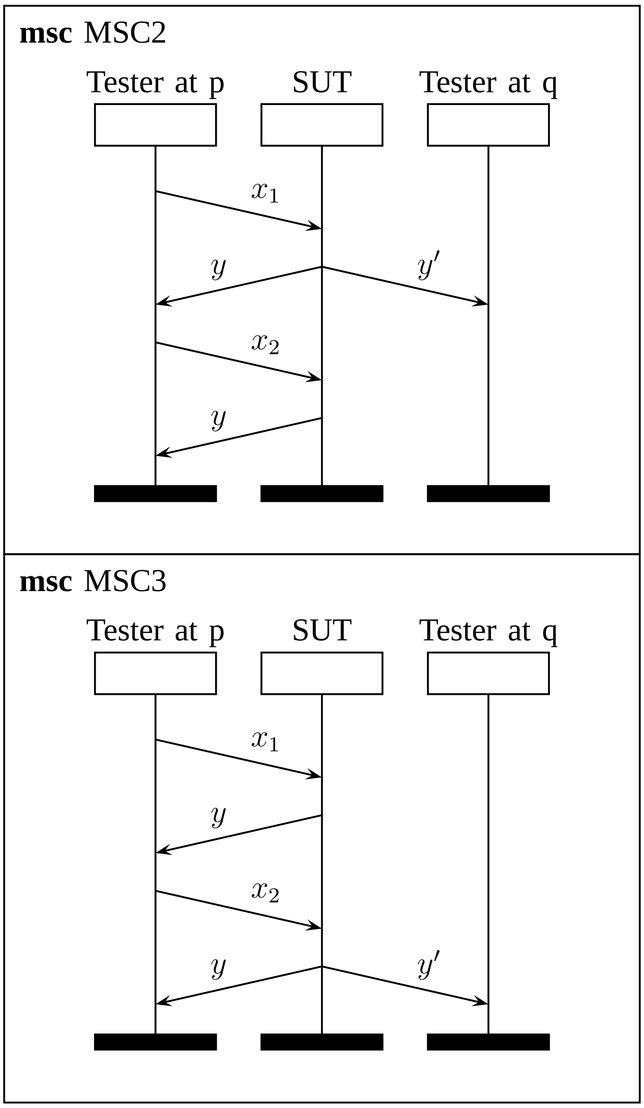

Fig. 2. An observability problem

observe the global trace. Interestingly, the notion of making local observations has been explored in the context of refinement and CSP, although the technical issues are different [23]. However, the oracle problem has not previously been considered for these conformance relations and this is the problem studied here.

Previous work has aimed to determine the global trace that occurred in testing or to check properties of this. Examples include work on run-time verification (see, for example, [24]). In addition, there are approaches in which the testers communicate in order to determine the global trace that occurred (see, for example, [25], [26]). There has also been a significant amount of work on monitoring, in which we wish to determine the global state of the SUT (see, for example, [27], [28], [29], [30], [31]). In contrast to these, we are concerned with black-box testing and we are interested in conformance relations that capture the observational power of potential users. There is another line of work that has defined conformance relations such as $mioco$ for systems with distributed interfaces but this assumes that global traces are observed; it differs from traditional conformance relations such as $ioco$ by allowing the SUT to block all input at a given port (see, for example, [32], [33], [34]).

This paper investigates the oracle problem in the context of testing a black-box SUT with physically distributed ports against a (possibly nondeterministic) FSM. We need different oracles for different conformance relations so it considers the two previously defined conformance relations for testing from an FSM with distributed ports [22]. We give an algorithm for the weaker conformance relation $\sqsubseteq_w$ and prove that this operates in low order polynomial time. We give two algorithms for the other conformance relation $\sqsubseteq_s$: a general algorithm and an algorithm for the special case where we are testing from a DFSM with a controllable input sequence[1]. While it transpires that the algorithm for using controllable input sequences when testing from DFSMs operates in low order polynomial time, the general algorithm has exponential time complexity. We then prove that the general oracle problem for testing from a DFSM with $\sqsubseteq_s$ is NP-hard and this problem is NP-hard for NFSMs even if we restrict attention to controllable input sequences. We then give sufficient conditions, on the input sequence or on the NFSM, under which the oracle problem for NFSMs can be solved in polynomial time. If it is not feasible to solve the oracle problem for $\sqsubseteq_s$ then we can instead use an oracle for $\sqsubseteq_w$ and this provides a sound approximation: it will never declare an SUT that conforms to the specification to be faulty but may miss failures.

The paper is structured as follows. Section II provides preliminary definitions while Section III shows how the oracle problem can be solved for $\sqsubseteq_w$. Section IV then explores properties of $\sqsubseteq_s$ and Section V gives algorithms for solving the oracle problem for $\sqsubseteq_s$. Section VI then gives the complexity results for the oracle problem with $\sqsubseteq_s$ and finally Section VII gives conclusions and describes avenues for future work.

[1]In Section II we formally define what it means for an input sequence to be controllable.

II. Preliminaries

A. Basic definitions

Given sets A and B, $A \leftrightarrow B$ denotes the set of relations between A and B. Given a set A we let A^* denote the set of finite sequences of elements of A and given $a \in A$ we let a^* denote the set $\{a\}^*$. Given a sequence σ, $pre(\sigma)$ is the set of prefixes of σ and given a set Z of sequences we let $pre(Z)$ denote the set of prefixes of sequences from Z. We use ϵ to represent the empty sequence.

In this paper we consider systems that have multiple ports (interfaces). If there are m ports then we represent these with integers and so let the set $\mathcal{P}$ of ports equal $\{1, \ldots, m\}$. Typically we will use x_p to denote input at port p and y_p to denote output at port p, in each case possibly priming names.

B. Finite state machines

A (completely specified) multi-port finite state machine M with m ports is defined by a tuple (S, s_0, X, Y, h) in which:

1) S is a finite set of states;
2) $s_0 \in S$ is the initial state;
3) $X = X_1 \cup \ldots \cup X_m$ is the finite input alphabet in which for all $p \in \mathcal{P}$, X_p is the set of inputs that can be received at p. For all $p, q \in \mathcal{P}$ with $p \neq q$, $X_p \cap X_q = \emptyset$;
4) $Y = (Y_1 \cup \{-\}) \times \ldots \times (Y_m \cup \{-\})$ is the finite output alphabet, where for all $p \in \mathcal{P}$, Y_p denotes the outputs the SUT can send to port p. $(y_1, \ldots, y_m) \in Y$ denotes the value y_p being sent to port p for all $p \in \mathcal{P}$ while $-$ denotes no output being produced; and
5) h is the transition relation of type $S \times X \leftrightarrow S \times Y$.

As a consequence of the definition, an FSM can respond to an input with at most one output at each port. In this paper we only consider completely specified FSMs: if an FSM M is not completely specified then typically it is possible to complete M by either adding an error state or by adding self-loop transitions, that do not change the state, with no output. Since this paper concerns systems with multiple ports, a multi-port finite state machine will be called a *finite state machine (FSM)* and when we wish to refer to an FSM with one port we call it a *single-port FSM*. Note that while we require the X_p and also the Y_p to be disjoint, this can always be achieved by labelling an input or output with the corresponding port number.

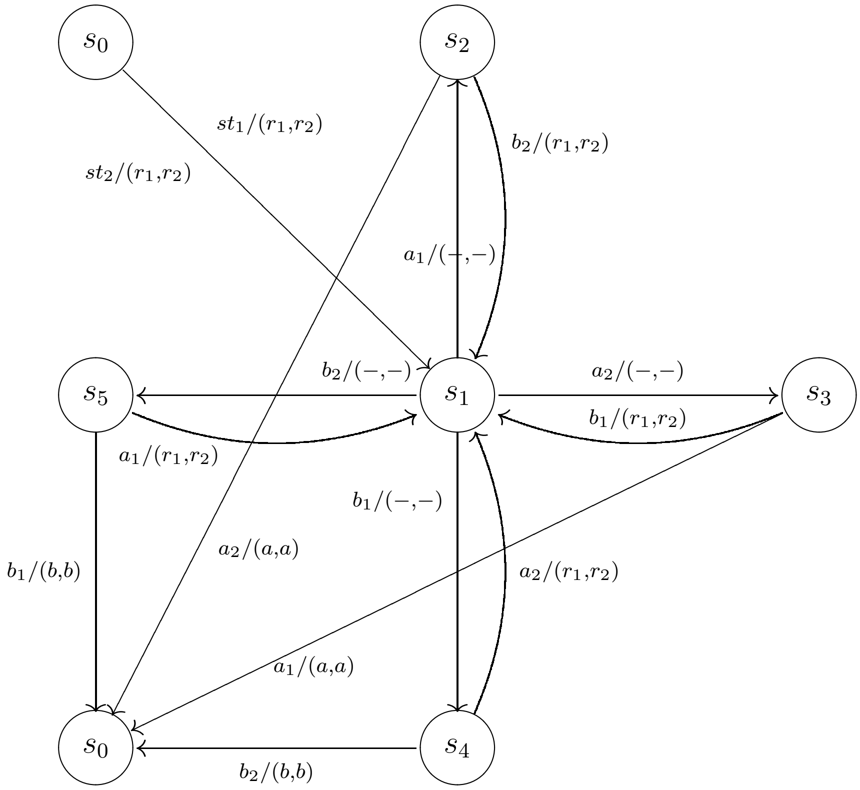

Fig. 3. Finite State Machine M_0

Figure 3 gives an example of an FSM with two ports. This is a simple model of a voting system in which two agents vote either a or b and if they agree then the result is returned to them. Either party can start the process, sending a start message (st_1 at port 1 and st_2 at port 2) and in response the model sends a request r_p to port p ($p \in \{1, 2\}$). Each agent can then vote either a (inputs a_1, a_2 at ports 1 and 2 respectively) or b (inputs b_1, b_2 at ports 1 and 2 respectively). If the two votes are the same then output is sent to each agent confirming the vote and otherwise the system returns to a state from which the agents can vote and requests them to vote. In order to simplify Figure 3 we have not included all of the transitions; where no transition from state s_i with an input x is shown there is an implicit transition from s_i to s_i with input x and output $(-, -)$. In addition, in Figure 3 we have included two copies of state s_0; one defines the transitions leaving s_0 and the other defines the transitions that end in s_0. Figure 3 is based on an input output transition system given in [21].

If $(s', y) \in h(s, x)$ then this means that if M receives input x when in state s then it can move to state s' and produce output y. This defines a transition $t = (s, s', x/y)$. Consider, for example, the FSM M_0 shown in Figure 3. Here $h(s_0, st_1) = \{(s_1, (r_1, r_2))\}$ and so if M_0 receives input st_1 when in state s_0 then it moves to state s_1 and outputs r_1 to port 1 and r_2 to port 2. This defines the transition $(s_0, s_1, st_1/(r_1, r_2))$.

FSM M is a *deterministic FSM (DFSM)* if for all $s \in S$ and $x \in X$, we have that $|h(s, x)| = 1$. Clearly M_0 is deterministic. A sequence of consecutive transitions $\rho = t_1 \ldots t_k$, $t_i = (s_i, s_{i+1}, x_i/y_i)$, is a *path* that has *label* $\sigma = x_1/y_1, \ldots, x_k/y_k$ and *starting state* s_1. The label of ρ is said to be an input/output sequence and also a *global trace*. In addition, the *input portion* of σ is the input sequence $x_1, \ldots, x_k$. For example, path $(s_0, s_1, st_1/(r_1, r_2))(s_1, s_2, a_1/(-, -))$ of M_0 has label $st_1/(r_1, r_2)a_1/(-, -)$, which has input portion st_1a_1, and starting state s_0. The FSM M defines the regular language $L(M)$ of labels of paths with starting state s_0. Similarly, $L_M(s)$ is the set of labels of paths with starting state s. If w is an input sequence then we let $M(w)$ denote the set of global traces in $L(M)$ that have input portion w. For example, $M_0(st_1a_1) = \{st_1/(r_1, r_2)a_1/(-, -)\}$. An FSM N with the same input and output alphabets as M is said to be a *reduction* of M if $L(N) \subseteq L(M)$. FSMs M and N are *equivalent* if $L(N) = L(M)$ and a DFSM M is *minimal* if no DFSM with fewer states is equivalent to M. When testing from a single-port FSM M it is normal to use the conformance relation that requires the implementation FSM to be a reduction of M.

We can define the projection of a global trace. Given $y = (y_1, \ldots, y_m) \in Y$ and $p \in \mathcal{P}$ we let $\pi_p(y)$ denote y_p if $y_p \neq -$ and otherwise $\pi_p(y) = \epsilon$. We can extend this to global traces in the following way.

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

For example $\pi_1(st_1/(r_1, r_2)a_1/(-, -)) = st_1r_1a_1$ and $\pi_2(st_1/(r_1, r_2)a_1/(-, -)) = r_2$.

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 $\pi_p(\sigma_1) = \pi_p(\sigma_2)$. For example, $st_1/(r_1, r_2)a_1/(-, -)a_2/(a, a) \sim st_1/(r_1, r_2)a_2/(-, -)a_1/(a, a)$. Clearly $\sim$ is an equivalence relation.

C. Controllability problems

It is well known that the presence of multiple ports can lead to controllability problems in testing. Essentially, a controllability problem occurs when the tester at a port $p \in \mathcal{P}$ is meant to apply an input x but cannot know when to do this based on the observations that have been made at p. For DFSMs, this has been characterised in terms of global traces being controllable (see, for example, [4]).

Definition 1 *A path $\rho = t_1 \dots t_k$, $t_i = (s_i, s_{i+1}, x_i/y_i)$, is* controllable *if for all $1 < i \leq k$ we have that the port $p \in \mathcal{P}$ such that $x_i \in X_p$ satisfies the condition that $\pi_p(x_{i-1}/y_{i-1}) \neq \epsilon$. We also say that the label of ρ is* controllable.

It is straightforward to see that the path $(s_0, s_1, st_1/(r_1, r_2))(s_1, s_2, a_2/(-, -))(s_2, s_0, a_1/(a, a))$ of M_0 is not controllable since the third input is at port 1 but the second transition does not have either input or output at 1.

Definition 2 *Given a DFSM M an input sequence $w = x_1, \dots, x_k$ is said to be* controllable for *M if the trace $M(w)$ is controllable. When M is clear we simply say that w is* controllable.

Recent work [22] has looked at testing from a possibly nondeterministic FSM M. Here, we need a slightly different definition of what it means for an input sequence to be controllable since an input sequence may be capable of triggering more than one path through M. The corresponding global traces might lead to different possible observations at a port $p \in \mathcal{P}$ and we require that irrespective of which trace occurs, the tester at p must be able to determine when to apply its input.

Consider, for example, an FSM with two ports and input sequence $w = x_1x_1x_2$, in which x_1 is at port 1 and x_2 is at port 2, that can lead to traces $x_1/(y_1, -)x_1/(-, y_2)x_2/(y_1, y_2)$ and $x_1/(-, y_2)x_1/(-, y_2)x_2/(y_1, y_2)$. Here both traces are controllable but after observing y_2 the tester at port 2 does not know whether to wait for another y_2, which is required if the second trace occurs, or apply input x_2, which is required if the first trace occurs. Here, a controllability problem occurs because a tester must make a decision regarding when to send an input but cannot do this on the basis of its own observations. This happens if there are two possible traces σ_1 and σ_2 such that the tester at port p should send input after σ_1, it should not send input after σ_2 (or should send a different input) and yet the tester at p cannot distinguish between σ_1 and σ_2 ($\pi_p(\sigma_1) = \pi_p(\sigma_2)$). This can only happen if σ_1 and σ_2 have different numbers of inputs. The following defines what it means for an input sequence to be controllable for an FSM that might be nondeterministic and is based on a definition in [22].

Definition 3 *Given FSM M an input sequence w is* controllable *for M if there does not exist $\sigma_1, \sigma_2 \in pre(M(w))$ that have different numbers of inputs such that the next input to be applied after σ_1 is to be applied at a port $p \in \mathcal{P}$ such that $\pi_p(\sigma_1) = \pi_p(\sigma_2)$. Where M is clear from the context we say that w is* controllable.

The following gives an alternative characterisation.

Proposition 1 *Given FSM M an input sequence w is controllable for M if there does not exist input $x_p \in X_p$ and $\sigma_1, \sigma_2 \in M(w)$ with prefixes σ_1' and σ_2' respectively such that $\pi_p(\sigma_1') = \pi_p(\sigma_2')$ and the following hold:*

1) *There exists $y \in Y$ such that $\sigma_1' x_p/y \in pre(M(w))$; and*
2) *There does not exist $y \in Y$ such that $\sigma_2' x_p/y \in pre(M(w))$.*

III. Weak conformance and local oracles

In some situations the agents at the separate ports of the SUT will never interact with one another or share information with other agents that can interact with one another. If this is the case then it is sufficient that the local behaviour observed at a port p is a local behaviour of M. This situation is captured by the following conformance relation [22].

Definition 4 *Given FSMs N and M with the same input and output alphabets and the same set of ports, $N \sqsubseteq_w M$ if for every global trace $\sigma \in L(N)$ and port $p \in \mathcal{P}$ there exists some $\sigma' \in L(M)$ such that $\pi_p(\sigma') = \pi_p(\sigma)$. FSM N is then said to* weakly conform *to FSM M.*

In testing on the basis of $\sqsubseteq_w$ it is sufficient to place a local tester at each port and give each local tester its own local oracle. This allows each local tester to return a verdict: pass if the behaviour it observes is consistent with its local oracle and otherwise fail. Since for each transition there is only one port that provides input, FSMs are not the best formalism for describing these local oracles and instead we use finite automata.

A *finite automaton (FA)* F is defined by a tuple (Q, q_0, A, δ, Q_F) in which Q is a finite set of states, $q_0 \in Q$ is the initial state, A is the finite input alphabet, δ is the state transfer relation of type $Q \times (A \cup \{\tau\}) \leftrightarrow Q$, and $Q_F \subseteq Q$ is the set of final states. Here τ is used to represent empty/silent transitions that require no input. If F receives $a \in A$ when in state $q \in Q$ then it moves to a state in $\delta(q, a)$. If $\delta(q, \tau)$ is defined and $q' \in \delta(q, \tau)$ then when F is in state q it is possible for it to move to state q' spontaneously without receiving input. We can use the following notation to represent the possible states of F after receiving an input sequence.

1) $q \xrightarrow{a} q'$ if $q' \in \delta(q, a)$ for $a \in (A \cup \{\tau\})$
2) $q \stackrel{\epsilon}{\Rightarrow} q'$ if there exists states $q_1, \ldots, q_k$, with $q_1 = q$ and $q_k = q'$, such that for all $1 \leq i < k$ we have that $q_i \xrightarrow{\tau} q_{i+1}$. Note that for all states q we have that $q \stackrel{\epsilon}{\Rightarrow} q$.
3) $q \stackrel{a}{\Rightarrow} q'$ for $a \in A$ if there exists states q_1, q_2 such that $q \stackrel{\epsilon}{\Rightarrow} q_1$, $q_1 \xrightarrow{a} q_2$, and $q_2 \stackrel{\epsilon}{\Rightarrow} q'$.
4) Given $\sigma = a_1, \ldots, a_k \in A^*$ we write $q \stackrel{\sigma}{\Rightarrow} q'$ if there exist $q_1, \ldots, q_{k+1}$ with $q_1 = q$ and $q_{k+1} = q'$ such that for all $1 \leq i \leq k$ we have that $q_i \stackrel{a_i}{\Rightarrow} q_{i+1}$.

Essentially, for a sequence $\sigma = a_1, \ldots, a_k \in A^*$, $q \stackrel{\sigma}{\Rightarrow} q'$ holds if and only if it is possible to move from state q to state q' using input sequence σ. FA F defines the language $L(F)$ of sequences that can take F from its initial state to a final state. More formally, $L(F)$ is the set of sequences $\sigma \in A^*$ such that there is a state $q \in Q_F$ such that $q_0 \stackrel{\sigma}{\Rightarrow} q$.

Algorithm 1 takes an FSM M and port p and builds a local oracle M_p. It achieves this by replacing each transition of M, of the form $t = (s, s', x/y)$, by a path from s to s' in M_p with label $\pi_p(x/y)$. There are essentially three cases to consider. If $\pi_p(x/y)$ is the empty sequence then we add a transition from s to s' with label τ. If $\pi_p(x/y)$ contains one element (an input or an output) then we add a transition from s to s' with this element as its label. Finally, if $\pi_p(x/y) = xy_p$ for some $y_p \in Y_p$ then we add an intermediate state s_t, a transition from s to s_t with label x and a transition from s_t to s' with label y_p. We make S the set of final states in order to avoid the language $L(M_p)$ including the label of a path that ends at one of the new intermediate states and thus that includes the input of a transition but not the output.

Consider again FSM M_0 and port 1. Then transition $(s_1, s_2, a_1/(-, -))$ would be represented by a transition (s_1, s_2, a_1). Transition $(s_1, s_5, b_2/(-, -))$ would be represented by transition

(s_1, s_5, τ). For transition $(s_0, s_1, st_1/(r_1, r_2))$ we would have to add an intermediate state s_t and two transitions (s_0, s_t, st_1) and (s_t, s_1, r_1). State s_t is not a final state since otherwise it would suggest that in state s_0 it is possible for the input of st_1 to not produce output at port 1.

Algorithm 1 Building the local oracle M_p

Input FSM $M = (S, s_0, X, Y, h)$ and port p

Let X_p denote the set of inputs at p and Y_p denote the set of outputs at p

Let $S' := S$; $\delta := \emptyset$

for all $((s_i, x), (s_j, y)) \in h$ with $y_p = \pi_p(y)$ **do**

  if $x \in X_p$ and $y_p \neq -$ **then**

    Define a new state s_t and let $S' := S' \cup \{s_t\}$; $\delta := \delta \cup \{((s_i, x), s_t), ((s_t, y_p), s_j)\}$

  else

    if $x \in X_p$ and $y_p = -$ **then**

      $\delta := \delta \cup \{((s_i, x), s_j)\}$

    else

      if $x \notin X_p$ and $y_p \neq -$ **then**

        $\delta := \delta \cup \{((s_i, y_p), s_j)\}$

      else

        if $x \notin X_p \wedge y_p = -$ **then**

          $\delta := \delta \cup \{((s_i, \tau), s_j)\}$

        end if

      end if

    end if

  end if

end for

Output FA $M_p = (S', s_0, X_p \cup Y_p \cup \{\tau\}, \delta, S)$

Proposition 2 *Algorithm 1 is correct in the sense that, when given FSM M and port p it returns FA M_p such that $L(M_p) = \{\sigma_p | \exists \sigma \in L(M). \sigma_p = \pi_p(\sigma)\}$.*

Proof: We will prove that $\sigma_p \in L(M_p)$ if and only if there exists $\sigma \in L(M)$ such that $\sigma_p = \pi_p(\sigma)$.

First assume that $\sigma_p \in L(M_p)$ and thus that there is a path ρ_p from the initial state s_0 of M_p that has label σ_p. We will use proof by induction on the length of the shortest such path. The base case, which is the empty path (and so $\sigma_p = \epsilon$) holds immediately. Let ρ_p denote a shortest path of M_p with label σ_p and assume that the result holds for all shorter paths (the inductive hypothesis). Let $\rho_p = \rho'_p\rho''_p$ such that ρ'_p is the shortest non-empty prefix of ρ_p that ends in a final state (a state from S) and let this state be denoted s. Let σ'_p and σ''_p denote the labels of ρ'_p and ρ''_p respectively. By the definition of M_p, there is a path in M from s_0 to s with label x/y such that $\sigma'_p = \pi_p(x/y)$. Let M_s denote M with s as its initial state. By the inductive hypothesis applied to sequence σ''_p and M_s, there is some $\sigma'' \in L(M_s)$ such that $\pi_p(\sigma'') = \sigma''_p$. Thus, $\sigma_p = \pi_p(x/y)\sigma''_p$, $\sigma''_p = \pi_p(\sigma'')$ for some $\sigma'' \in L(M_s)$ and so $\sigma_p = \pi_p(x/y\sigma'')$ and $x/y\sigma'' \in L(M)$ as required.

Now assume that $\sigma \in L(M)$ and we require to prove that $\sigma_p = \pi_p(\sigma) \in L(M_p)$. We will use proof by induction on the length of σ. The result holds immediately for the base case with length 0. Inductive hypothesis: for every sequence σ with length less than k we have that if $\sigma \in L(M)$ then $\sigma_p = \pi_p(\sigma) \in L(M_p)$. Let $\sigma = x_1/y_1, \ldots, x_k/y_k$ and let s denote a state reached by the first transition in a path ρ that has starting state s_0 and label σ. By construction, there is a path in M_p from s_0 to s with label $\pi_p(x_1/y_1)$. The result thus follows by applying the inductive hypothesis to $x_2/y_2, \ldots, x_2/y_2$ and M_s. ■

The following result says that if the local tester at port p observes a local trace that is not in $L(M_p)$ then we know that the SUT has produced a global trace that is not allowed.

Proposition 3 *If Algorithm 1 returns FA M_p when given FSM M and port $p \in \mathcal{P}$ and the SUT N has a global trace σ such that $\pi_p(\sigma) \notin L(M_p)$ then we do not have that $N \sqsubseteq_w M$. In addition, if for all $\sigma \in L(N)$ and $p \in \mathcal{P}$ we have that $\pi_p(\sigma) \in L(M_p)$ then $N \sqsubseteq_w M$.*

Proof: First assume that Algorithm 1 returns FA M_p when given FSM M and port $p \in \mathcal{P}$ and the SUT N has a global trace σ such that $\pi_p(\sigma) \notin L(M_p)$. By Proposition 2, this means that there does not exist $\sigma' \in L(M)$ such that $\pi_p(\sigma') = \pi_p(\sigma)$. By Definition 4, this means that we do not have that $N \sqsubseteq_w M$ as required.

Now assume that for all $\sigma \in L(N)$ and $p \in \mathcal{P}$ we have that $\pi_p(\sigma) \in L(M_p)$. By Proposition 2, this means that for all $\sigma \in L(N)$ and $p \in \mathcal{P}$ there exists $\sigma' \in L(M)$ such that $\pi_p(\sigma') = \pi_p(\sigma)$.

By Definition 4, this means that we have that $N \sqsubseteq_w M$ as required. ■

Thus, in order to solve the oracle problem for an FSM M and a set of local traces $\sigma_1, \ldots, \sigma_m$, when using $\sqsubseteq_w$ it is sufficient to solve the oracle problem for each M_p and σ_p. Thus, the oracle problem for $\sqsubseteq_w$ reduces to solving m instances of the membership problem for finite automata and so can be solved in low order polynomial time.

IV. A STRONGER FORM OF CONFORMANCE

We have seen that the M_p returned by Algorithm 1 can be used as oracles when testing with $\sqsubseteq_w$. However, in some situations the traces observed at the different ports can be brought together afterwards, possibly through the agents placed at these ports interacting with other agents. Consider, for example, the FSM M'_0 shown in Figure 4. This, for example, contains the trace $st_1/(r_1, r_2)a_1/(-,-)b_2/(a, b)$. This clearly is not equivalent to any trace of M_0 under $\sim$ and should correspond to an incorrect behaviour: each user believes that other party has agreed to their vote. However, if we consider the projections of this trace we find that $\pi_1(st_1/(r_1, r_2)a_1/(-,-)b_2/(a, b)) = st_1r_1a_1a = \pi_1(st_1/(r_1, r_2)a_1/(-,-)a_2/(a, a))$ and $\pi_2(st_1/(r_1, r_2)a_1/(-,-)b_2/(a, b)) = st_2r_2b_2b = \pi_1(st_1/(r_1, r_2)b_1/(-,-)b_2/(b, b))$. Thus, neither tester observes a failure.

In order to overcome this issue we get the following notion of conformance in which we require every global trace of the implementation to be indistinguishable from a global trace of the specification [22].

Definition 5 *Given FSMs N and M with the same input and output alphabets and the same set of ports, $N \sqsubseteq_s M$ if for all $\sigma \in L(N)$ there exists some $\sigma' \in L(M)$ such that $\sigma' \sim \sigma$.*

We can test for $\sqsubseteq_s$ by placing local testers at each port and bringing together the observed local traces after testing. While the testers cannot synchronise during testing they can send their observations to a single agent after testing.

The conformance relation $\sqsubseteq_s$ places stronger constraints on the SUT than $\sqsubseteq_w$. Proposition 5 below says that it is possible for the verdicts returned based on the local oracles to be pass and yet the set of local traces to not be consistent with any behaviour of M and thus proves that $\sqsubseteq_w$ is weaker than $\sqsubseteq_s$.

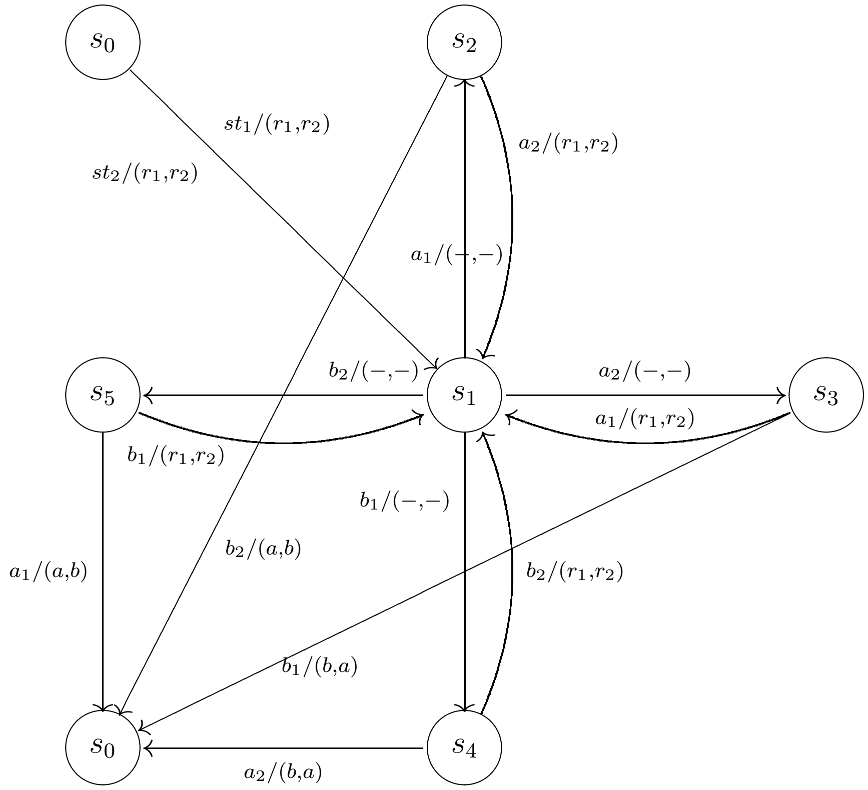

Fig. 4. Finite State Machine M'_0

Proposition 4 *Given an FSM M with m ports and a trace σ, let us suppose that for every port $p \in \mathcal{P}$ we have that $\pi_p(\sigma) \in L(M_p)$ for the FA M_p returned by Algorithm 1 when given M and p. It is possible that there is no global trace $\sigma' \in L(M)$ such that $\sigma' \sim \sigma$.*

Proof: It is sufficient to consider M_0 and the trace $st_1/(r_1, r_2)a_1/(-,-)b_2/(a, b)$ of M'_0. ■

Proposition 5 *Given FSMs N and M with the same input and output alphabets and the same set of ports, if $N \sqsubseteq_s M$ then $N \sqsubseteq_w M$. The converse is not the case in the sense that it is possible that $N \sqsubseteq_w M$ but we do not have that $N \sqsubseteq_s M$.*

Proof: For the first part, assume that $N \sqsubseteq_s M$, $\sigma \in L(N)$, and $p \in \mathcal{P}$. It is sufficient to prove that there exists $\sigma' \in L(M)$ such that $\pi_p(\sigma) = \pi_p(\sigma')$. But, since $N \sqsubseteq_s M$, there exists $\sigma' \in L(M)$ such that $\sigma \sim \sigma'$ and so the result follows.

For the second part, consider an FSM M with one state and two ports in which the response to input x at port 1 is either y_1 at 1 and y_2 at 2 or y_1' at 1 and y_2' at 2. Further, assume that x is the only input. Now let N denote an FSM with one state and two ports in which the response to input x at port 1 is y_1 at 1 and y_2' at 2. We do not have that $N \sqsubseteq_s M$ since the non-empty traces of N are not equivalent to traces of M under $\sim$. Further, for every trace σ of N and port p we have that $\pi_p(\sigma)$ is a projection of a trace of M: the trace with the same number of inputs that always takes the transition that has the same output at p as the transition in N. Thus, we have that $N \sqsubseteq_w M$ as required. ■

Thus, we know that $\sqsubseteq_w$ and $\sqsubseteq_s$ differ in general. It is natural to ask how they relate to one another and to the reduction relation if we have only one port. As we would expect, if there is only one port then these three conformance relations are equivalent.

Proposition 6 *Given single-port FSMs N and M with the same input and output alphabets we have that $N \sqsubseteq_s M$ if and only if N is a reduction of M. In addition, $N \sqsubseteq_w M$ if and only if N is a reduction of M.*

Proof: The first part follows from observing that when there is only one port we have that equivalence under $\sim$ is just equality and so $N \sqsubseteq_s M$ if and only if every global trace of N is a trace of M.

For the second part observe that when there is only one port, for every trace σ we have that $\pi_1(\sigma) = \sigma$. Thus, $N \sqsubseteq_w M$ if and only if for every trace σ of N we have a trace σ' of M such that $\pi_1(\sigma) = \pi_1(\sigma')$ and this holds if and only if N is a reduction of M. ■

It is therefore interesting to consider how $\sqsubseteq_w$ and $\sqsubseteq_s$ relate to reduction for FSMs with more than one port.

Proposition 7 *Given FSMs N and M with the same input and output alphabets and the same sets of ports, if N is a reduction of M then $N \sqsubseteq_s M$, but the converse is not true.*

Proof: First assume that N is a reduction of M and that $\sigma \in L(N)$. It is sufficient to prove that there is some $\sigma' \sim \sigma$ such that $\sigma' \in L(M)$. However, since N is a reduction of M we must have that $\sigma \in L(M)$ and so we can simply choose $\sigma' = \sigma$.

For the second part, consider the DFSMs M and N shown in Figure 5 that have two ports 1 and 2. Here $L(N) = ((x_1/(y_1, -) + x_2/(-, y_2'))^*$ and it is clear that all sequences in $L(N)$

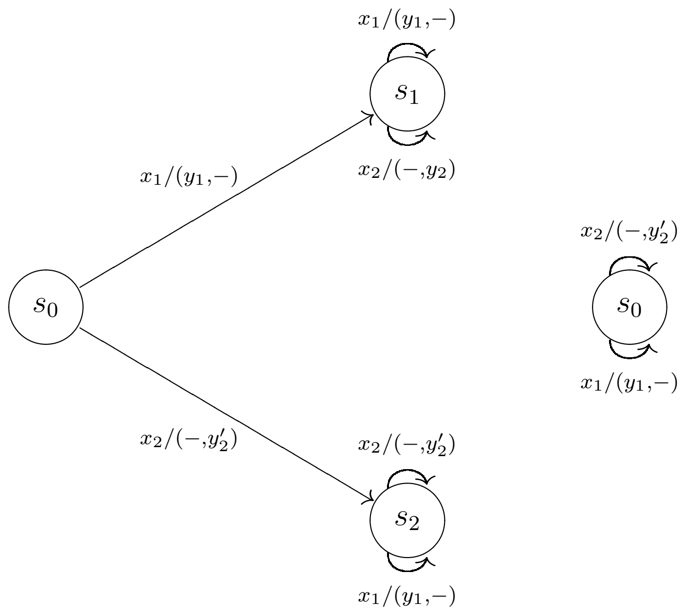

Fig. 5. DFSMs M and N

that start with $x_2/(-, y_2')$ are also in $L(M)$. It is also clear that all sequences in $L(N)$ that do not contain input x_2 are also in $L(M)$ since these are all words in the language $(x_1/(y_1, -))^*$. Finally, if a sequence from $L(N)$ is of the form $\sigma = (x_1/(y_1, -))^n(x_2/(-, y_2'))\sigma_0$ for some n and $\sigma_0 \in ((x_1/(y_1, -)) + (x_2/(-, y_2'))^*$ then $\sigma \sim (x_2/(-, y_2'))(x_1/(y_1, -))^n\sigma_0 \in L(M)$. Thus, $N \sqsubseteq_s M$ and yet it is clear that M is minimal and N is not a reduction of M. ■

Proposition 8 *Given FSMs N and M with the same input and output alphabets and the same sets of ports, if N is a reduction of M then $N \sqsubseteq_w M$, but the converse is not true.*

Proof: First assume that N is a reduction of M, $\sigma \in L(N)$, and $p \in \mathcal{P}$. It is sufficient to prove that there is some $\sigma' \in L(M)$ such that $\pi_p(\sigma') = \pi_p(\sigma)$. However, since N is a reduction of M we must have that $\sigma \in L(M)$ and so we can simply choose $\sigma' = \sigma$.

For the second part, again consider the DFSMs M and N shown in Figure 5. Since we have that $N \sqsubseteq_s M$, from Proposition 7 we know that $N \sqsubseteq_w M$. However, as established in the proof of Proposition 7, N is not a reduction of M and so the result follows. ■

We now know that $\sqsubseteq_s$ is weaker than the conformance relation usually used when testing from an FSM. Since the reduction relation is an equivalence relation when we consider (completely specified) DFSMs it is natural to ask whether $\sqsubseteq_s$ is an equivalence relation on such DFSMs.

Proposition 9 *The relation $\sqsubseteq_s$ is not an equivalence relation on (completely specified) DFSMs.*

Proof: Consider the two DFSMs M_1 and M_2 that are shown in Figure 6; M_1 is at the top and M_2 is at the bottom. In these FSMs there are three ports, x_p denotes input at port $p \in \mathcal{P}$ and y_p (or y'_p, y''_p) denotes output at port p, $p \in \mathcal{P}$. The differences in behaviour are only in response to x_3 and there are only differences after both x_1 and x_2 have been received.

The traces of M_2 that are not in $L(M_1)$ are those that start with an input sequence of the form $w_1x_2w_2x_1w_3x_3$ for some input sequences $w_1 \in x_3^*$, $w_2 \in \{x_2, x_3\}^*$, and $w_3 \in \{x_1, x_2\}^*$. However, for each trace $\sigma \in L(M_2)$ that has input portion $w_1x_2w_2x_1w_3x_3$ for some such w_1, w_2, w_3 there is a trace $\sigma' \in L(M_1)$ with input portion $w_1x_1w'_2x_2w_3x_3$ such that $\sigma' \sim \sigma$. Thus, $M_2 \sqsubseteq_s M_1$. Since $M_2 \sqsubseteq_s M_1$ if $\sqsubseteq_s$ was an equivalence relation, and so symmetric, we would have that $M_1 \sqsubseteq_s M_2$. However, M_1 has the global trace $\sigma = x_2/y_2x_1/y_1x_3/y''_3$ and there is no $\sigma' \in L(M_2)$ such that $\sigma' \sim \sigma$. Thus, $M_1 \not\sqsubseteq_s M_2$ and so $\sqsubseteq_s$ is not an equivalence relation as required. ■

Proposition 10 *The relation $\sqsubseteq_s$ is a pre-order.*

Proof: It is clear that $\sqsubseteq_s$ is reflexive and thus it suffices to prove that $\sqsubseteq_s$ is transitive: if $N_1 \sqsubseteq_s N_2$ and $N_2 \sqsubseteq_s N_3$ then $N_1 \sqsubseteq_s N_3$. We therefore assume that $N_1 \sqsubseteq_s N_2$ and $N_2 \sqsubseteq_s N_3$.

Since $N_1 \sqsubseteq_s N_2$, for all $\sigma \in L(N_1)$ there exists $\sigma' \in L(N_2)$ such that $\sigma' \sim \sigma$. Further, since $N_2 \sqsubseteq_s N_3$, for all $\sigma' \in L(N_2)$ there exists $\sigma'' \in L(N_3)$ such that $\sigma'' \sim \sigma'$. Thus, for all $\sigma \in L(N_1)$ there exists $\sigma'' \in L(N_3)$ such that $\sigma'' \sim \sigma$ and so $N_1 \sqsubseteq_s N_3$ as required. ■

V. THE ORACLE PROBLEM FOR $\sqsubseteq_s$

In testing we need to determine whether an observed behaviour is consistent with the specification. This is trivial for testing from a single-port DFSM since here the input sequence w defines a single input/output sequence and it is not much more difficult for an NFSM. We have seen that it is also straightforward when testing with the conformance relation $\sqsubseteq_w$: we simply construct the M_p and use these. In this section we explore the oracle problem for $\sqsubseteq_s$.

Algorithm 2 takes an FSM M and observed local traces $\sigma_1, \ldots, \sigma_m$ and decides whether there is some $\sigma' \in L(M)$ such that $\pi_p(\sigma') = \sigma_p$ for all $p \in \mathcal{P}$. This algorithm operates in the following

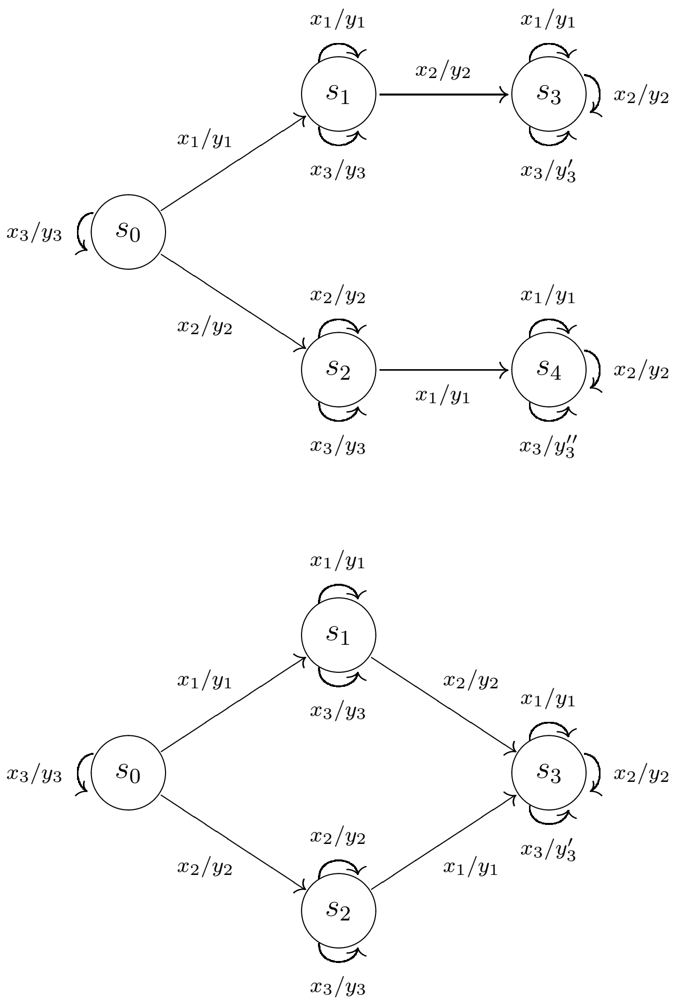

Fig. 6. DFSMs M_1 and M_2

way. At each step it considers a current tuple containing a state s and local traces $\sigma_1', \ldots, \sigma_m'$ and determines whether M has any transitions that are consistent with this. Here, a transition $t = (s, s', x/y)$ is consistent with this if we have that for every port p, $\pi_p(x/y)$ is a prefix of σ_p'. If transition t is consistent with such a current tuple then we create a new tuple in which the state is s' and the local trace for a port p is defined by removing $\pi_p(x/y)$ from the front of σ_p'. The algorithm processes one input in each iteration and in iteration i it forms a set Z_i of tuples.

Each iteration leads to a set of tuples of the form $(s, \sigma_1', \ldots, \sigma_m')$ such that σ_p' is a suffix of σ_p ($p \in \mathcal{P}$) and so $\sigma_p = \sigma_p''\sigma_p'$ for some σ_p''. This tuple has the property that it is possible for

M to move to state s with a global trace σ such that for all $p \in \mathcal{P}$ we have that $\pi_p(\sigma) = \sigma''_p$. Given a set Z_i of such tuples formed in the operation of Algorithm 2, the algorithm will return True if there is some $(s, \sigma'_1, \ldots, \sigma'_m)$ in Z_i such that in M there is a trace σ from state s with $\pi_p(\sigma) = \sigma'_p$ for all $p \in \mathcal{P}$. In each iteration we therefore consider the set of such tuples and for each such $(s, \sigma'_1, \ldots, \sigma'_m)$ we find the set of transitions from s whose input/output x/y has the property that for all $p \in \mathcal{P}$ we have that $\pi_p(x/y)$ is a prefix of σ'_p. We then generate a new set of tuples. Since there are k inputs there are k iterations. The global trace σ is consistent with M if and only if we end with a tuple that is of the form $(s, \epsilon, \ldots, \epsilon)$.

Let us suppose that we wish to apply Algorithm 2 with M_0 and the local traces $\sigma_1 = st_1r_1a_1a$ and $\sigma_2 = r_2a_2a$. Initially we have $Z_0 = \{(s_0, st_1r_1a_1a, r_2a_2a)\}$. The only transition consistent with this one tuple is $(s_0, s_1, st_1/(r_1, r_2))$. The new tuple is formed by changing the state to s_1, removing $\pi_1(st_1/(r_1, r_2)) = st_1r_1$ from the front of σ_1 and removing $\pi_2(st_1/(r_1, r_2)) = r_2$ from the front of σ_2. Thus, after the first iteration we have $Z_1 = \{(s_1, a_1a, a_2a)\}$. The one tuple in this set is consistent with two transitions: $(s_1, s_2, a_1/(-, -))$ and $(s_1, s_3, a_2/(-, -))$ and so we get $Z_2 = \{(s_2, a, a_2a), (s_3, a_1a, a)\}$. The first tuple is consistent with $(s_2, s_0, a_2/(a, a))$ and the second tuple is consistent with $(s_3, s_0, a_1/(a, a))$. In each case we obtain the tuple $(s_0, \epsilon, \epsilon)$ and so $Z_3 = \{(s_0, \epsilon, \epsilon)\}$. Thus, the verdict is pass.

Proposition 11 *Given FSM M and local traces $\sigma_1, \ldots, \sigma_m$, Algorithm 2 returns True if and only if there exists some $\sigma \in L(M)$ such that $\pi_p(\sigma) = \sigma_p$ for all $p \in \mathcal{P}$.*

Proof: Consider iteration i of Algorithm 2 and the set Z_i formed in this iteration. By construction each element $(s, \sigma'_1, \ldots, \sigma'_m) \in Z_i$ has the following properties:

1) There exist $\sigma''_1, \ldots, \sigma''_m$ such that $\sigma_p = \sigma''_p\sigma'_p$ for all $p \in \mathcal{P}$ and there is a path in M from s_0 to s with a label σ such that $\pi_p(\sigma) = \sigma''_p$ for all $p \in \mathcal{P}$
2) The set of local traces $\sigma'_1, \ldots, \sigma'_m$ contain exactly i fewer inputs than $\sigma_1, \ldots, \sigma_m$.

It is also clear by construction that Z_i contains all such tuples. From the second property we know that, since $\sigma_1, \ldots, \sigma_m$ contain a finite number of inputs, the algorithm must terminate. Finally, if $\sigma_1, \ldots, \sigma_m$ contain k inputs then there exists $\sigma \in L(M)$ such that $\pi_p(\sigma) = \sigma_p$ for all $p \in \mathcal{P}$ if and only if Z_k contains $(s, \epsilon, \ldots, \epsilon)$ and so the result follows. ■

Thus, the test oracle problem for $\sqsubseteq_s$ is decidable. We now consider the worst case complexity of Algorithm 2.

Algorithm 2 A test oracle for $\sqsubseteq_s$

Input FSM $M = (S, s_0, X, Y, h)$ and local traces $\sigma_1, \ldots, \sigma_m$ that contains k inputs.

Let $Z_0 := \{(s_0, \sigma_1, \sigma_2, \ldots, \sigma_m)\}$

for all $i := 1$ to k **do**

  Let $Z_i := \emptyset$

  for all $(s, \sigma'_1, \sigma'_2, \ldots, \sigma'_m) \in Z_{i-1}$ **do**

    for all $p \in \mathcal{P}$ such that σ'_p starts with an input $x \in X_p$ and $(s', y) \in h(s, x)$ **do**

      if For all $q \in \mathcal{P}$, $\pi_q(x/y) \in pre(\sigma'_q)$ **then**

        For all $q \in \mathcal{P}$ let σ''_q be defined by $\sigma'_q = \pi_q(x/y)\sigma''_q$

        Let $Z_i := Z_i \cup \{(s', \sigma''_1, \sigma''_2, \ldots, \sigma''_m)\}$

      end if

    end for

  end for

end for

if There exists $(s, \epsilon, \ldots, \epsilon) \in Z_k$ **then**

  Output True

else

  Output False

end if

Proposition 12 *Let us suppose that an FSM M has $m > 1$ ports and for each state s and input x there are at most q transitions from s with input x. Then Algorithm 2 operates in time of $O((max\{m, k\}q)^{k+1}m)$ when given M and local traces $\sigma_1, \ldots, \sigma_m$ that contain a total of k inputs.*

Proof: On each iteration of the outer loop, for each element of Z_{i-1} we have to consider at most $max\{m, k\}$ ports since here we are considering any σ_p that starts with an input; there are only m ports and k inputs in total. Each such input defines at most q transitions. For each such transition we take $O(m)$ time since we simply remove at most two elements from the front of the m sequences (the σ_p). Given a tuple in Z_{i-1} with state s and an input x at the front of some σ_p, at worst we include in Z_i one tuple for each transition leaving s with input x and there are at most q such transitions. Since there are at most $max\{m, k\}$ inputs at the front of the σ_p in a tuple in Z_{i-1}, each tuple in Z_{i-1} results in at most $max\{m, k\}q$ elements in Z_i. As a result, since Z_0 has size 1 the size of Z_{i-1} is bounded above by $(max\{m, k\}q)^{i-1}$. Thus, in iteration i we consider at most $(max\{m, k\}q)^{i-1}$ elements of Z_{i-1} and, as seen above, for each of these we consider at most $max\{m, k\}q$ transitions and each transition takes $O(m)$ time. The overall worst time complexity is thus of $O(max\{m, k\}qm + (max\{m, k\}q)(max\{m, k\}qm) + \ldots + (max\{m, k\}q)^{k-1}(max\{m, k\}qm))$. This can be simplified to $O(\sum_{i=1}^{k} m(max\{m, k\}q)^i)$. It is now sufficient to observe that $\sum_{i=1}^{k}(max\{m, k\}q)^i \leq (max\{m, k\}q)^{k+1}$. ■

We now consider the case in which we are testing against a DFSM using a controllable input sequence $w = x_1 \ldots x_k$. Let us suppose that $L(M)$ contains the global trace $x_1/y_1 \ldots x_k/y_k$. Since w is controllable we have that for all $1 \leq i < k$, if x_{i+1} is at port p then $\pi_p(x_i/y_i) \neq \epsilon$.

Algorithm 3 takes a DFSM M and $\sigma_1, \ldots, \sigma_m$ produced by applying a controllable input sequence $x_1, \ldots, x_k$ and decides whether there is some $\sigma' \in L(M)$ such that $\pi_p(\sigma') = \sigma_p$ for all $p \in \mathcal{P}$.

Before proving the correctness of Algorithm 3 we prove a property of controllable traces.

Proposition 13 *Let us suppose that σ is a controllable global trace in $L_M(s)$ for DFSM M. Then there is no global trace $\sigma' \in L_M(s)$ such that $\sigma' \sim \sigma$ and $\sigma' \neq \sigma$.*

Proof: Proof by induction on the number of inputs in σ. The result clearly holds for sequences with no inputs (and so of length 0) and this forms the base case. Inductive hypothesis: the result holds for every FSM M, state s, and controllable global trace $\sigma \in L_M(s)$ with fewer than k inputs ($k > 0$) and consider state s and controllable $\sigma \in L_M(s)$ with k inputs. We will assume that $\sigma' \sim \sigma$ for some $\sigma' \in L_M(s)$ and are required to prove that $\sigma' = \sigma$.

Let $\sigma = x_1/y_1, \ldots, x_k/y_k$ and $\sigma' = x'_1/y'_1, \ldots, x'_k/y'_k$. Since σ is controllable there can only be one port p such that $\pi_p(\sigma)$ starts with an input. Thus, since $\sigma' \sim \sigma$ we must have that $x'_1 = x_1$. Further, since M is deterministic we know that $y'_1 = y_1$. The result now follows by noting that $x_2/y_2, \ldots, x_k/y_k$ is controllable and by applying the inductive hypothesis to $x_2/y_2, \ldots, x_k/y_k$ and $x'_2/y'_2, \ldots, x'_k/y'_k$. ■

Proposition 14 *If Algorithm 3 is given DFSM M, local traces $\sigma_1, \ldots, \sigma_m$, and a controllable input sequence $x_1, \ldots, x_k$ then it returns True if and only if there is a global trace $\sigma \in L(M)$ with input portion $x_1, \ldots, x_k$ that has the property that $\pi_p(\sigma) = \sigma_p$ for all $p \in \mathcal{P}$.*

Algorithm 3 An oracle for $\sqsubseteq_s$ with controllable input sequences

Input DFSM $M = (S, s_0, X, Y, h)$, local traces $\sigma_1, \ldots, \sigma_m$, and controllable input sequence $x_1, \ldots, x_k$

Let $s := s_0$

for all $p \in \mathcal{P}$ **do**

  Let $\sigma_p^0 := \sigma_p$

end for

for all $i := 1$ to k **do**

  Let s' and y_i be defined by $\{(s', y_i)\} = h(s, x_i)$

  for all $p \in \mathcal{P}$ **do**

    if $\pi_p(x_i/y_i) \in pre(\sigma_p^{i-1}$**)** **then**

      Let σ_p^i be defined by $\sigma_p^{i-1} = \pi_p(x_i/y_i)\sigma_p^i$

    else

      Output False and Terminate

    end if

  end for

end for

if For all $p \in \mathcal{P}$ we have that $\sigma_p^k = \epsilon$ **then**

  Output True

else

  Output False

end if

Proof: We use proof by induction on k. The result clearly hold for the base case, which is the empty sequence. Now assume that for every DFSM M and controllable input sequence $x_1, \ldots, x_j$ of length less than k, we have that Algorithm 3 returns True if and only if there is a global trace $\sigma \in L(M)$ with input portion $x_1, \ldots, x_j$ that has the property that $\pi_p(\sigma) = \sigma_p$ for all $p \in \mathcal{P}$. Let $x_1, \ldots, x_k$ be a controllable input sequence.

Since M is deterministic, the result of applying x_1 is uniquely defined and let us suppose that $h(s_0, x_1) = \{(s, y)\}$. Further, $x_2, \ldots, x_k$ is controllable when applied from state s. Algorithm 3 returns True if and only if for all $p \in \mathcal{P}$ we have that $\sigma_p = \pi_p(x_1/y)\sigma'_p$ for some σ'_p such that Algorithm 3 returns True when given DFSM M with initial state s, local traces $\sigma'_1, \ldots, \sigma'_m$, and controllable input sequence $x_2, \ldots, x_k$. The result now follows from the inductive hypothesis. ■

Proposition 15 *Given a DFSM M with n transitions and m ports, a set of local traces and a controllable input sequence of length k, Algorithm 3 operates in time of $O(mk + k\log(n))$.*

Proof: The innermost nested loop iterates a total of mk times since the outermost loop iterates k times (once for each input) and for each such iteration the innermost loop has one iteration for each port. Each iteration takes constant time and so this contributes $O(mk)$. We have to apply the function h once for each input and so a total of k times. If this is achieved by searching through a table that represents h where the transition are listed in lexical order then this can be achieved using a binary search in $O(\log(n))$. Thus, this contributes $O(k\log(n))$ and so the overall worst case time complexity of $O(mk + k\log(n))$. ■

VI. The complexity of the oracle problem

We have seen that we can solve the oracle problem for controllable input sequences with DFSMs in low order polynomial time. However, the time complexity given for Algorithm 2 is exponential. It is thus natural to ask whether there might exist polynomial time algorithms for the general oracle problem. We now explore two cases: NFSMs and DFSMs when we are not using controllable input sequences. We prove that both of these oracle problems are NP-hard by showing that we can reduce the following problem to them.

Definition 6 *Given boolean variables $z_1, \ldots, z_r$ let $C_1, \ldots, C_k$ denote sets of three literals, where each literal is either a variable z_i or its negation. The three-in-one SAT problem is: Does there exist an assignment to the boolean variables such that each C_i contains exactly one true literal.*

The three-in-one SAT problem is motivated by a proposition being written in conjunctive normal form $C_1 \wedge \ldots \wedge C_k$, each conjunct C_i being the disjunction of three literals, and each literal being either a variable or its negation. Thus, $C_i = l_{i1} \vee l_{i2} \vee l_{i3}$ for three literals l_{i1}, l_{i2}, l_{i3}. This problem is known to be NP-hard [35]. We first consider the oracle problem for NFSMs.

Proposition 16 *Given local traces $\sigma_1, \ldots, \sigma_m$ at m ports and an FSM M with m ports, the problem of deciding whether there exists $\sigma' \in L(M)$ such that for all $p \in \mathcal{P}$ we have that $\pi_p(\sigma') = \sigma_p$ is NP-hard.*

Proof: We will show that we can reduce the three-in-one SAT problem to this problem. We therefore suppose that we have variables $z_1, \ldots, z_r$ and clauses $C_1, \ldots, C_k$. We will define an FSM M with $r + k$ ports, inputs $z_1, \ldots, z_r$ at ports $1, \ldots, r$ and outputs $y_1, \ldots, y_{r+k}$ at ports $1, \ldots, r + k$.

FSM M has one state s_0. For an input z_i there are two transitions:

1) A transition that, for all $1 \leq j \leq k$, sends output y_{r+j} to port $r + j$ if and only if C_j contains literal z_i and otherwise sends no output to port $r + j$. For all $1 \leq p \leq r$ it also sends output y_p to port p.
2) A transition that, for all $1 \leq j \leq k$, sends output y_{r+j} to port $r + j$ if and only if C_j contains literal $\neg z_i$ and otherwise sends no output to port $r + j$. For all $1 \leq p \leq r$ it also sends output y_p to port p.

Now consider the local traces $\sigma_1, \ldots, \sigma_{r+k}$ defined by: $\sigma_1 = z_1(y_1)^r, \sigma_2 = y_2 z_2 (y_2)^{r-1}, \ldots, \sigma_r = (y_r)^{r-1} z_r y_r$ and for all $1 \leq i \leq k$ we have that $\sigma_{r+i} = y_{r+i}$. Essentially, each input z_i is received once by the FSM and a nondeterministic choice is made: either an output is sent to all ports that correspond to clauses that contain literal z_i or output is sent to all ports that correspond to clauses that contain literal $\neg z_i$. It is thus clear that there exists $\sigma' \in L(M)$ such that for all $1 \leq p \leq r + k$ we have that $\pi_p(\sigma') = \sigma_p$ if and only if there exist an assignment to the boolean variables $z_1, \ldots, z_r$ such that each C_i contains exactly one true literal. The result thus follows from the three-in-one SAT problem being NP-hard and the fact that it is possible to construct M and the σ_p in polynomial time. ■

Note that the proof constructed an instance of the oracle problem for an NFSM and set of local traces that could correspond to the application of a controllable input sequence and thus the problem is NP-hard even if we restrict testing to using controllable input sequences.

The above proof uses nondeterminism in the FSM to allow an input representing a variable to lead to either a transition that corresponds to that variable being true or a transition that corresponds to the variable being false. We cannot do this in a DFSM and so we require some other mechanism. However, we can reduce the three-in-one SAT problem to the oracle problem for DFSMs.

Proposition 17 *Given local traces $\sigma_1, \ldots, \sigma_m$ at m ports and a DFSM M with m ports, the problem of deciding whether there exists $\sigma' \in L(M)$ such that for all $p \in \mathcal{P}$ we have that $\pi_p(\sigma') = \sigma_p$ is NP-hard.*

Proof: Again we will show that we can reduce the three-in-one SAT problem to this and suppose that we have variables $z_1, \ldots, z_r$ and clauses $C_1, \ldots, C_k$. We will define a DFSM M with $r + k + 1$ ports, inputs $z_0, z_1, \ldots, z_r$ at ports $0, 1, \ldots, r$ and outputs $y_1, \ldots, y_{r+k}$ at ports $1, \ldots, r + k$. Here we count ports from 0 rather than 1 since the role of input at 0 will be rather different from the role of the other inputs.

DFSM M has two states s_0, s_1. For an input z_i with $1 \leq i \leq r$ there are two transitions:

1) From state s_0 there is a transition that, for all $1 \leq j \leq k$, sends output y_{r+j} to port $r + j$ if and only if C_j contains literal z_i and otherwise sends no output to port $r + j$. The transition sends no output to ports $0, \ldots, r$ and does not change state.
2) From state s_1 there is a transition that, for all $1 \leq j \leq k$, sends output y_{r+j} to port $r + j$ if and only if C_j contains literal $\neg z_i$ and otherwise sends no output to port $r + j$. The transition sends no output to ports $0, \ldots, r$ and does not change state.

If M receives input z_0 in state s_0 then it moves to state s_1, producing no output. If M receives z_0 when in state s_1 there is no change in state and no output is produced. In effect, the input of the first z_0 moves us from a state in which the output in response to z_i, $1 \leq i \leq r$, corresponds to z_i being true to a state in which the response to z_i corresponds to z_i being false.

Now consider the local traces $\sigma_0, \sigma_1, \ldots, \sigma_{r+k}$ defined by: $\sigma_0 = z_0, \sigma_1 = z_1, \sigma_1 = z_2, \ldots, \sigma_r = z_r$ and for all $1 \leq i \leq k$ we have that $\sigma_{r+i} = y_{r+i}$. Each input z_i is received once by the DFSM and these could have been received in any order and so for all $1 \leq i \leq r$ we do not know whether z_i has been received before z_0 or after z_0. If z_i is received before z_0 then an output is sent to all ports that correspond to clauses that contain literal z_i. If z_i is received after z_0 then an output is sent to all ports that correspond to clauses that contain literal $\neg z_i$. Thus there exists $\sigma' \in L(M)$ such that for all $0 \leq p \leq r + k$ we have that $\pi_p(\sigma') = \sigma_p$ if and only if there exist an assignment to the boolean variables $z_1, \ldots, z_r$ such that each C_i contains exactly one true literal. The result follows from the three-in-one SAT problem being NP-hard and the fact that it is possible to construct M and the σ_i in polynomial time. ■

We have conditions under which the oracle problem can be solved in polynomial time for DFSMs: we simply use controllable input sequences. While this does not work with NFSMs, we can add a condition that makes is sufficient.

Definition 7 *The NFSM $M = (S, s_0, X, Y, h)$ is* locally observable *if for every state s and input x there exists a port $p \in \mathcal{P}$ such that for all $(s', y) \in h(s, x)$ we have that $\pi_p(y) \neq -$ and for all $(s', y'), (s'', y'') \in h(s, x)$ with $(s', y') \neq (s'', y'')$ we have that $\pi_p(y') \neq \pi_p(y'')$.*

The intuition behind this is that if an NFSM is locally observable then we can look at the output at one port, in response to an input, and determine what the overall output should have been. This clearly simplifies the oracle problem: if an NFSM is locally observable, we have a set of local traces and we know which input was first then from the first output at the appropriate port we can also determine what output must have been produced in response to this input if there was no failure. Thus, if we have a controllable input sequence then we can repeat this process.

Proposition 18 *If Algorithm 2 is given a locally observable FSM M with n transitions and a set of local traces $\sigma_1, \ldots, \sigma_m$ with k inputs that was produced by applying a controllable input sequence then it operates in time that is of $O(k(m + \log(n)))$.*

Proof: First observe that since a controllable input sequence of length k was used and M is locally observable, on each iteration the current set Z_i contains at most one tuple. We can assume that when an input x is considered from state s we know which local trace to study in order to determine the output that must have been produced in response to x and thus the computation within the loop takes $\log(n)$ to locate the appropriate transition and $O(m)$ to compute the value to place in Z_i. Since there are k iterations, the result thus follows. ■

Thus, when testing from an NFSM with distributed ports it is desirable to use controllable input sequences and for the NFSM to be locally observable. However, this places a restriction on the entire NFSM and instead it is sufficient for the input sequences used in testing to lead to paths through the NFSM that have a similar property. The following achieves this by placing a condition on the input sequences used.

Definition 8 *Given FSM M an input sequence $x_1, \ldots, x_k$ is* strongly controllable *for M if the following hold:*

1) $x_1, \ldots, x_k$ *is controllable for* M*; and*
2) *for all* $1 \leq i < k$*, if there is a path from* s_0 *to state* s *with a label that has input portion* $x_1, \ldots, x_{i-1}$ *then there is a port* $p \in \mathcal{P}$ *such that for all* $(s', y) \in h(s, x_i)$ *we have that* $\pi_p(y) \neq -$ *and for all* $(s', y'), (s'', y'') \in h(s, x_i)$ *with* $(s', y') \neq (s'', y'')$ *we have that* $\pi_p(y') \neq \pi_p(y'')$*.*

If an input sequence is strongly controllable then at each point the tester to apply the next input is aware of when to apply the input since the input sequence is controllable. As a result, when considering the oracle problem at each point we know which input is applied next. In addition, the next output produced at an appropriate $p \in \mathcal{P}$ identifies the transition that occurred and so in Algorithm 2 the new set Z_i formed contains at most one tuple. As a result, the proof of the following result is equivalent to that of Proposition 18.

Proposition 19 *If Algorithm 2 is given an FSM* M *with* n *transitions and a set of local traces* $\sigma_1, \ldots, \sigma_m$ *with* k *inputs that was produced by applying a strongly controllable input sequence then it operates in time that is of* $O(k(m + \log(n)))$*.*

The concepts of an input sequence being strongly controllable and an FSM being locally observable are related.

Proposition 20 *If FSM* M *is locally observable then every controllable input sequence is strongly controllable for* M*.*

Proof: We will assume that M is locally observable and consider some controllable input sequence $x_1, \ldots, x_k$: it is sufficient to prove that this input sequence is strongly controllable for M.

Let $1 \leq i < k$ and let s be such that there is a path from s_0 to state s with a label that has input portion $x_1, \ldots, x_{i-1}$. Then it is sufficient to prove that there is a port $p \in \mathcal{P}$ such that for all $(s', y) \in h(s, x_i)$ we have that $\pi_p(y) \neq -$ and for all $(s', y'), (s'', y'') \in h(s, x_i)$ with $(s', y') \neq (s'', y'')$ we have that $\pi_p(y') \neq \pi_p(y'')$. Since M is locally observable, for every state s and input x there exists a port $p \in \mathcal{P}$ such that for all $(s', y) \in h(s, x)$ we have that $\pi_p(y) \neq -$ and for all $(s', y'), (s'', y'') \in h(s, x)$ with $(s', y') \neq (s'', y'')$ we have that $\pi_p(y') \neq \pi_p(y'')$. The result therefore follows. ∎

The notion of an FSM being locally observable could potentially be seen as a testability property: a property that makes testing easier. However, where such a property has not been deliberately designed into a system it seems extremely strong and instead it is more likely that we will be able to test using strongly controllable input sequences, the challenge being to produce strongly controllable input sequences that satisfy a given test criterion.

VII. Conclusions

If a system has physically distributed interfaces, called ports, then in testing and in use observations are made locally. Thus, we observe a local trace at each interface rather than a global trace. This form of observation is strictly weaker than when we observe global traces and leads to new notions of conformance. This paper has considered testing from a (possibly nondeterministic) finite state machine (FSM) and two corresponding conformance relations. One conformance relation $\sqsubseteq_w$ involves simply comparing each observed local trace with a projection of the specification and represents the situation in which no agent can receive information regarding observations made at more than one port. A stronger conformance relation $\sqsubseteq_s$ corresponds to the situation in which an agent might have access to the local traces observed at all of the ports.

The conformance relations $\sqsubseteq_w$ and $\sqsubseteq_s$ have previously been defined. However, in testing we also need to determine whether an observation (set of local traces) is consistent with the specification and this is the oracle problem. This paper has given algorithms for solving the oracle problem for $\sqsubseteq_w$ and $\sqsubseteq_s$. We showed that the oracle problem can be solved in low order polynomial time for $\sqsubseteq_w$ but is NP-hard for $\sqsubseteq_s$. This result holds even if the FSM is deterministic. We then investigated conditions under which the oracle problem for $\sqsubseteq_s$ can be solved efficiently. We proved that if we are testing from a deterministic FSM with input sequences that satisfy the traditional notion of controllability then the oracle problem can be solved in low order polynomial time. We gave stronger sufficient conditions for nondeterministic FSMs: either the FSM is locally observable or the input sequence is strongly controllable. When it is not feasible to solve the oracle problem when using $\sqsubseteq_s$ we can instead use the algorithm for $\sqsubseteq_w$ since this provides a sound approximation.

There are many avenues for future work. First, while we have given conditions under which the oracle problem for $\sqsubseteq_s$ can be solved in polynomial time, these are not necessary conditions. It would therefore be interesting to develop weaker sufficient conditions. We have shown that an oracle for $\sqsubseteq_w$ defines a conservative approximation for $\sqsubseteq_s$ and there may be scope to develop better conservative approximations. There has been work on adapting the $ioco$ conformance relation, traditionally used with input output transition systems (IOTSs), to the scenario in which we only make local observations [20], [21] and it would be interesting to investigate the oracle problem for such conformance relations. However, since IOTSs can have an infinite number of states and input and output need not alternate, it seems likely that strong restrictions will be required in order to allow polynomial time solutions to the oracle problem for IOTSs. Finally, it would be interesting to extend this work to formalisms in which a transition is triggered by a set of inputs rather than a single input (see, for example, [36], [37]).

ACKNOWLEDGMENT

I would like to thank the anonymous referees for their many helpful comments, which have strengthened the paper. I would also like to thank Manuel Núñez and Mercedes Merayo for discussions regarding distributed testing and for producing the example adapted for this paper.

REFERENCES

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

[2] 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.

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

[4] 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.

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

[6] 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.

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

[8] 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.

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

[10] E. Farchi, A. Hartman, and S. Pinter, "Using a model-based test generator to test for standard conformance," *IBM systems journal*, vol. 41, no. 1, pp. 89–110, 2002. [Online]. Available: www.research.ibm.com/journal/sj/411/farchi.pdf

[11] 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*, ser. Lecture Notes in Computer Science, vol. 2931. Montreal, Canada: Springer-Verlag, 2003, pp. 252–266.

[12] 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.

[13] 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.

[14] E. P. Moore, “Gedanken-experiments,” in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton University Press, 1956.

[15] A. Petrenko, S. Boroday, and R. Groz, “Confirming configurations in EFSM testing,” *IEEE Transactions on Software Engineering*, vol. 30, no. 1, pp. 29–42, 2004.

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

[17] 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.

[18] 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.

[19] D. Lee and M. Yannakakis, “Principles and methods of testing finite-state machines - a survey,” *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1089–1123, 1996.

[20] R. M. Hierons, M. G. Merayo, and M. Núñez, “Implementation relations for the distributed test architecture,” in *20th IFIP TC 6/WG 6.1 International Conference on the Testing of Software and Communicating Systems (TestCom/FATES 2008)*, ser. Lecture Notes in Computer Science, vol. 5047. Springer, 2008, pp. 200–215.

[21] ——, “Implementation relations and test generation for systems with distributed interfaces,” *submitted*.

[22] R. M. Hierons, “Controllable testing from nondeterministic finite state machines with multiple ports,” vol. submitted.

[23] J. L. Jacob, “Refinement of shared systems,” in *The Theory and Practice of Refinement: Approaches to the Formal Development of Large-Scale Software Systems*, J. McDermid, Ed. Butterworths, 1989, pp. 27–36.

[24] C. Kloukinas, G. Spanoudakis, and K. Mahbub, “Estimating event lifetimes for distributed runtime verification,” in *Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE’2008)*. Knowledge Systems Institute Graduate School, 2008, pp. 117–122.

[25] L. Cacciari and O. Rafiq, “Controllability and observability in distributed testing,” *Information and Software Technology*, vol. 41, no. 11–12, pp. 767–780, 1999.

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

[27] A. Bauer, M. Leucker, and C. Schallhart, “Model-based runtime analysis of distributed reactive systems,” in *17th Australian Software Engineering Conference (ASWEC 2006)*. IEEE Computer Society, 2006, pp. 243–252.

[28] P. S. Dodd and C. V. Ravishankar, “Monitoring and debugging distributed real-time programs,” *Software, Practice and Experience*, vol. 22, no. 10, pp. 863–877, 1992.

[29] D. Gunter, B. Tierney, B. Crowley, M. Holding, and J. Lee, “Netlogger: A toolkit for distributed system performance analysis,” in *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE Computer Society, 2000, pp. 267–273.

[30] M. Mansorui-Samani and M. Sloman, “Monitoring distributed systems,” *IEEE Network*, pp. 20–30, 1993.

[31] M. Zulkernine and R. E. Seviora, “A compositional approach to monitoring distributed systems,” in *International Conference on Dependable Systems and Networks (DSN 2002)*. IEEE Computer Society, 2002, pp. 763–772.

[32] E. Brinksma, L. Heerink, and J. Tretmans, “Factorized test generation for multi-input/output transition systems,” in *IFIP TC6 11th International Workshop on Testing Communicating Systems (IWTCS)*, ser. IFIP Conference Proceedings, vol. 131. Kluwer, 1998, pp. 67–82.

[33] L. Heerink and J. Tretmans, “Refusal testing for classes of transition systems with inputs and outputs,” in *Formal Description Techniques and Protocol Specification, Testing and Verification (FORTE X/PSTV XVII)*, ser. IFIP Conference Proceedings, vol. 107. Chapman & Hall, 1997, pp. 23–38.

[34] Z. Li, J. Wu, and X. Yin, “Testing multi input/output transition system with all-observer,” in *16th IFIP International Conference on Testing of Communicating Systems (TestCom 2004)*, ser. Lecture Notes in Computer Science, vol. 2978. Springer, 2004, pp. 95–111.

[35] T. J. Schaefer, “The complexity of satisfiability problems,” in *Tenth Annual ACM Symposium on Theory of Computing (STOC)*, 1978, pp. 216–226.

[36] S. Haar, C. Jard, and G.-V. Jourdan, “Testing input/output partial order automata,” in *(TestCom/FATES 2007)*, ser. Lecture Notes in Computer Science, vol. 4581. Springer, 2007, pp. 171–185.

[37] G. von Bochmann, S. Haar, C. Jard, and G.-V. Jourdan, “Testing systems specified as partial order input/output automata,” in *20th IFIP TC 6/WG 6.1 International Conference on Testing of Software and Communicating Systems (TestCom/FATES)*, ser. Lecture Notes in Computer Science, vol. 5047. Springer, 2008, pp. 169–183.

PLACE
PHOTO
HERE

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.