Verifying and Comparing Finite State Machines for Systems that have Distributed Interfaces

Robert M. Hierons *Senior Member, IEEE*

***Abstract*—This paper concerns state-based systems that interact with their environment at physically distributed interfaces, called ports. When such a system is used a projection of the global trace, a local trace, is observed at each port. As a result the environment has reduced observational power: the set of local traces observed need not define the global trace that occurred. We consider the previously defined implementation relation $\sqsubseteq_s$ and prove that it is undecidable whether $N \sqsubseteq_s M$ and so it is also undecidable whether testing can distinguishing two states or FSMs. We also prove that a form of model-checking is undecidable when we have distributed observations and give conditions under which $N \sqsubseteq_s M$ is decidable. We then consider implementation relation $\sqsubseteq_s^k$ that concerns input sequences of length k or less. If we place bounds on k and the number of ports then we can decide $N \sqsubseteq_s^k M$ in polynomial time but otherwise this problem is NP-hard.**

***Keywords*-D2.4: Software Engineering/Software/Program Verification, D2.5: Software Engineering/Testing and Debugging, distributed systems, finite state machine, distributed test architecture.**

I. Introduction

Many systems interact with their environment at multiple physically distributed interfaces, called ports, with web-services, communications protocols, cloud systems and wireless sensor networks being important classes of such systems. When we test such a system we place a local tester at each port and the local tester at port p only observes the events at p. This has led to the (ISO standardised) definition of the distributed test architecture in which we have a set of distributed testers, the testers do not communicate with one another during testing, and there is no global clock [1]. While it has been shown that it is sometimes possible to make testing more effective by allowing the testers to exchange coordination messages during testing (see, for example, [2]), this is not always feasible and the distributed test architecture is typically simpler and cheaper to implement. Importantly, the situation in which separate agents (users or testers) interact with the system at its ports can correspond to the expected use of the system.

Distributed systems often have a persistent internal state and such systems are thus represented using state-based languages. In the context of testing the focus has largely been on finite state machines (FSMs) and input output transition systems (IOTSs), with IOTSs being labelled transition systems (LTSs) where we distinguish between input and output. The interest in FSMs and IOTSs is partly due to them being suitable for representing state-based systems. In addition, many tools and techniques for model-based testing[1] transform a model, written in a high-level notation, to an FSM or IOTS and test from this (see, for example, [3], [4], [5], [6]). Model-based testing has received much attention since it facilitates test automation, the results of a recent major industrial project showing the potential for significant cost reductions [7].

The approach of testing from a formal model, such as an FSM or IOTS, is often described as formal testing. Given a formal model M, ideally we wish to produce a test suite that has some desirable properties such as being guaranteed to find certain types of faults. In order to reason about testing it is normal to assume that the system under test (SUT) behaves like an unknown model N, typically described using the same formalism as the specification M: an approach used originally by Moore [8] that has been formalised and generalised by Gaudel[2] [9]. Once we have made this assumption, that the SUT behaves like an unknown model N written in a given formalism, we can say what it means for the SUT to be correct by

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

[1]In model-based testing, test automation is based on a model of the expected behaviour of the system or some aspect of this expected behaviour.

[2]Gaudel calls this the minimal hypothesis.

defining the required relationship between M and N; this relationship is usually called either an *implementation relation* or a *conformance relation*. If N and M are related under the implementation relation used then N is said to *conform* to M. Naturally, the implementation relation used should reflect the observational power of the environment: given specification M, if it is not possible to distinguish between two models N_1 and N_2 through interacting with them then either both should conform to M or neither should conform to M. Good description of formal testing have been produced by a number of authors including Gaudel [9] and Tretmans [6].

This paper concerns verification and testing of multi-port systems. Much of the work in the area of distributed testing has focussed on FSM models (see, for example, [10], [11], [12], [13]), although there has also been work that considers more general models such as IOTSs and variants of IOTSs (see, for example, [14], [15]). While IOTSs are more expressive, this paper explores decidability and complexity issues in distributed testing and so we restrict attention to multi-port FSMs. Naturally, the negative decidability and complexity results proved in this paper extend immediately to IOTSs.

When a state-based system interacts with its environment there is a sequence of inputs and outputs called a *global trace*, with the user or tester at a port p only observing the sequence of events at p (a *local trace*). It is known that this introduces additional controllability and observability problems in testing (see, for example, [10], [11], [14], [12], [13]). A controllability problem occurs when a tester does not know when to supply an input due to it not observing the events at the other ports [12], [10]. Consider, for example, the global trace shown in Figure 1. We use diagrams (Message Sequence Charts) such as this to represent scenarios. In such diagrams vertical lines represent processes and time progresses as we go down a line. In this case the system under test (SUT) has two ports, 1 and 2, we have one vertical line representing the SUT, one representing the local tester at port 1, and one representing the local tester at port 2. There is a controllability problem because the tester at port 2 should send input x' after y has been sent by the SUT but cannot know when this has happened since it does not observe the events at port 1 and there are no communications between the testers.

Observability problems refer to the fact that the

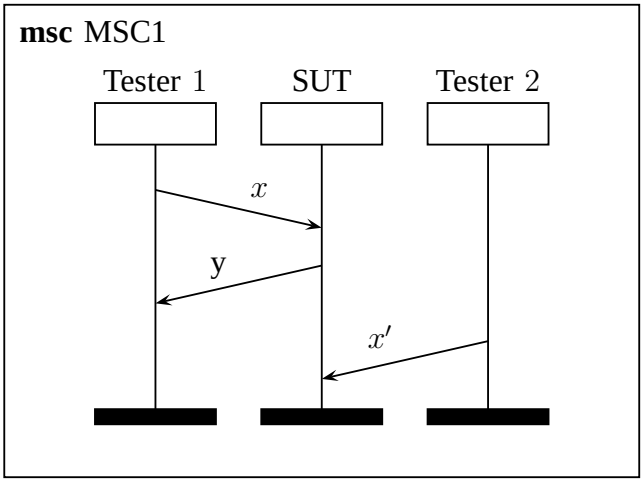

Fig. 1. A controllability problem caused by input x'

observational ability of a set of distributed testers is less than that of a global tester since the local traces observed need not uniquely define the global trace that occurred [11]. Consider, for example, global traces σ and σ' shown in Figures 2 and 3 respectively. The global traces are different but the local testers observe the same local traces: in each case the tester at port 1 observes $xyxy$ and the tester at port 2 observes y'.

Controllability problems lead to situations in which the testers cannot know whether a particular input sequence has been received by the SUT and observability problems lead to the testers not being able to determine the output sequence produced. Thus, both affect the notion of conformance used. Recent work has defined new notions of conformance (implementation relations) that recognise this reduced observational power and these have been defined for FSMs [16] and IOTSs [15]. These implementation relations essentially say that the SUT conforms to the specification if the environment (or set of testers) cannot distinguish observations made from behaviours allowed by the specification. If global trace σ of the SUT is observationally equivalent to one in the specification then σ is considered to be an allowed behaviour since a set of distributed testers/users would not observe a failure.

Given multi-port FSMs N and M, there are two notions of conformance for situations in which distributed observations are made: weak conformance ($\sqsubseteq_w$) and strong conformance ($\sqsubseteq_s$). Under $\sqsubseteq_w$, it is sufficient that for every global trace σ of N and port p there is a global trace σ_p of M such that σ and σ_p

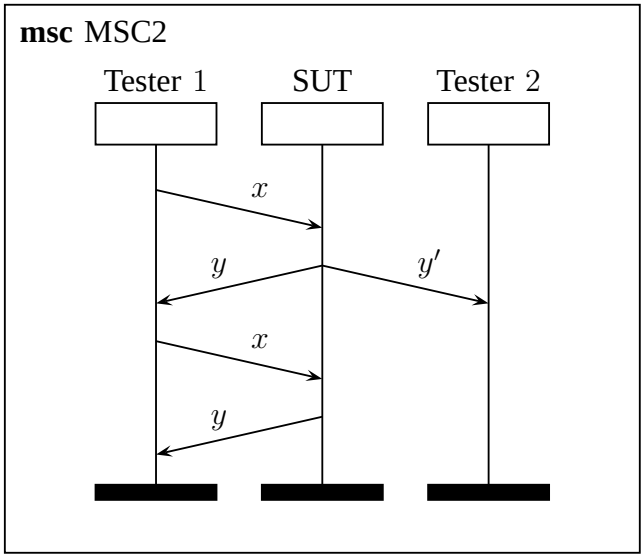

Fig. 2. Global trace σ.

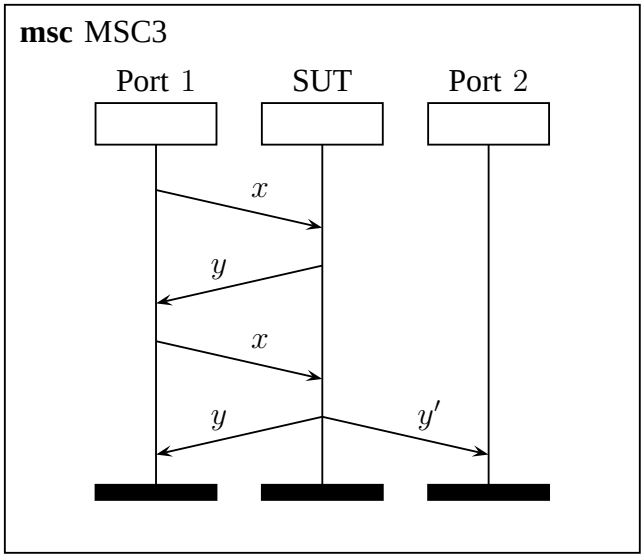

Fig. 3. Global trace σ'.

are indistinguishable at port p; they have the same local traces at p. A similar notion has been discussed for Message Sequence Charts (MSCs), where the weak closure of a language has been defined [17]. In contrast, under $\sqsubseteq_s$ we require that for every global trace σ of N there is some global trace σ' of M such that σ and σ' are indistinguishable at all of the ports. To see the difference, let us suppose that there are two allowed responses to input x_1 at port 1: either y_1 at port 1 and y_2 at port 2 (global trace σ) or y_1' at port 1 and y_2' at port 2 (global trace σ'). Under $\sqsubseteq_w$ it is acceptable for the SUT to respond to x_1 with y_1 at port 1 and y_2' at port 2 since the local trace at port 1 is x_1y_1, which is a projection of σ, and the local trace at port 2 is y_2', which is a projection of σ'. However, this is not acceptable under $\sqsubseteq_s$ since no global trace of the specification has projection x_1y_1 and y_2'.

One of the benefits of using an FSM when there is only one port is that there are standard algorithms for many problems that are relevant to test generation. For example, we can decide whether there are strategies (test cases) that reach or distinguish states [18] and such strategies are used by many test generation algorithms (see, for example, [19], [20], [21], [22]). In addition, if we have an FSM specification M and an FSM design N then we can decide whether N conforms to M. Thus, if we wish to adapt standard FSM test techniques to the situation where we have distributed testers then we need to investigate corresponding problems for multi-port FSMs. Recent work has shown that it is undecidable whether there is a strategy that is guaranteed to reach a state or that distinguishes two states of an FSM in distributed testing [23]. However, this left open the question of whether one can decide whether one FSM conforms to another. It also left open the related question of whether it is decidable whether there is a strategy that is capable of distinguishing two FSMs of two states of an FSM[3]. There also appears to have been no work on model checking for such models.

This paper concerns two main problems. The first is that of deciding, for multi-port FSMs M and N, whether N conforms to M. This can be decided in low order polynomial time for $\sqsubseteq_w$: for each port p we simply compare the projections of N and M at p. However, $\sqsubseteq_w$ is often too weak since it assumes that the agents at the separate ports cannot log their observations and communicate these to a common agent. We therefore focus on the implementation relation $\sqsubseteq_s$. The second problem relates to a type of model checking where we have a model M and a finite automaton P defining a property and we want to know whether any observation that might be made of M is in the language defined by P.

We prove that it is generally undecidable whether $N \sqsubseteq_s M$ for multi-port FSMs N and M but we also give some conditions under which $N \sqsubseteq_s M$ is decidable. This problem is important when we are

[3]It is decidable whether there is a strategy that is capable of reaching a given state of an FSM.

checking an FSM design against an FSM specification. In addition, $N \sqsubseteq_s M$ if no *possible* behaviour of N can be distinguished from the behaviours of M. Thus, it is also undecidable whether there is a test case that is *capable* of distinguishing two states or FSMs. This complements the result that it is undecidable whether there is a test case that is guaranteed to distinguish two states or FSMs [23]. However, the proofs use very different approaches: the proof of the previous result [23] used results from multi-player games while in this paper we use results regarding multi-tape automata. Note that many traditional methods for testing from an FSM use sequences that distinguish between states, in order to check that a (prefix of a) test case takes the SUT to a correct state (see, for example, [19], [20], [24], [21], [22]). The results in this paper and in [23] suggest that it will be difficult to adapt such techniques for distributed testing.

In addition to considering conformance, we also investigate two forms of model checking given model M and property P. One problem involves asking whether any of the observations that might be made of M (sets of local traces) are consistent with sequences in the regular language defined by P. The second problem asks whether any of the observations that might be made of M could also be made when interacting with P through distributed interfaces. It transpires that both types of model checking are undecidable.

Since it is undecidable whether $N \sqsubseteq_s M$, we define a weaker implementation relation $\sqsubseteq_s^k$ that considers sequences of length k or less. This is relevant when we know a bound on the length of sequences in use or we know that the system will be reset after at most k inputs have been received. For example, a protocol might have a bound on the number of steps that can occur before a 'disconnect' happens. In addition, embedded systems are often designed to repeat a sequence of activities in a schedule, returning to the initial state at the end of such a sequence: we might use the bound defined by this (see, for example, [25]). It is also relevant if we want a test case of length at most k that is capable of distinguishing two FSMs or states. Naturally, it is decidable whether $N \sqsubseteq_s^k M$ since it is sufficient to reason about finite sets of global traces. We prove that if we place a bound on k and the number of ports then we can decide whether $N \sqsubseteq_s^k M$ in polynomial time but the problem is NP-hard without such bounds.

There are several factors that make results, regarding strong conformance, highly relevant to distributed testing. First, they provide information regarding implementation relations for testing distributed systems. Given a model N' that represents a possible SUT and specification M we might want to know whether there is a test that is capable of distinguishing N' from M in testing and this is the case if and only if we do not have that $N' \sqsubseteq_s M$. This is important if we produce a set of models that represent possible behaviours of the SUT, such a set often being called a *fault domain*. A fault domain might be explicitly generated, as is done in approaches to mutation testing (see, for example, [26]), or it may be implicit. For example, there are test generation algorithms that take an FSM M and return a test (called a checking experiment) that is guaranteed to determine whether the unknown FSM N that models the SUT conforms to M as long as N has no more than m states for some given m (see, for example, [8], [20], [21]). As noted above, many FSM based test techniques use tests that distinguish states of the specification FSM M in order to check that a transition takes the SUT to the correct state; two states s_1 and s_2 can only be distinguished in distributed testing if we do not have that the FSMs formed by starting M in states s_1 and s_2 conform to one another under $\sqsubseteq_s$.

This paper is structured as follows. Section II provides preliminary definitions. In Section III we discuss results regarding multi-tape automata that we use in Section IV to prove the decidability results. Section IV also gives conditions under which $N \sqsubseteq_s M$ is decidable. In Section V we explore $\sqsubseteq_s^k$. Finally, in Section VI we draw conclusions and discuss possible lines of future work.

II. Preliminaries

This paper concerns the testing of state-based systems whose behaviour is characterised by the input/output sequences (global traces) that they can produce. Given a set A we let A^* denote the set of sequences formed from elements of A and we let ϵ denote the empty sequence. In addition, A^+ denotes the set of non-empty sequences in A^*. Given sequence $\sigma \in A^*$ we let $\text{pref}(\sigma)$ denote the set of prefixes of σ. We are interested in finite state machines, which define global traces (input/output

sequences). Given a global trace $\sigma = x_1y_1 \dots x_ky_k$, in which $x_1, \dots, x_k$ are inputs and $y_1, \dots, y_k$ are outputs, the prefixes of σ are the global traces of the form $x_1y_1 \dots x_jy_j$ with $0 \leq j \leq k$.

In this paper we investigate the situation in which a system interacts with its environment at n physically distributed interfaces, called ports. We let $\mathcal{P} = \{1, \dots, n\}$ denote the names of these ports. A *multi-port FSM* M is defined by a tuple (S, s_0, I, O, h) in which S is the finite set of states, $s_0 \in S$ is the initial state, I is the finite input alphabet, O is the finite output alphabet, and h is the transition relation. The set of inputs is partitioned into subsets $I_1, \dots, I_n$ such that for $p \in \mathcal{P}$ we have that I_p is the set of inputs that can be received at port p. Similarly, for port p we let O_p denote the set of outputs that can be observed at p. As is usual we allow an input to lead to outputs at several ports and so we let $O = ((O_1 \cup \{-\}) \times \dots \times (O_n \cup \{-\}))$ in which $-$ denotes null output. We ensure that the O_p are pairwise disjoint by labelling with port names, where necessary. We let $\mathcal{A}ct = I \cup O$ denote the set of possible observations and for $p \in \mathcal{P}$ we let $\mathcal{A}ct_p = I_p \cup O_p$ denote the set of possible observations at port p.

The transition relation h is of type $S \times I \to 2^{S \times O}$ and should be interpreted as follows: if $(s', y) \in h(s, x)$, $y = (z_1, \dots, z_n)$, and M receives input x when in state s then it can move to state s' and send output z_p to port p (all $p \in \mathcal{P}$). This defines the *transition* $(s, s', x/y)$, which is a *self-loop transition* if $s = s'$. Since we only consider multi-port FSMs in this paper we simply call them FSMs. The FSM M is said to be a *deterministic FSM (DFSM)* if $|h(s, x)| \leq 1$ for all $s \in S$ and $x \in I$.

FSM M is completely-specified if for every state s and input x, we have that $h(s, x) \neq \emptyset$. A sequence $(s_1, s_2, x_1/y_1)(s_2, s_3, x_2/y_2) \dots (s_k, s_{k+1}, x_k/y_k)$ of consecutive transitions is said to be a *path*, which has *starting state* s_1 and *ending state* s_{k+1}. This path has *label* $x_1y_1 \dots x_ky_k$, which is called a (global) *trace*. Further, $x_1 \dots x_k$ and $y_1 \dots y_k$ are the *input portion* and the *output portion* respectively of $x_1y_1 \dots x_ky_k$. A path is a *cycle* if its starting and ending states are the same. The FSM M defines the regular language $L(M)$ of the labels of paths of M that have starting state s_0. Given state $s \in S$ of M we let $L_M(s)$ denote the set of global traces that are labels of paths of M with starting state s, and so $L(M) = L_M(s_0)$. We say that M is *initially connected* if for every state s of M there is a path that has starting state s_0 and ending state s. We assume that any FSM considered is completely-specified and initially connected since this simplifies the analysis. Where this condition does not hold we can remove the states that cannot be reached and we can complete the FSM by, for example, either adding self-loop transitions with null output or transitions to an error state.

At times we will use results regarding finite automata (FA) and so we briefly define FA here. A FA M is defined by a tuple (S, s_0, X, h, F) in which S is the finite set of states, $s_0 \in S$ is the initial state, X is the finite alphabet, h is the transition relation, and $F \subseteq S$ is the set of final states. The transition relation has type $S \times (X \cup \{\tau\}) \to 2^S$ where τ represents a silent transition that is not observed. Similar to IOTSs, a sequence $(s_1, s_2, a_1)(s_2, s_3, a_2) \dots (s_k, s_{k+1}, a_k)$ of consecutive transitions is a *path* that has *starting state* s_1 and *ending state* s_{k+1}. The label of this path is the sequence formed by removing all instances of τ from $a_1 \dots a_k$. The FA M defines the language $L(M)$ of labels of paths that have starting state s_0 and an ending state in F.

For a global trace σ and port $p \in \mathcal{P}$ we let $\pi_p(\sigma)$ denote the *local trace* formed by removing all elements that do not occur at p. This is defined by the following rules in which σ is a global trace and $y = (z_1, \dots, z_n)$ is an output (see, for example, [15]).

$$
\begin{aligned}
\pi_p(\epsilon) &= \epsilon \\
\pi_p(x\sigma) &= \pi_p(\sigma) \text{ if } x \in I_q \text{ for some } q \neq p \\
\pi_p(x\sigma) &= x\pi_p(\sigma) \text{ if } x \in I_p \\
\pi_p(y\sigma) &= \pi_p(\sigma) \text{ if } z_p = - \\
\pi_p(y\sigma) &= z_p\pi_p(\sigma) \text{ if } z_p \neq -
\end{aligned}
$$

Given a set A of global traces and port p we let $\pi_p(A) = \{\pi_p(\sigma) | \sigma \in A\}$ denote the set of projections of sequences in A.

In the distributed test architecture, a local tester at port $p \in \mathcal{P}$ only observes events from $\mathcal{A}ct_p$. Thus, two global traces σ and σ' are indistinguishable if they have the same projections at every port and we denote this $\sigma \sim \sigma'$. More formally, we say that $\sigma \sim \sigma'$ if for all $p \in \mathcal{P}$ we have that $\pi_p(\sigma) = \pi_p(\sigma')$.

Given an FSM M, we let $\mathcal{L}(M) = \{\sigma' | \exists \sigma \in L(M). \sigma \sim \sigma'\}$ denote the set of global sequences

Fig. 4. Finite State Machines M_1 and N_1

that are equivalent to elements of $L(M)$ under $\sim$. These are the sequences that are indistinguishable from sequences in $L(M)$ when distributed observations are made. Previous work has defined two conformance relations for testing from an FSM that reflect the observational power of distributed testing [16]. Sometimes the agents at the separate ports of the SUT will never interact with one another or share information with other agents that can interact. In such cases a global trace is acceptable if the local trace observed at a port p is a local trace of M (all $p \in \mathcal{P}$). This situation is captured by the following.

Definition 1: 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 $\sigma \in L(N)$ and $p \in \mathcal{P}$ there exists $\sigma_p \in L(M)$ with $\pi_p(\sigma_p) = \pi_p(\sigma)$. N is said to *weakly conform* to M.

However, sometimes there is the potential for information from separate testers to be logged and later received by an external agent. For example, there may be a central controller that receives the observations made by each tester once testing is complete. This leads to the following stronger conformance relation.

Definition 2: Given FSMs N and M with the same input and output alphabets and the same set of ports, $N \sqsubseteq_s M$ if for every $\sigma \in L(N)$ there exists $\sigma' \in L(M)$ such that $\sigma' \sim \sigma$. N is said to *strongly conform* to M.

Given FSMs N and M we have that $N \sqsubseteq_s M$ if and only if $L(N) \subseteq \mathcal{L}(M)$ and this is the case if and only if $\mathcal{L}(N) \subseteq \mathcal{L}(M)$. It is also clear that $N \sqsubseteq_s M$ implies that $N \sqsubseteq_w M$. In order to see that $\sqsubseteq_s$ is stronger than $\sqsubseteq_w$ it is sufficient to consider M_1 and N_1 shown in Figure 4. We do not have that $N_1 \sqsubseteq_s M_1$ since M_1 has no global trace equivalent to $x_1(y_1, y'_2)$ under $\sim$. However, for every global trace σ of N_1 and port p there is a global trace σ' of M_1 such that $\pi_p(\sigma) = \pi_p(\sigma')$. Thus, $N_1 \sqsubseteq_w M_1$.

III. Conformance and multi-tape automata

While we can decide (in polynomial time) whether $N \sqsubseteq_w M$, by comparing projections of N and M on different ports, this is quite a weak conformance relation since it does not allow us to bring together local traces observed at the separate ports. It seems likely that normally $\sqsubseteq_s$ will be more suitable and so we consider the problem of deciding whether $N \sqsubseteq_s M$. In this section we study language inclusion for multi-tape automata; in Section IV we use the results described here to show that it is generally undecidable whether $N \sqsubseteq_s M$ for FSMs M and N and also that a type of model checking is undecidable. We first define multi-tape FA [27].

Definition 3: An r-tape FA with disjoint alphabets Σ_i, $1 \leq i \leq r$, $\Sigma = \bigcup_{i=1}^{r} \Sigma_i$, is a tuple (S, s_0, Σ, h, F) in which S is a finite set of states, $s_0 \in S$ is the initial state $F \subseteq S$ is the set of final states and $h : S \times \Sigma \rightarrow 2^S$ is the transition relation.

An r-tape FA N is thus a FA with alphabet Σ that is partitioned into $\Sigma_1, \ldots, \Sigma_r$. As a result, it defines a regular language $L(N)$: the set of labels of paths from the initial state of N that end in a final state. However, it also defines a language of r-tuples: N accepts tuple $(w_1, \ldots, w_r) \in \Sigma_1^* \times \ldots \times \Sigma_r^*$ if and only if there is some sequence $\sigma \in L(N)$ such that $\pi_i(\sigma) = w_i$ for all $1 \leq i \leq r$. We let $\mathcal{T}(N)$ denote the set of tuples accepted by N.

Deciding whether $N \sqsubseteq_s M$ is similar to deciding whether, for multi-tape FA N' and M', $\mathcal{T}(N')$ is a subset of $\mathcal{T}(M')$. This problem, regarding multi-tape FA, is known to be undecidable [27]. However, the proof of this result uses FA in which not all states are final and in FSMs there is no concept of a state not being a final state.

We now prove that language inclusion is undecidable even if we require all states to be final states.

Theorem 1: Let us suppose that N and M are multi-tape FA in which all states are final states. The following problem is undecidable, even when there are only two tapes: do we have that $\mathcal{T}(N) \subseteq \mathcal{T}(M)$?

Proof: We will show that if we can decide this problem for arbitrary multi-tape FA in which all states are final states then we can prove this problem for arbitrary multi-tape FA that may have states that are not final states. Let us suppose that N_1 and M_1 are multi-tape FA in which there may be states that

are not final states. We will assume that for every state of N_1 and M_1 there is a path to a final state; any state not satisfying this property can be removed. We introduce a new elements to the alphabet of the first tape and call this x. Form N_1' and M_1' from N_1 and M_1 in the following way: from each final state add a transition with label x to a new sink state (with no transitions leaving it) and make all of the states final states. Thus, $L(N_1') = \text{pref}(L(N_1)\{x\})$ and $L(M_1') = \text{pref}(L(M_1)\{x\})$. As a result, $\mathcal{T}(N_1) \subseteq \mathcal{T}(M_1)$ if and only if $\mathcal{T}(N_1') \subseteq \mathcal{T}(M_1')$. Thus, if $\mathcal{T}(N_1') \subseteq \mathcal{T}(M_1')$ is decidable for multi-tape FA in which all states are final then $\mathcal{T}(N_1) \subseteq \mathcal{T}(M_1)$ is decidable for multi-tape FA. The result thus follows from this latter problem being undecidable [27]. ■

Observe that this makes it straightforward to show that it is undecidable whether $\mathcal{L}(N) \subseteq \mathcal{L}(M)$ for LTSs (or IOTSs) N and M. In the next section we show how this can be extended to FSMs. We now prove some additional decidability results; these will be used to prove that a type of model checking is undecidable.

Theorem 2: Let us suppose that N and M are multi-tape FA in which all states are final states. Then it is undecidable whether $\mathcal{T}(N) \cap \mathcal{T}(M)$ contain a tuple in which one or more components are non-empty.

Proof: It is known that given multi-tape FA M_1 and N_1, it is undecidable whether $\mathcal{T}(M_1) \cap \mathcal{T}(N_1) = \emptyset$ [27]. We define M_1' and N_1' in which we add new tapes $r+1$ and $r+2$ with alphabets $\{x\}$ and $\{x'\}$ respectively for symbols x and x' not used in N_1 and M_1. We define M_1' and N_1' in the following way.

- Form M_1' from M_1 by adding a new start state with a single transition, with label x', from this to the start state of M_1 and by adding a transition with label x from each final state to a new sink state (with no transitions leaving it) and make all of the states final states.
- Form N_1' from N_1 by adding a new start state with a single transition, with label x, from this to the start state of N_1 and by adding a transition with label x' from each final state to a new sink state (with no transitions leaving it) and make all of the states final states.

Thus, $L(N_1') = \text{pref}(\{x\}L(N_1)\{x'\})$ and $L(M_1') = \text{pref}(\{x'\}L(M_1)\{x\})$. It is clear that $\mathcal{T}(M_1') \cap \mathcal{T}(N_1')$ contain a tuple in which one or more components are non-empty if and only if $\mathcal{T}(M_1) \cap \mathcal{T}(N_1) \neq \emptyset$. The result thus follows from it being undecidable for multi-tape FA M_1 and N_1 whether $\mathcal{T}(M_1) \cap \mathcal{T}(N_1) = \emptyset$. ■

This result shows that a type of model checking is undecidable for LTSs and IOTSs. Specifically, given a model M and a property defined by FA P we can define the following type of model checking: deciding whether $\mathcal{L}(M) \cap \mathcal{L}(P)$ contains a non-empty sequence. Here M is a model and P a property and we wish to know whether any observations that might be made of M might also be made of P. We have seen that this is undecidable[4].

Model checking has been considered for high level message sequence charts (HMSCs), where we ask whether the *sets of linearisations* of HMSCs M and P intersect [17], [28]. This is conceptually similar to the problem above. There has also been work in the context of trace theory [29] but this previous work does not require that all of the states of M are final states.

Finally, we prove that equivalence is undecidable for multi-tape FA in which all states are final states.

Theorem 3: Let us suppose that N and M are multi-tape FA in which all states are final states. The following problem is undecidable, even when there are only two tapes: do we have that $\mathcal{T}(N) = \mathcal{T}(M)$?

Proof: First observe that given sets A and B we have that $A \subseteq B$ if and only if $A \cup B = B$. Let us suppose that we have multi-tape automata N_1 and N_2 with the same numbers of tapes and the same alphabets and assume that all states of N_1 and N_2 are final states.

We will first show that we can construct a multi-tape automaton N_3 such that $\mathcal{T}(N_3) = \mathcal{T}(N_1) \cup \mathcal{T}(N_2)$ and all states of N_3 are final states. Assume that $N_1 = (S, s_0, \Sigma, h_1, S)$ and $N_2 = (Q, q_0, \Sigma, h_2, Q)$ We will use $A \uplus B$, for sets A and B, to denote the disjoint union of A and B. Then we let N_3 be $(S \uplus Q \uplus \{r_0\}, r_0, h', S \uplus Q \uplus \{r_0\})$ for $r_0 \notin S \uplus Q$, in which h is the union of h_1 and h_2 plus the following transitions: for every $(s_0, a, s) \in h_1$ we include in h' the tuple (r_0, a, s); and for every $(q_0, a, q) \in h_2$ we include in h' the tuple (r_0, a, q).

Thus, if we can decide whether $\mathcal{T}(N) = \mathcal{T}(M)$ for two multi-tape FA that have only final states then we can also decide whether $\mathcal{T}(N_1) \cup \mathcal{T}(N_2) = \mathcal{T}(N_2)$ for two multi-tape FA that only have final

[4]While these results are based on Theorem 2, in which all states of the multi-tape FA are final states, this result immediately generalises to the case where we allow some states not to be final states.

states. However, this holds if and only if $\mathcal{T}(N_1) \subseteq \mathcal{T}(N_2)$. The result thus follows from Theorem 1. ■

IV. Results for Finite State Machines

In this section we consider FSMs. In contrast to finite automata and IOTSs, a transition has an input/output pair as a label and there is an associated atomicity assumption (input cannot be received before output has been produced by the previous transition). We now show how a multi-tape FA can be represented using an FSM before using this to prove results regarding FSMs.

In order to extend Theorem 1 to FSMs we define a function that takes a multi-tape FA and returns an FSM. This construction is similar to one previously produced for single-port models [30].

Definition 4: Let us suppose that $N = (S, s_0, \Sigma, h, S)$ is a FA with r tapes with alphabets $\Sigma_1, \ldots, \Sigma_r$. We define the FSM $\mathcal{F}(N)$ with $r+1$ ports as defined below in which for all $1 \leq p \leq r$ we have that the input alphabet of N at p is Σ_p and the output alphabet is empty and further we have that the input alphabet at port $r+1$ is empty and the output alphabet at $r+1$ is $\{0,1\}$. In the following for $a \in \{0,1\}$ we use a_k to denote the k-tuple whose first $k-1$ elements are empty and whose kth element is a.

$\mathcal{F}(N) = (S \cup \{s_e\}, s_0, \Sigma, \{0_{n+1}, 1_{n+1}\}, h')$ in which $s_e \notin S$, for all $z \in \Sigma$ we have that $h'(s_e, z) = \{(s_e, 0_{r+1})\}$ and for all $s \in S$ and $z \in \Sigma$ we have that $h'(s,z)$ is defined by the following:

1) If $h(s,z) = S' \neq \emptyset$ then $h'(s,z) = \{(s', 1_{r+1}), (s', 0_{r+1}) | s' \in S'\}$;
2) If $h(s,z) = \emptyset$ then $h'(s,z) = \{(s_e, 0_{r+1})\}$.

The idea is that while following a path of N the FSM $\mathcal{F}(N)$ can produce either 0 or 1 at port $r+1$ in response to each input but once we diverge from such a path the FSM can then only produce 0 (at $r+1$) in response to an input. An alternative would for the FSM to only be defined on sequences from $L(N)$ and to output 1 while a path from N is followed. However, this does not result in a completely-specified FSM.

Lemma 1: Let us suppose that N and M are r-tape FA with alphabets $\Sigma_1, \ldots, \Sigma_r$. Then $\mathcal{T}(N) \subseteq \mathcal{T}(M)$ if and only if $\mathcal{F}(N) \sqsubseteq_s \mathcal{F}(M)$.

Proof: First assume that $\mathcal{F}(N) \sqsubseteq_s \mathcal{F}(M)$ and we are required to prove that $\mathcal{T}(N) \subseteq \mathcal{T}(M)$. Assume that $\sigma \in \mathcal{T}(N)$ and so there exists some $\sigma' \sim \sigma$ such that $\sigma' \in L(N)$. Since $\sigma' \in L(N)$ we have that $L(\mathcal{F}(N))$ contains the global trace ρ' in which the input portion is σ' and each output is 1_{r+1}. Since $\mathcal{F}(N) \sqsubseteq_s \mathcal{F}(M)$ we must have that there is some $\rho'' \in L(\mathcal{F}(M))$ such that $\rho'' \sim \rho'$. However, since the outputs are all at port $r+1$ and the inputs are at ports $1, \ldots, r$ we must have that ρ'' has output portion that contains only 1_{r+1} and input portion σ'' for some $\sigma'' \sim \sigma'$. Thus, we must have that $\sigma'' \in L(M)$. Since $\sigma \sim \sigma'$ and $\sigma' \sim \sigma''$ we must have that $\sigma \in \mathcal{T}(M)$ as required.

Now assume that $\mathcal{T}(N) \subseteq \mathcal{T}(M)$ and we are required to prove that $\mathcal{F}(N) \sqsubseteq_s \mathcal{F}(M)$. Let ρ be some element of $L(\mathcal{F}(N))$ and it is sufficient to prove that $\rho \in \mathcal{L}(\mathcal{F}(M))$. Then $\rho = \rho_1\rho_2$ for some maximal ρ_2 such that all outputs in ρ_2 are 0_{r+1}. Let the input portions of ρ_1 and ρ_2 be σ_1 and σ_2 respectively. By the maximality of ρ_2 we must have that ρ_1 is either empty or ends in output 1_{r+1}. Thus, $\sigma_1 \in L(N)$ and so, since $\mathcal{T}(N) \subseteq \mathcal{T}(M)$, there exists $\sigma_1' \sim \sigma_1$ with $\sigma_1' \in L(M)$. But this means that M can produce the output portion of ρ_1 in response to σ_1' and so there exists $\rho_1' \in L(\mathcal{F}(M))$ with $\rho_1' \sim \rho_1$. By the definition of $\mathcal{F}(M)$, since all outputs in ρ_2 are 0_{r+1} we have that $\rho' = \rho_1'\rho_2 \in L(\mathcal{F}(M))$. The result therefore follows from observing that $\rho' = \rho_1'\rho_2 \sim \rho_1\rho_2 = \rho$. ■

Theorem 4: The following problem is undecidable: given FSMs N and M with the same alphabets, do we have that $N \sqsubseteq_s M$?

Proof: This follows from Lemma 1 and Theorem 1. ■

We can extend this to show that state equivalence[5] is undecidable.

Theorem 5: The following problem is undecidable: given FSM M and two states s and s' of M, are s and s' equivalent.

Proof: We will prove that we can express the problem of deciding multi-port FSMs equivalence in terms of state equivalence. We assume that we have multi-port FSMs M_1 and M_2 with the same input and output alphabets and we wish to decide whether M_1 and M_2 are equivalent. Let s_{01} and s_{02} be the initial states of M_1 and M_2 respectively. We will construct an FSM M in the following way. We add a new port p and input x_p at p. The input of x_p in the initial state s_0 of M moves M to either s_{01} or s_{02}

[5]Two states s_1 and s_2 of an FSM M are equivalent if the FSMs M_1 and M_2 formed by changing the initial state of M to be s_1 and s_2 respectively are equivalent: $\mathcal{L}(M_1) = \mathcal{L}(M_2)$.

and produces no output. All other input in state s_0 moves M to a state $s'_0 \neq s_0$, that is not a state of M_1 or M_2, from which all transitions are self-loops with no output. The input of x_p in a state of M_1 or M_2 leads to no output and no change of state. Now we can observe that a sequence in the language defined by starting M in state s_{0i}, $i \in \{0,1\}$, is equivalent under $\sim$ to a sequence from $\mathcal{L}(M_i)$ followed by a sequence of zero or more instances of x_p. Thus, s_{01} and s_{02} are equivalent if and only if M_1 and M_2 are equivalent. The result thus follows from Theorem 4 and the fact that if we can decide equivalence then we can decide inclusion. ■

We now consider problems relating to distinguishing FSMs and states in testing. We can only distinguish between FSMs and states on the basis of observations and each observation, in distributed testing, defines an equivalence class of $\sim$.

Definition 5: It is possible to distinguish FSM N from FSM M in distributed testing if and only if $\mathcal{L}(N) \not\subseteq \mathcal{L}(M)$. Further, it is possible to distinguish between FSMs N and M in distributed testing if and only if $\mathcal{L}(N) \not\subseteq \mathcal{L}(M)$ and $\mathcal{L}(M) \not\subseteq \mathcal{L}(N)$.

The first part of the definition says that we can only distinguish N from M if there is some global trace of N that is not observationally equivalent to a global trace of M. The second part strengthens this by requiring that we can distinguish N from M and also M from N. The following is an immediate consequence of Theorem 4.

Corollary 1: The following problems are undecidable in distributed testing.

- Is it possible to distinguish FSM N from FSM M in distributed testing?
- Is it possible to distinguish between FSMs N and M in distributed testing?

Similar to the proof of Theorem 5, we can express the problem of distinguishing between two FSMs as that of distinguishing two states s and s' of an FSM M. Thus, the above shows that it is undecidable whether there is a test case that is capable of distinguishing two states of an FSM or two FSMs. This complements a previous result [23], that it is undecidable whether there is some test case that is *guaranteed* to distinguish two states or FSMs. It also suggests that it will be difficult to extend traditional methods, for generating tests from FSMs, where they use sequences to distinguish states.

Before we prove that a form of model checking is undecidable for FSMs we define a function similar to $\mathcal{F}$.

Definition 6: Let us suppose that $N = (S, s_0, \Sigma, h, S)$ is a FA with r tapes with alphabets $\Sigma_1, \ldots, \Sigma_r$. Given integer k we define the FSM $\mathcal{F}(N,k)$ with $r+1$ ports as defined below in which for all $1 \leq p \leq r$ we have that the input alphabet of N at p is Σ_p and the output alphabet is empty and further we have that the input alphabet at port $r+1$ is empty and the output alphabet at $r+1$ is $\{1,k\}$. In the following for $a \in \{1,k\}$ we use a_j to denote the j-tuple whose first $j-1$ elements are empty and whose jth element is a.

$\mathcal{F}(N,k) = (S \cup \{s_e\}, s_0, \Sigma, \{1_{r+1}, k_{r+1}\}, h')$ in which $s_e \notin S$, for all $z \in \Sigma$ we have that $h'(s_e, z) = \{(s_e, k_{r+1})\}$ and for all $s \in S$ and $z \in \Sigma$ we have that $h'(s,z)$ is defined by the following:

1) If $h(s,z) = S' \neq \emptyset$ then $h'(s,z) = \{(s', 1_{r+1}) | s' \in S'\}$;
2) If $h(s,z) = \emptyset$ then $h'(s,z) = \{(s_e, k_{r+1})\}$.

The idea is that while following a path of N the FSM $\mathcal{F}(N,k)$ produces 1 at port $r+1$ in response to each input but once we diverge from such a path the FSM then produces k (at $r+1$) in response to an input. The following result is an immediate consequence of the definition.

Lemma 2: Let us suppose that N and M are r-tape FA with alphabets $\Sigma_1, \ldots, \Sigma_r$. Then $\mathcal{T}(N) \cap \mathcal{T}(M)$ contains a tuple with at least one non-empty element if and only if $\mathcal{L}(\mathcal{F}(N,2)) \cap \mathcal{L}(\mathcal{F}(M,3))$ contains a non-empty sequence.

Theorem 6: Given FSMs M and P the following problem is undecidable: do we have that $\mathcal{L}(M) \cap \mathcal{L}(P)$ contains a non-empty sequence?

Proof: These results follow from Theorem 2 and Lemma 2. ■

Finally, we give conditions under which equivalence and inclusion are decidable. The proof of the following uses a results from (Mazurkiewicz) trace theory. We will only use trace theory in proofs (Proposition 1, Proposition 5, and Lemma 3); some readers might skip the following description and just read the results. In trace theory, if the alphabet is Σ then there is a symmetric independence relation $\mathcal{I}$ of type $\Sigma \times \Sigma$. If $(a,b) \in \mathcal{I}$ then a and b are said to be independent and ab and ba are equivalent [31]. Relation $\mathcal{I}$ defines an equivalence relation $\sim_{\mathcal{I}}$: $\sigma, \sigma' \in \Sigma^*$ are equivalent under $\sim_{\mathcal{I}}$ if σ can be rewritten to σ' using a sequence of rewrites of the form $\sigma_1 ab \sigma_2 \rightarrow \sigma_1 ba \sigma_2$ for $(a,b) \in \mathcal{I}$.

Independence relation $\mathcal{I}$ can be represented using an independence graph $G_{\mathcal{I}}$ in which each vertex represents an element of Σ and there is an edge between the vertex representing $a \in \Sigma$ and the vertex representing $b \in \Sigma$ if and only if a and b are independent. Similarly, there is a dependence graph $G_{\mathcal{D}}$ defined by there being an edge between the vertex representing $a \in \Sigma$ and the vertex representing $b \in \Sigma$ if and only if a and b are not independent. For FSMs, a and b are independent if and only if they are at different ports.

Proposition 1: Let us suppose that multi-port FSMs N and M have the same input and output alphabets and that for each port $p \in \mathcal{P}$, $|\mathcal{A}ct_p| \leq 1$. Then it is decidable whether $N \sqsubseteq_s M$.

Proof: This can be seen as being an instance of the inclusion problem for rational trace languages[6], which is decidable if the independence relation is transitive (Theorem 66, [31]). However, since $|\mathcal{A}ct_p| \leq 1$ for all $p \in \mathcal{P}$, the elements of $\mathcal{A}ct$ are pairwise independent and so the independence relation is transitive. The result thus follows. ■

This condition, that the alphabet at each port contains only one symbol, is quite strong and it seems unlikely that many systems will have this property. However, it is relevant if there are properties to be tested that relate to the sequencing of interactions between the ports and not the actual values observed at the ports. For example, there may be a schedule that determines how the system should interact with the agents at its ports and, in turn, this schedule may relate to quality of service agreements that determine the relative amount of access to resources that the system provides to the agents.

We now consider the case where each transition produces output at all ports.

Proposition 2: Let us suppose that M is an FSM in which all transitions produce output at all ports. Then $N \sqsubseteq_s M$ if and only if $L(N) \subseteq L(M)$.

Proof: First observe that if $N \sqsubseteq_s M$ then each transition of N must also produce output at every port. Consider a sequence $\sigma \in L(M) \cup L(N)$ that contains k inputs. Since every transition produces output at all ports, for a port p we have that $\pi_p(\sigma)$ contains k outputs with each input x_i at p being between the output produced at p by the previous input and the output produced at p in response to x_i. Thus, given sequences $\sigma, \sigma' \in L(N) \cup L(M)$ we must have that $\sigma' \sim \sigma$ if and only if $\sigma' = \sigma$. The result therefore holds. ■

Decidability results have been proved for classes of deterministic multi-tape automata, where a multi-tape automaton M with r tapes is deterministic if the state set S can be partitioned into subsets $S_1, \ldots, S_r$ such that all transitions from a state in S_i have a label in the alphabet Σ_i for tape i, $1 \leq i \leq r$. An FSM M can be represented by a multi-tape automaton M' in which the states of M are represented by final states of M' and if M has transition $(s, s', x/y)$ then in M' there is a path from the state that represents s to the state that represents s' with a label σ that starts with x and has that $\pi_p(xy) = \pi_p(\sigma)$ for all $p \in \mathcal{P}$. By definition, for M' to be deterministic we require that the FSM M has no state in which it can receive input at more than one port. Since we consider completely-specified FSMs, this requires that input can only be received at one port.

Proposition 3: Let us suppose that M and N are FSMs that can receive input at only one port. If either every state of M is involved in at most one cycle or every state of N is involved in at most one cycle then it is decidable whether $N \sqsubseteq_s M$.

Proof: A multi-tape FA is simple if no state is in more than one cycle. It is known that language inclusion is decidable for two deterministic multi-tape FA in which at least one is simple [32]. The result thus follows from the fact that, under the conditions in the hypothesis, we can form deterministic multi-tape FA M' and N' to represent M and N and at least one of these is simple. ■

If we are interested in equivalence then we can weaken the conditions on the FSMs.

Proposition 4: If M and N are FSMs that can receive input at only one port then it is decidable whether both $N \sqsubseteq_s M$ and $M \sqsubseteq_s N$ hold.

Proof: It is known that equivalence is decidable for deterministic multi-tape FA [33]. The result thus follows from the fact that, under the conditions in the hypothesis, we can form deterministic multi-tape FA M' and N' to represent M and N. ■

Finally, we will use the result that a rational trace language[7] is a regular language if every cycle (star) in the expression that defines the language

[6] A trace language is rational if it is can be formed from finite sets using a finite number of union, concatenation, and star-iteration operations.

[7] The result is actually proved for a generalisation of traces called semi-commutations.

has a label σ such that the dependence graph $G_{\mathcal{D}}$ restricted to the letters in σ is strongly connected[8] [34]. Importantly, the proof also showed how we can construct a FA M' such that $L(M') = \mathcal{L}(M)$. Thus, if we consider two FSMs that satisfy this condition then the inclusion problem can be reduced to deciding inclusion for two regular languages and so is decidable. For FSMs the dependence relation simply relates all elements at the same port and so we obtain the following result.

Proposition 5: Let us suppose that FSMs M and N are such that every cycle has a label σ such that there is a port p where all inputs and outputs in σ are at p. Then it is decidable whether $N \sqsubseteq_s M$.

V. Bounded conformance

We have seen that it is undecidable whether two FSMs are related under $\sqsubseteq_s$. However, we might use a weaker notion of conformance that considers sequences of length at most k for some k. This would be relevant when the expected usage does not involve sequences of length greater than k since, for example, the system will be reset after at most k inputs. For example, in a communications protocol we might have that a 'disconnect' must happen after at most k steps. Systems that implement atomic transactions might also have this property: once a transaction has been completed the state of the system returns to its original value once one abstracts away any changes to, for example, a database that has been accessed. In addition, embedded systems are often designed to repeat a sequence of activities in a schedule, returning to the initial state at the end of such a sequence: we might use the bound defined by this (see, for example, [25]). It is also relevant where we want a test case of length at most k that distinguishes two FSMs or states. Finally, the tester might use such a notion and choose a suitable value of k based on a trade-off between cost and effectiveness or start with a small value of k and gradually increase it if necessary. In this section we define such an implementation relation and explore the problem of deciding whether two FSMs are related under this.

First, we introduce some notation. We let IO_k denote the set of global traces that have at most k inputs. In addition, for an FSM N we let $L_k(N) = L(N) \cap IO_k$ denote the set of global traces of N that have at most k inputs. We can now define our implementation relation.

Definition 7: Given FSMs N and M with the same input and output alphabets, we say that N strongly k-conforms to M if for all $\sigma \in L_k(N)$ there exists some $\sigma' \in L(M)$ such that $\sigma' \sim \sigma$. If this is the case then we write $N \sqsubseteq_s^k M$.

Clearly, given N and M it is decidable whether $N \sqsubseteq_s^k M$: we can simply generate every element of $L_k(N)$ and for each $\sigma \in L_k(N)$ we determine whether $\sigma \in \mathcal{L}(M)$. The following shows that this can be achieved in polynomial time if we have a bound on the number of ports.

Lemma 3: Given a sequence $\sigma \in IO_k$ and FSM M with n ports, we can decide whether $\sigma \in \mathcal{L}(M)$ in time of $O(|\sigma|^n)$.

Proof: The membership problem for a sequence σ and rational trace language with alphabet Σ and independence relation $\mathcal{I}$ can be solved in time of $O(|\sigma|^\alpha)$ where α is the size of the largest clique in the independence graph [35]. Since each observation is made at exactly one port and two observations are independent if and only if they are at different ports, we have that the maximal cliques of the independence graph all have size n and so $\alpha = n$. The result therefore follows. ■

Theorem 7: If there are bounds on the value of k and the number n of ports then the following problem can be solved in polynomial time: given FSMs N and M with at most n ports, do we have that $N \sqsubseteq_s^k M$?

Proof: First observe that $L_k(N)$ has $O(q^k)$ elements, where q denotes the maximum number of transitions leaving a state of N. Thus, since k is bounded, the elements in $L_k(N)$ can be produced in polynomial time. It only remains to consider each σ in $L_k(N)$ and decide whether it is in $\mathcal{L}(M)$. However, by Lemma 3, this can be decided in polynomial time. The result therefore follows. ■

Thus, if there are bounds on the number of ports and the length of sequences considered then we can decide $N \sqsubseteq_s^k M$ in polynomial time. However, the proof of Theorem 7 introduced terms that are exponential in n and k. It transpires that the problem is NP-hard without such bounds even for DFSMs. The proof uses the following.

Definition 8: 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.

[8]Directed graph G is strongly connected if for any two vertices v and v' it is possible to find a path from v to v'.

The three-in-one SAT problem is: does there exist an assignment to the boolean variables such that each C_j contains exactly one true literal.

The three-in-one SAT problem is known to be NP-complete [36].

The construction of the FSM M_1 in the following proof is similar to one from a proof in [16].

Theorem 8: The following problem is NP-hard: given k and completely specified DFSMs N and M, do we have that $N \sqsubseteq_s^k M$?

Proof: 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 $C_1, \ldots, C_q$. Define a DFSM M_1 with $r+q+2$ ports, inputs $z_{-1}, z_0, z_1, \ldots, z_r$ at ports $-1, 0, 1, \ldots, r$ and outputs $y_1, \ldots, y_{r+q}$ at ports $1, \ldots, r+q$. We count ports from -1 since the roles of inputs at -1 and 0 will be different from the roles of the other inputs.

DFSM M_1 has states s_0, s_1, s_2, s_3 in which s_0 is the initial state. The states represent different 'modes' and we now describe the roles of s_1 and s_2. In state s_1 an input at port p will lead to output at all ports corresponding to clauses with literal z_p. In state s_2 an input at port p will lead to output at all of the ports corresponding to clauses with literal $\neg z_p$. The input z_0 moves M_1 from s_1 to s_2. The special input z_{-1} takes M_1 from state s_0 to state s_1.

Overall, input z_0 does not produce output and only changes the state of M_1 if it is in s_1, in which case it takes M to s_2. Input z_{-1} does not produce output and only changes the state of M_1 if it is in state s_0, in which case it takes M_1 to state s_1.

For input z_p, $1 \leq p \leq r$, there are four transitions:

1) From s_1 there is a transition that, for all $1 \leq j \leq k$, sends y_{r+j} to $r+j$ if C_j contains z_p and otherwise sends no output to $r+j$. The transition sends no output to ports $-1, \ldots, r$ and does not change state.
2) From s_2 there is a transition that, for all $1 \leq j \leq k$, sends y_{r+j} to $r+j$ if C_j contains $\neg z_p$ and otherwise sends no output to $r+j$. The transition sends no output to ports $-1, \ldots, r$ and does not change state.
3) From s_0 there is a transition to state s_3 that produces no output.
4) From s_3 there is a transition to state s_3 that produces no output.

Now consider the global trace σ that starts with input sequence $z_{-1}z_0z_1 \ldots z_{r-1}$ and then has input z_r producing the outputs $y_{r+1} \ldots y_{r+q}$; all outputs are produced in response to the last input. Clearly we do not have $\sigma \in L(M_1)$. We now prove that $\sigma \in \mathcal{L}(M_1)$ if and only if there is a solution to the instance of the three-in-one SAT problem. Consider the problem of deciding whether there exists $\sigma' \in L(M_1)$ such that $\sigma' \sim \sigma$. Clearly the first input in σ' must be z_{-1}. Each input z_p is received once by the DFSM and these can be received in any order after z_{-1}. Thus, for all $1 \leq p \leq r$ we do not know whether z_p will be received before or after z_0 in σ'. If z_p is received before z_0 then an output is sent to all ports that correspond to clauses that contain literal z_p. If z_p is received after z_0 then an output is sent to all ports that correspond to clauses that contain literal $\neg z_p$. Thus there exists $\sigma' \in L(M_1)$ such that $\sigma' \sim \sigma$ if and only if there exists an assignment to the boolean variables $z_1, \ldots, z_r$ such that each C_j contains exactly one true literal.

We now construct DFSMs N and M such that $N \sqsubseteq_s^k M$ if and only if $\sigma \in \mathcal{L}(M_1)$. In the following we assume that $r > 1$ and let σ_1 be the global trace formed from σ by replacing the prefix $z_{-1}z_0z_1$ by $z_1z_{-1}z_0$. Thus, $\sigma_1 \sim \sigma$. We form N from M_1 by adding a new path that has label σ_1. We add state s'_3 such that the input of z_1 in state s_0 leads to state s'_3 (instead of s_3) and no output. From s'_3 we add a transition with label z_0 to another new state s'_4. We repeat this process, adding new states, until we have a path from s_0 with label $z_1z_0z_{-1}z_2z_3 \ldots z_{r-1}$ ending in state s'_{r+3}. We then add a transition from s'_{r+3} to s'_{r+4} with input z_r and the outputs $y_{r+1}, \ldots, y_{r+q}$. Finally, we complete N by adding a transition to s_3 with input z_p and null output from a state s'_j if there is no transition from s'_j with input z_p. Clearly, $L(N) = L(M_1) \cup \text{pref}(\sigma_1)I^*$. Let σ'_1 be defined such that $\sigma_1 = z_1\sigma'_1$. We can similarly form an FSM M from M_1 such that $L(M) = L(M_1) \cup \text{pref}(\{z_1\}I\{\sigma'_1\})I^*$. Since each I_p contains only one input we have that $\{z_1\}I\{\sigma'_1\}I^*$ and $\{\sigma_1\}I^+$ define the same sets of equivalence classes under $\sim$. Thus, the equivalence classes of $\text{pref}(\sigma_1)I^*$ and $\text{pref}(\{z_1\}I\{\sigma'_1\})I^*$ under $\sim$ differ only in the one that contains σ_1 and we know that $\sigma_1 \sim \sigma$. We therefore have that $N \sqsubseteq_s^k M$, for $k > r+1$, if and only if $\sigma \in \mathcal{L}(M_1)$ and we know that this is the case if and only if the instance of the three-in-one SAT problem has a solution. The result follows from the three-in-one SAT problem being NP-hard. ■

The results suggest that it is likely to be feasible

to decide $N \sqsubseteq_s^k M$ if there are only a few ports and k is not large. For some classes of system, such as communications protocols, we have only two ports. Thus, it likely to be feasible to determine conformance and generate test cases to distinguish states and FSMs for such systems where the bound on sequence length is not too large.

VI. Conclusions

There are important classes of systems such as cloud systems, communications protocols, web services and wireless sensor networks, that interact with their environment at physically distributed ports. In testing such a system we place a local tester at each port and the local tester (or user) at port p only observes the events that occur at p. It is known that this reduced observational power, under which a set of local traces is observed, can introduce additional controllability and observability problems.

This paper investigated testing from a finite state machine (FSM) model M that is the specification for a system that interacts with its environment at physically distributed ports. We considered implementation relation $\sqsubseteq_s$ that requires the set of local traces observed to be consistent with a global trace of the specification. We proved that it is undecidable whether $N \sqsubseteq_s M$ even if there are only two ports and gave conditions under which this is decidable. We also showed that a form of model checking is undecidable. The results also apply to labelled transition systems and input output transition systems.

There are several additional ramifications for testing. One such consequence is that it is undecidable whether there is a test case that is *capable* of distinguishing two states of an FSM or two FSMs. This complements results that show that it is undecidable whether there is a test case that is *guaranteed* to distinguish between two states or two FSMs [23]. While these results appear to be related the proofs used different approaches: the earlier result used results from multi-player games while this paper used results regarding multi-tape automata. Many methods for generating tests from a single-port FSM use sequences that either distinguish FSMs or distinguish states. The former is relevant when we have a fault domain $\mathcal{F}$ that contains a finite set of FSMs that model possible behaviours of the SUT: when testing from a single-port FSM M we know that it is possible to produce a test suite that distinguishes all elements of $\mathcal{F}$, that do not conform to M, from M. The results in this paper suggest that it will not be possible to extend such fault-based techniques to distributed testing. In addition, many techniques for generating tests from a single-port FSM M use sequences that distinguish states of M and, again, it seems unlikely that we will be able to extend such techniques to distributed testing. However, the paper also gave a number of conditions under which conformance is decidable.

Since it is undecidable whether $N \sqsubseteq_s M$ we defined a weaker implementation relation $\sqsubseteq_s^k$ that only considers input sequences of length k or less. This is particularly relevant in situations in which it is known that input sequences of length greater than k need not be considered since, for example, the system must be reset before this limit has been reached. The tester might also either choose a value for k based on an analysis of the cost/benefit trade-off or potentially start with a small value for k and increase it if no problems are found. We proved that if we place a bound on k and the number of ports then we can decide whether $N \sqsubseteq_s^k M$ in polynomial time but otherwise this problem is NP-hard.

There are several avenues for future work. First, there is the problem of finding weaker conditions under which we can decide $N \sqsubseteq_s M$. There is also the problem of extending the results to situation in which we can make additional observations such as refusals, where we observe the system not being able to accept an input. The results regarding $\sqsubseteq_s^k$ suggest that it will be feasible to determine conformance, and generate test cases to distinguish states and FSMs, for some classes of systems. For example, in communications protocols normally there are only two ports: deciding $\sqsubseteq_s^k$ is likely to be feasible if there is a (not too large) bound on the length of sequences before a disconnect must occur. It would be interesting to investigate the practical boundaries for such systems through significant case studies. Finally, while we have proved that deciding $\sqsubseteq_s^k$ is NP-hard it is still open whether it is in NP.

Acknowledgements

I would like to thank the anonymous referees whose comments led to significant improvements in this paper including a number of the proofs becoming shorter and more elegant.

REFERENCES

[1] J. T. C. ISO/IEC JTC 1, *Int. Standard ISO/IEC 9646-1. Info. Tech. - Open Sys. Interconn. - Conformance testing methodology and framework - Part 1: General concepts*. 1994.

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

[3] E. G. Cartaxo, P. D. L. Machado, and F. G. O. Neto, "On the use of a similarity function for test case selection in the context of model-based testing," *Software Testing, Verification and Reliability*, vol. 21, no. 2, pp. 75–100, 2011.

[4] A. Cavalcanti and M.-C. Gaudel, "Testing for refinement in Circus," *Acta Informatica*, vol. 48, no. 2, pp. 97–147, 2011.

[5] W. Grieskamp, "Multi-paradigmatic model-based testing," in *Formal Approaches to Software Testing and Runtime Verification (FATES/RV 2006)*, ser. Lecture Notes in Computer Science, vol. 4262. Springer, 2006, pp. 1–19.

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

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

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

[9] M. C. Gaudel, "Testing can be formal too," in *6th International Joint Conference CAAP/FASE Theory and Practice of Software Development (TAPSOFT'95)*, ser. Lecture Notes in Computer Science, vol. 915. Springer, 1995, pp. 82–96.

[10] 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.

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

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

[13] H. Ural and C. Williams, "Constructing checking sequences for distributed testing," *Formal Aspects of Computing*, vol. 18, no. 1, pp. 84–101, 2006.

[14] 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 Int. Conf. on Testing of Software and Communicating Systems (TestCom/FATES)*, ser. Lecture Notes in Computer Science, vol. 5047. Springer, 2008, pp. 169–183.

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

[16] R. M. Hierons, "Oracles for distributed testing," *IEEE Transactions on Software Engineering*, vol. to appear.

[17] R. Alur, K. Etessami, and M. Yannakakis, "Realizability and verification of MSC graphs," *Theoretical Computer Science*, vol. 331, no. 1, pp. 97–114, 2005.

[18] 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.

[19] 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," *IEEE Transactions on Communications*, vol. 39, no. 11, pp. 1604–1615, 1991.

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

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

[22] H. Ural, X. Wu, and F. Zhang, "On minimizing the lengths of checking sequences," *IEEE Transactions on Computers*, vol. 46, no. 1, pp. 93–99, 1997.

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

[24] G. Luo, A. Petrenko, and G. v. Bochmann, "Selecting test sequences for partially-specified nondeterministic finite state machines," in *The 7th IFIP Workshop on Protocol Test Systems*. Tokyo: Chapman and Hall, November 8–10 1994, pp. 95–110.

[25] F. Werner and D. Faragó, "Correctness of sensor network applications by software bounded model checking," in *FMICS 2010*, ser. Lecture Notes in Computer Science, vol. 6371. Springer, 2010, pp. 115–131.

[26] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Transactions on Software Engineering*, vol. 37, no. 5, pp. 649–678, 2011.

[27] M. O. Rabin and D. Scott, "Finite automata and their decision problems," *IBM Journal of Research and Development*, vol. 3, no. 2, pp. 114–125, 1959.

[28] B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun, "Infinite-state high-level MSCs: Model-checking and realizability," *Comp. and Sys. Science*, vol. 72, no. 4, pp. 617–647, 2006.

[29] A. Bouajjani, A. Muscholl, and T. Touili, "Permutation rewriting and algorithmic verification," in *16th Symp. on Logic in Computer Science (LICS'01)*, 2001.

[30] R. M. Hierons, "Checking experiments for stream X-machines," *Theoretical Comp. Sci.*, vol. 411, no. 37, pp. 3372–3385, 2010.

[31] V. Diekert and Y. Métivier, "Partial commutation and traces," in *Handbook of formal languages, vol. 3*, 1997, pp. 457–533.

[32] K. Culik II and J. Karhumäki, "HDTOL matching of computations of multitape automata," *Acta Informatica*, vol. 27, no. 2, pp. 179–191, 1989.

[33] T. Harju and J. Karhumäki, "The equivalence problem of multitape finite automata," *Theor. Comput. Sci.*, vol. 78, no. 2, pp. 347–355, 1991.

[34] M. Clerbout and M. Latteux, "Semi-commutations," *Information and Computation*, vol. 73, no. 1, pp. 59–74, 1987.

[35] A. Bertoni, G. Mauri, and N. Sabadini, "Equivalence and membership problems for regular trace languages," in *9th Coll. on Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, vol. 140. Springer, 1982, pp. 61–71.

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

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.