The effect of the distributed test architecture on the power of testing

R. M. Hierons[1] and H. Ural[2]

[1]School of Information Systems, Computing, and Mathematics, Brunel University, Uxbridge, Middlesex, UB8 3PH.
[2]School of Information Technology and Engineering, Faculty of Engineering, University of Ottawa, 800 King Edward Avenue, Ottawa, Ontario, K1N 6N5, Canada

There has been much interest in testing from finite state machines. If the system under test can be modeled by the (minimal) FSM N then testing from a (minimal) finite state machine (FSM) M is testing to check that N is isomorphic to M. In the distributed test architecture, there are multiple interfaces/ports and there is a tester at each port. This can introduce controllability/synchronization and observability problems. This paper shows that the restriction to test sequences that do not cause controllability problems and the inability to observe the global behaviour in the distributed test architecture, and thus relying only on the local behaviour at remote testers, introduces fundamental limitations into testing. There exist minimal FSMs that are not equivalent, and so are not isomorphic, and yet cannot be distinguished by testing in this architecture without introducing controllability problems. Similarly, an FSM may have non-equivalent states that cannot be distinguished in the distributed test architecture without causing controllability problems: these are said to be locally s-equivalent and otherwise they are locally s-distinguishable. This paper introduces the notion of two states or FSMs being locally s-equivalent and formalizes the power of testing in the distributed test architecture in terms of local s-equivalence. It introduces a polynomial time algorithm that, given an FSM M, determines which states of M are locally s-equivalent and produces minimal length input sequences that locally s-distinguish states that are not locally s-equivalent. An FSM is locally s-minimal if it has no pair of locally s-equivalent states. This paper gives an algorithm that takes an FSM M and returns a locally s-minimal FSM M' that is locally s-equivalent to M.

Keywords: testing; finite state machine; distributed test architecture; equivalence.

Received month date, year; revised month date, year; accepted month date, year

1. INTRODUCTION

The finite state machine (FSM) model of deterministic Mealy machines has been widely used to specify the behaviour of systems in diverse areas such as sequential circuits, lexical analysis, pattern matching, machine learning, and telecommunications as well as to describe the control structure of concurrent and distributed systems specified using a language such as SDL, Estelle or Statecharts (see, for example, [1, 2, 3, 4, 5]). Given an FSM M, an input sequence $\bar{x}$ *distinguishes* between states s_1 and s_2 if it leads to different output sequences when applied in these states. States s_1 and s_2 of M are said to be *equivalent* if no input sequence distinguishes between them. An FSM M is *minimal* if no two states of M are equivalent. Every FSM can be converted into an equivalent minimal FSM.

A particular example for the use of the FSM model is the specification of the control structures of distributed systems that are event-driven systems whose functionality is well-characterized by state-based models in terms of sequences of interactions between the system and its environment [6, 7]. Such a system has a number of interfaces, called *ports*, that are remote to each other. Programs or users at these ports are sources of inputs and destinations of outputs of the system.

When testing a distributed system in order to ensure its correct functionality, a distributed test architecture is needed where a tester is placed at each port of the system under test (SUT) and a test sequence (a sequence of input/output pairs constructed from the FSM model of the required behaviour of the SUT) is applied. During the application of a test sequence within a distributed test architecture, the existence of multiple remote testers and the absence of a global clock brings out the possibility of coordination problems among testers known as *controllability* (also known as *synchronization*) and *observability* problems.

A controllability problem occurs if a tester cannot determine when to send a particular input to the SUT. Let us suppose, for example, that there are two ports U and L, the first input x_1 is at port U and is expected to lead to output y_1 at port U only and the second input x_2 is at L. Since the tester at L does not observe either the input or output at U it does not know when this has occurred and so cannot determine when to send input x_2. An observability problem occurs when a tester cannot determine whether a particular output from the SUT is generated in response to a specific input. Let us suppose, for example, that the first input x_1 is to be applied at port U, this is expected to lead to output y_1 at L only, this is to be followed by input x_2 at L, which should lead to output y_2 at U. If, instead, the input of x_1 leads to output y_1 at L and y_2 at U and the input of x_2 leads to no output then the remote testers at each port observe the behaviour they were expecting: the tester at U sees input x_1 and then output y_2 while the tester at L sees output y_1 and then input x_2. Thus, the difference between the expected and observed behaviours are not observed in testing: fault masking has occurred. There has been much interest in the problem of testing in order to avoid controllability and observability problems (see, for example, [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]).

This paper shows that, in the distributed test architecture, it is necessary to use a different notion of what it means for an input sequence to distinguish between two states. Where an input sequence $\bar{x}$ distinguishes states s_1 and s_2 in this test architecture without causing a controllability problem then $\bar{x}$ is said to *locally s-distinguish* s_1 and s_2. This requires both that the input sequence $\bar{x}$ can be applied in states s_1 and s_2 without causing a controllability problem but also that if $\bar{x}$ is applied in states s_1 and s_2 then a different sequences of inputs and outputs is observed at one or more ports. Where states, and machines, cannot be locally s-distinguished they are said to be *locally s-equivalent* and testing in the distributed test architecture without introducing controllability problems is testing for local s-equivalence. This paper proves that local s-equivalence is weaker than classical FSM equivalence. While it has previously been observed that the distributed test architecture can lead to faults being missed in testing, this is the first paper to characterize the power of testing in this architecture.

The notion of local s-equivalence leads to a natural definition of minimality: an FSM is *locally s-minimal* if it does not contain locally s-equivalent states. If an FSM is minimal in the classical sense it is said to be *globally minimal*. We show, in this paper, that an FSM can be globally minimal but not locally s-minimal.

Testing from a globally minimal (deterministic and completely specified) FSM M is testing to decide whether the SUT is isomorphic to M. This is not the case in the distributed test architecture since it is only possible to test for local s-equivalence. Thus, the SUT may be locally s-equivalent to M, and thus indistinguishable from M in the distributed test architecture, without being isomorphic to M. This shows that there is a fundamental limitation to testing within the distributed test architecture: if we wish to avoid controllability problems then rather than test to check that the SUT is globally equivalent, and thus isomorphic, to M testing can at most check that the SUT is locally s-equivalent to M.

While many notions of equivalence, such as bisimulation, failure equivalence, and trace equivalence, have been explored within the literature on testing from labelled transition systems (see, for example [23, 24]), all of these collapse to isomorphism when testing from globally minimal, initially connected, completely specified, deterministic FSMs. This contrasts with the notion of local s-equivalence which is strictly weaker than isomorphism in such cases.

This paper makes the following contributions. It introduces the notion of two states of an FSM being locally s-distinguishable or locally s-equivalent and shows that in the distributed test architecture testing can distinguish between two states or FSMs, without introducing a controllability problem, if and only if they are locally s-distinguishable. Testing in the distributed test architecture without causing controllability problems is thus testing for local s-equivalence rather than isomorphism. It is proved that if an FSM M has n states and m ports and s_1 and s_2 are states of M then s_1 and s_2 are locally s-distinguishable by an input sequence starting with input at a given port p_i if and only if they are locally s-distinguished by an input sequence of length at most $m(n-1)$ that starts with input at p_i. We also prove that this is a tight bound. The paper gives a polynomial time algorithm that takes an FSM M and produces minimal length input sequences that locally s-distinguish the locally s-distinguishable states of M. As a consequence, this algorithm decides whether there is such a sequence in polynomial time. It can also decide whether two FSMs can be distinguished within this test architecture since we can apply the algorithm to the disjoint union of these FSMs. A second polynomial time algorithm takes an FSM M and generates a locally s-minimal FSM M' that is locally s-equivalent to M.

The results of this paper could have several practical ramifications. First, many algorithms for generating test sequences in the distributed test architecture have been motivated by the objective of deciding whether the SUT is equivalent to the FSM M, in the presence of a fault domain[3], without causing controllability problems. Such test generation algorithms typically place many restrictions on the FSM M and this means that they are not generally applicable: these restrictions can be seen as conditions under which equivalence and

[3] A fault domain is a set of models such that the tester believes that the SUT is behaviourally equivalent to an (unknown) element of this set [25].

local s-equivalence coincide for the given fault domain. However, since we now know that testing can only distinguish FSMs if they are not locally s-equivalent, the challenge is to produce test generation algorithms that are motivated by local s-equivalence and not equivalence and thus that do not place restrictions on M. Second, by capturing the power of testing in the distributed test architecture the consequences of using this test architecture should be clearer. This could help a tester to decide whether to use this architecture, which can be relatively easy and inexpensive to implement, or to add an external network through which messages can be sent between the remote testers in order to overcome the controllability and observability problems (see, for example, [9, 16, 18]). Finally, if the use of the SUT should correspond to input sequences that do not cause controllability problems and in use we only observe behaviour locally then we may be happy to produce a locally s-minimal FSM M' from M and this may lead to a smaller model and thus a more compact SUT.

This paper is structured as follows. Section 2 describes the distributed test architecture, multi-port FSMs, and controllability and observability problems introduced by the use of the distributed test architecture. Section 3 states what it means to globally distinguish two states and shows that a sequence $\bar{x}$ that globally distinguishes two states s_1 and s_2 of an FSM need not help distinguish these states in the distributed test architecture. It then introduces the notion of an input sequence locally s-distinguishing two states of an FSM. Section 4 places an upper bound on the length of a minimal length input sequence that locally s-distinguishes two states and shows that this bound is tight. Section 5 gives a polynomial time algorithm that produces input sequences that locally s-distinguish states of an FSM and Section 6 gives a polynomial time algorithm that takes an FSM and transforms it into a locally s-minimal FSM. Finally, Section 7 draws conclusions.

2. PRELIMINARIES

2.1. Multi-port FSMs

A multi-port FSM has $m > 1$ interfaces/ports at which it interacts with its environment as shown in Figure 1. The set of ports will be denoted $P = \{p_1, \ldots, p_m\}$. All examples use two ports denoted L and U. A (completely-specified and deterministic) multi-port FSM is defined by a tuple $(S, X, Y, \delta, \lambda, s_0)$ in which:

- S is the finite *set of states* of M;
- $s_0 \in S$ is the *initial state* of M;
- $X = X_1 \cup \ldots \cup X_m$ is the finite *input alphabet* of M, where for $1 \leq i \leq m$, X_i is the input alphabet at port p_i and for all $1 \leq i < j \leq m$ we have that $X_i \cap X_j = \emptyset$;
- $Y = (Y_1 \cup \{-\}) \times \ldots \times (Y_m \cup \{-\})$ is the *output alphabet* of M, where for $1 \leq i \leq m$, Y_i is the output alphabet at port p_i, $-$ denotes no output, and for all $1 \leq i < j \leq m$ we have that $Y_i \cap Y_j = \emptyset$;
- δ is the *transition function* of type $S \times X \rightarrow S$; and
- λ is the *output function* of type $S \times X \rightarrow Y$.

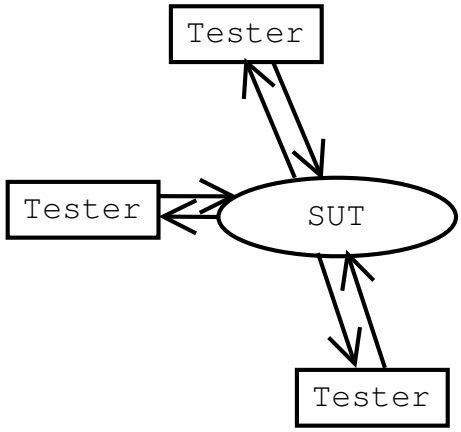

FIGURE 1. An SUT with multiple-ports

This paper only considers completely-specified and deterministic FSMs. Note that while a transition can send output to multiple ports it can receive input from one port only. There is thus an implicit assumption that inputs are processed individually but it seems likely that the results in this paper can be extended to the case where multiple inputs can trigger a transition. Naturally, the inputs may be buffered to be processed when the system is ready or the SUT might process inputs sufficiently quickly so that buffering is not required (the 'slow environment' assumption). However, we can restrict testing so that an input is only sent once the previous input has been processed.

A variable that represents an element of Y will have a 'hat' above its name in order to distinguish it from an element of some Y_i, an example being $\hat{y}$. Normally the elements of $\hat{y}$ will be denoted $y_1, \ldots, y_m$ and so $\hat{y} = (y_1, \ldots, y_m)$.

This paper deals with multi-port FSMs and thus a multi-port FSM will simply be called an FSM and will be denoted by M. An FSM in which there is only one port will be called a *single-port FSM*. A variable name will have a bar over it (for example, $\bar{x}$) if this variable represents a sequence and ϵ will denote the empty sequence. A sequence or word will be expressed by the listing of its elements. For example abc will denote the sequence with three elements: a then b and then c. Given a sequence $\bar{a} = a_1 \ldots a_k$, a sequence $\bar{z}$ is a *prefix* of $\bar{a}$ if $\bar{z} = a_1 \ldots a_j$ for some $1 \leq j \leq k$. Further, $\bar{z}$ is a *suffix* of $\bar{a}$ if $\bar{z} = a_j \ldots a_k$ for some $1 \leq j \leq k$.

A transition of an FSM M is a triple $(s_j, s_k, x/\hat{y})$, where $s_j, s_k \in S$, $x \in X$, and $\hat{y} \in Y$ such that $\delta(s_j, x) = s_k$, $\lambda(s_j, x) = \hat{y}$ and the *input/output pair* $x/\hat{y}$ is the *label* of the transition. A transition $(s_j, s_k, x/\hat{y})$ is a *self-loop* if $s_j = s_k$.

A sequence of consecutive transitions $\bar{\rho} = t_1 \dots t_k$ of M, $t_i = (s_i, s_{i+1}, x_i/\hat{y}_i)$, is said to be a path. The path $\bar{\rho}$ has *label* $\bar{z} = x_1/\hat{y}_1 \dots x_k/\hat{y}_k$ and $\bar{z}$ has *input portion* $x_1 \dots x_k$.

It is possible to extend δ and λ to input sequences in the following way.

$$\begin{aligned} \delta(s, \epsilon) &= s \\ \delta(s, x\bar{x}) &= \delta(\delta(s, x), \bar{x}) \end{aligned}$$

$$\begin{aligned} \lambda(s, \epsilon) &= \epsilon \\ \lambda(s, x\bar{x}) &= \lambda(s, x)\lambda(\delta(s, x), \bar{x}) \end{aligned}$$

An FSM M is *initially connected* if every state s of M can be reached from the initial state using some input sequence: for every state $s \in S$ there exists an input sequence $\bar{x}_s$ such that $\delta(s_0, \bar{x}_s) = s$. If M is not initially connected then the unreachable states can be removed if we require an initially connected FSM. An FSM M is *globally minimal*[4] if none of its states are *globally equivalent*[5] (i.e., for all $s_i, s_j \in S$, $s_i \neq s_j$, there exists an input sequence $\bar{x} \in X^*$ such that $\lambda(s_i, \bar{x}) \neq \lambda(s_j, \bar{x})$). An FSM M is said to be *completely specified* if, for each input $x \in X$, there is a transition with input x defined from each state of M.

Given two FSMs $M_1 = (S_1, X, Y, \delta_1, \lambda_1, s_{01})$ and $M_2 = (S_2, X, Y, \delta_2, \lambda_2, s_{02})$ with $S_1 \cap S_2 = \emptyset$ we can define the FSM $M_1 \oplus M_2$ produced by taking their disjoint union. More formally, $M_1 \oplus M_2$ is the FSM $(S_1 \cup S_2, X, Y, \delta, \lambda, s_{00})$ in which for all $x \in X$ and $s \in S_1 \cup S_2$ we have that: if $s \in S_1$ then $\lambda(s, x) = \lambda_1(s, x)$ and $\delta(s, x) = \delta_1(s, x)$ and otherwise $\lambda(s, x) = \lambda_2(s, x)$ and $\delta(s, x) = \delta_2(s, x)$. Naturally, the initial state s_{00} of $M_1 \oplus M_2$ is one of s_{01} and s_{02} but the use of $M_1 \oplus M_2$ in this paper will make this choice irrelevant. $M_1 \oplus M_2$ will be used in order to be able to apply results, regarding distinguishing states, to reason about distinguishing machines. Thus, since $M_1 \oplus M_2$ is not initially connected, in this paper we *do not* assume that any FSM considered is initially connected.

Given state s and input sequence $\bar{x}$, $\gamma(s, \bar{x})$ will denote the input/output sequence resulting from applying $\bar{x}$ when M is in state s. This may be recursively defined in the following manner.

$$\begin{aligned} \gamma(s, \epsilon) &= \epsilon \\ \gamma(s, x\bar{x}) &= (x/\lambda(s, x))\gamma(\delta(s, x), \bar{x}) \end{aligned}$$

In this paper we assume that in testing we can wait for all output to be produced in response to an input before the next input is applied. This is the case where communication is synchronous or the time taken to receive the output in response to an input is negligible (the *slow environment* assumption). It has also been shown how this can be achieved in the case that the testers can directly communicate and there are bounds on the time taken for messages transmitted [16].

[4]Normally such an FSM is said to be minimal. In this paper the phrase globally minimal is used in order to distinguish this from the notion of locally s-minimal defined in Section 3.

[5]The term globally equivalent is used, rather than equivalent, in order to distinguish this from the term locally s-equivalent introduced in Section 3.

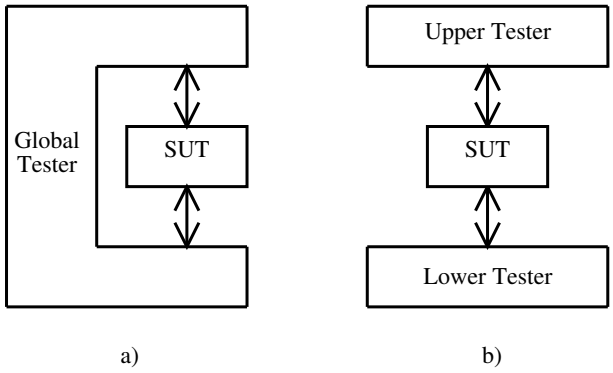

FIGURE 2. Two test architectures: a) local; b) distributed

2.2. The distributed test architecture

An FSM M defines the expected global behaviour of any potential implementation. Each expected global behaviour is expressed as the label of a sequence of transitions from M. An expected global behaviour is called a global test sequence.

Testing an SUT whose externally observable behaviour is defined by an FSM M can be carried out as a fault detection experiment in some specific test architecture. Two such architectures [26] are shown in Figure 2 for a two-port SUT. The two ports, called U and L, represent the upper interface and lower interface of the SUT respectively. The local architecture in Figure 2a) has a global tester that controls and observes both interfaces of the SUT during the application of a global test sequence constructed from the FSM specification of the SUT. Figure 2b) shows the distributed test architecture where the lower interface and the upper interface of the SUT are controlled and observed by the lower tester and by the upper tester, respectively. In a distributed test architecture involving multiple testers, each tester applies a local test sequence constructed from a global test sequence for the SUT and there is no global clock. In the local test sequence, a tester can't observe the inputs or outputs of the other testers.

In this architecture, two remote testers at ports U and L are required to coordinate the application of a preset test sequence. However, they cannot directly communicate with one another and there is no global clock. This requirement may lead to controllability and observability problems, in addition to those that stem from the black box nature of the SUT.

The controllability and observability problems are a

consequence of the tester at each port seeing only a portion of an input/output sequence: the parts that involved that port. This may be considered to be the *actual local behaviour*. The tester then compares this with the *expected local behaviour*. Let $\bar{z}$ be the label of a transition sequence of an FSM M. The sequence $\bar{z}$ is thus an expected global behaviour. Given $\bar{z}$, $\pi_i(\bar{z})$ will denote the expected local behaviour at p_i. The function π_i can be recursively defined in the following way in which $\bar{z}$ is an input/output sequence and x is an input.

$$\begin{aligned}
\pi_i(\epsilon) &= \epsilon \\
\pi_i((x/(y_1,\ldots,y_m))\bar{z}) &= \pi_i(\bar{z}) \text{ if } x \notin X_i \wedge y_i = - \\
\pi_i((x/(y_1,\ldots,y_m))\bar{z}) &= x\pi_i(\bar{z}) \text{ if } x \in X_i \wedge y_i = - \\
\pi_i((x/(y_1,\ldots,y_m))\bar{z}) &= y_i\pi_i(\bar{z}) \text{ if } x \notin X_i \wedge y_i \neq - \\
\pi_i((x/(y_1,\ldots,y_m))\bar{z}) &= xy_i\pi_i(\bar{z}) \text{ if } x \in X_i \wedge y_i \neq -
\end{aligned}$$

2.3. The controllability problem

Let us suppose that the distributed test architecture is being used and the intention is to input x_1x_2 when M is in state s, x_1 is input at port $P \in \{U, L\}$, and x_2 is input at port $Q \in \{U, L\}$. If $P \neq Q$ and, 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 introduces a controllability (synchronization) problem. A controllability (synchronization) problem exists when a tester is required to send an input to the SUT in the current transition, and because it is not involved in the previous transition, i.e., it didn't send the input or receive an output in the previous transition, it does not know when to send the input. Where a sequence of transitions does not have this problem it is said to be synchronized.

DEFINITION 2.1. *A sequence of two transitions* t_1t_2, $t_1 = (s_1, s_2, x/(y_1,\ldots,y_m))$, $t_2 = (s_2, s_3, x'/(y'_1,\ldots,y'_m))$, *in which* x' *is input at port* $p_i \in P$ *is* synchronized *if* t_1 *either has input from port* p_i *or sends output to* p_i ($\pi_i(t_1) \neq \epsilon$).

DEFINITION 2.2. *A sequence* $t_1 \ldots t_k$ *of transitions starting at* s_1 *is* synchronized *if for all* $1 \leq i < k$, t_it_{i+1} *is synchronized. If* $\bar{z}$ *is the label of* $t_1 \ldots t_k$ *and this has input portion* $\bar{x}$ *then* $\bar{z}$ *and* $\bar{x}$ *are said to be* synchronized from s_1 *or simply* synchronized *if* s_1 *is clear.*

When working within the distributed test architecture, if a sequence of transitions in M is not synchronized then the corresponding test sequence cannot be applied without causing a controllability problem. However, in general there may be no such test sequence that satisfies a given test objective such as executing a particular transition [19]. In common with other work in this area, in this paper we only consider synchronized test sequences. This paper investigates the impact of the distributed test architecture on the effectiveness of testing. If this impact is too severe then it is often possible to introduce an external network through which the testers can communicate and use this to overcome controllability and observability problems (see, for example, [9, 16, 18]).

2.4. The observability problem

In the distributed test architecture each tester sees only the behaviour at a single port. This section discusses some consequences of this more limited ability to observe the behaviour.

Let us suppose that the distributed test architecture is being used and x_1x_2 is to be input when M is in state s, $x_1, x_2 \in X_U$. Suppose further that the input of x_1 is expected to trigger output (y_U, y_L) and the input of x_2 is expected to trigger output $(y'_U, -)$. Then $x_1y_Ux_2y'_U$ should be observed at U and y_L should be observed at L. These are still the observations if $(y_U, -)$ is produced in response to x_1 and (y'_U, y_L) is produced in response to x_2. Thus, since only local sequences of interactions are observed, it is possible for one output fault[6] to mask another. Each of these transitions might lead to incorrect output in use if it is executed within a *different* sequence. By contrast, output faults are always observed when testing a single-port FSM or using the local test architecture. Note that if x_2 had been from X_L, this combination of faults would have been detected, in the distributed test architecture, by the tester at port L since this would have observed x_2y_L rather than y_Lx_2.

The faults described above mask one another because the correct value y_L is observed at L and, while it is produced in response to the wrong transition, the tester at port L does not know when to stop waiting for y_L. A similar situation would have occurred if output at L is expected in response to x_2 but not x_1 and instead it was produced in response to x_1. These situations correspond to the notion of an undetectable output-shifting fault in which output can be seen as being shifted between (not necessarily adjacent) transitions in a sequence (see, for example, [22]).

Note that when such fault masking between two transitions t and t' occurs in the given sequence $t\bar{t}t'$, the faults may be detectable if one or more of the transitions t and t' are used in a different sequence. Where this is the case, and we wish to test t and t', we need a test sequence that contains t and t' in contexts in which such fault masking cannot occur.

Undetectable output-shifting faults describe situations in which the distributed test architecture allows output faults to mask one another. This masking is a consequence of the structure of the test sequence used: such faults might not be masked in other sequences and thus may be observed by the user. However, the focus of this paper is finding input sequences that can be used

[6] An output fault is a fault in which a transition produces the wrong output.

to distinguish states or machines in the distributed test architecture and thus observability problems are only a concern when they lead to an input sequence failing to achieve this.

3. GLOBALLY DISTINGUISHING AND LOCALLY S-DISTINGUISHING STATES

In order to distinguish two states in the distributed test architecture, it is necessary to apply an input sequence that leads to different sequences of local observations at some port. This section defines what it means for an input sequence to achieve this. Section 4 proves that the length of such sequences is bounded above by $m(n-1)$ for an FSM with n states and m ports: for every pair of states s and s' and port p_i if s and s' can be distinguished in the distributed test architecture by a synchronized input sequence starting at p_i then they can be distinguished in the distributed test architecture by a synchronized input sequence of length at most $m(n-1)$ that starts at p_i. It also proves that this bound is tight. Section 5 gives a polynomial time algorithm that determines whether such sequences exist and generates them when they do. This algorithm may thus be used to decide, in polynomial time, whether such sequences exist for a pair of states and thus which states of an FSM are locally s-equivalent.

3.1. Testing with a global tester

If there is a global tester, an input sequence $\bar{x}$ distinguishes two states if the input of $\bar{x}$ leads to different output sequences when applied in these states.

Definition 3.1. *Input sequence* $\bar{x}$ globally distinguishes *states* s_1 *and* s_2 *of* M *if* $\lambda(s_1, \bar{x}) \neq \lambda(s_2, \bar{x})$.

This corresponds to the classical notion of distinguishing states of a single-port FSM M.

3.2. Testing in the distributed test architecture

Consider the FSM M_0 given in Figure 3 in which $x_1, x_2 \in X_U$, $a_1, a_2 \in Y_U$, and $b \in Y_L$. The input sequence x_1x_2 globally distinguishes states s_1 and s_2. However, neither tester observes a difference: the expected local behaviour at L is b and the expected local behaviour at U is $x_1a_1x_2a_2$. In the distributed test architecture, this input sequence does not help distinguish between s_1 and s_2; instead it is necessary to observe a different sequence of input and output values at one of the ports. Thus, it is not sufficient that an input sequence globally distinguishes two states: a stronger property is required if we wish to distinguish between them. Note that in the example the prefix x_1 of x_1x_2 does allow us to distinguish the states.

If input sequence $\bar{x}$ is applied when M is in state s_1 the sequence $\pi_i(\gamma(s_1, \bar{x}))$ is observed at port p_i. This leads to the following definition.

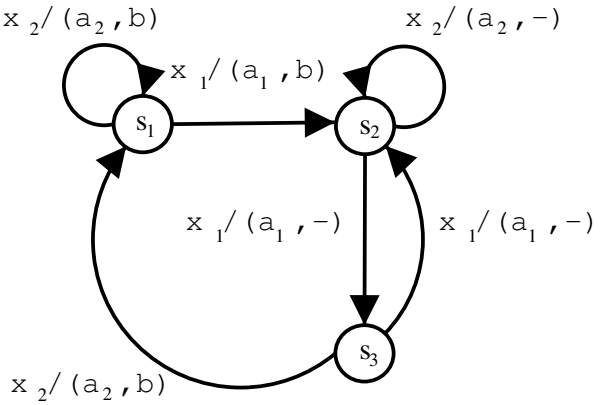

FIGURE 3. The FSM M_0

Definition 3.2. *Input sequence* $\bar{x}$ locally s-distinguishes *states* s_1 *and* s_2 *of* M at port $p_i \in P$ *if* $\bar{x}$ *is synchronized from* s_1 *and* s_2 *and* $\pi_i(\gamma(s_1, \bar{x})) \neq \pi_i(\gamma(s_2, \bar{x}))$. *Further,* $\bar{x}$ locally s-distinguishes *states* s_1 *and* s_2 *of* M *if there exists a port* $p_i \in P$ *such that* $\bar{x}$ *locally s-distinguishes* s_1 *and* s_2 *at* p_i.

These definitions can be extended to locally s-distinguishing FSMs.

Definition 3.3. *Synchronized input sequence* $\bar{x}$ locally s-distinguishes *FSMs* $M_1 = (S_1, X, Y, \delta_1, \lambda_1, s_{01})$ *and* $M_2 = (S_2, X, Y, \delta_2, \lambda_2, s_{02})$ at port $p_i \in P$ *if* $\bar{x}$ *locally s-distinguishes states* s_{01} *and* s_{02} *of* $M_1 \oplus M_2$ *at port* p_i. *Synchronized input sequence* $\bar{x}$ locally s-distinguishes M_1 *and* M_2 *if there exists a port* $p_i \in P$ *such that* $\bar{x}$ *locally s-distinguishes* M_1 *and* M_2 *at* p_i.

Proposition 3.1. *If* $\bar{x} \in X^*$ *locally s-distinguishes states* s_1 *and* s_2 *of* M *then* $\bar{x}$ *globally distinguishes* s_1 *and* s_2. *However, an input sequence may globally distinguish two states* s_1 *and* s_2 *of* M *but not locally s-distinguish them.*

Proof

The first part of the result follows immediately from the definitions.

Consider the FSM M_0 shown in Figure 3. Recall that $x_1, x_2 \in X_U$, $a_1, a_2 \in Y_U$, and $b \in Y_L$. The input of x_1x_2 from s_1 should lead to the output sequence $(a_1, b)(a_2, -)$ and the input of x_1x_2 from s_2 should lead to the output sequence $(a_1, -)(a_2, b)$. It is clear that the input of x_1x_2 is synchronized from s_1 and s_2.

Input sequence x_1x_2 globally distinguishes states s_1 and s_2 since it leads to different input/output sequences. However, in each case $x_1a_1x_2a_2$ is observed at U and b is observed at L. Thus, x_1x_2 does not locally s-distinguish s_1 and s_2 as required. □

Proposition 3.2. *Given states* s_1 *and* s_2 *of an FSM* M *and input sequence* $\bar{x}$, *it is possible to decide whether* $\bar{x}$ *locally s-distinguishes* s_1 *and* s_2 *in* $O(|\bar{x}|)$ *time.*

Proof

Observe that given state s_1 and input sequence $\bar{x}$, it is possible to determine $\pi_i(\gamma(s_1, \bar{x}))$ in $O(|\bar{x}|)$. The result now follows from the fact that it is sufficient to find $\pi_i(\gamma(s_1, \bar{x}))$ and $\pi_i(\gamma(s_2, \bar{x}))$ for each $p_i \in P$ and then apply syntactic comparisons. □

If an input sequence $\bar{x}$ is synchronized from states s_1 and s_2 and globally distinguishes s_1 and s_2 then s_1 and s_2 must be locally s-distinguishable.

PROPOSITION 3.3. *Let us suppose that $\bar{x}$ is synchronized from states s_1 and s_2 of M and globally distinguishes s_1 and s_2. If $\bar{x}_1$ is a minimal prefix of $\bar{x}$ that globally distinguishes s_1 and s_2 then $\bar{x}_1$ locally s-distinguishes s_1 and s_2.*

Proof

Let $\bar{x}_1 = \bar{x}'_1 x_2$ for some $x_2 \in X$, $\bar{x}'_1 \in X^*$. By the minimality of $\bar{x}_1$, $\lambda(s_1, \bar{x}'_1) = \lambda(s_2, \bar{x}'_1)$ and thus $\pi_i(\gamma(s_1, \bar{x}'_1)) = \pi_i(\gamma(s_2, \bar{x}'_1))$ for all $p_i \in P$. It is now sufficient to observe that since $\bar{x}_1$ globally distinguishes between s_1 and s_2, $\lambda(\delta(s_1, \bar{x}'_1), x_2)$ and $\lambda(\delta(s_2, \bar{x}'_1), x_2)$ must differ at some port p_i and thus that $\pi_i(\gamma(s_1, \bar{x}_1)) \neq \pi_i(\gamma(s_2, \bar{x}_1))$. □

This helps illustrate an important property of testing in the distributed test architecture: test sequence effectiveness is not monotonic in that in some cases we can take an input sequence $\bar{x}$ that locally s-distinguishes two machines M_1 and M_2 and extend it to an input sequence $\bar{x}\bar{x}'$ that does not locally s-distinguish M_1 and M_2. An example of this is given by the FSM M_0 shown in Figure 3: we can take M_0 and the FSM formed by making s_2 the initial state. We have already seen that $x_1 x_2$ globally distinguishes these states but does not locally s-distinguish them and it is straightforward to see that x_1 does locally s-distinguish them. This property, of test sequence effectiveness not being monotonic, is not a problem when applying a single sequence $\bar{x}$ that is intended to locally s-distinguish two states or FSMs if we can terminate testing after applying $\bar{x}$. However, it is likely to have practical consequences when generating sequences to locally s-distinguish states with the intention of embedding these sequences within a larger test sequence: here we will want to ensure that the effectiveness of a sequence is not affected by the sequences that precede it and follow it.

If the distributed test architecture is used and we wish to avoid controllability problems then in order to distinguish between two states it is necessary to locally s-distinguish these states. This leads to a new definition of equivalence.

DEFINITION 3.4. *Two states s_1 and s_2 of M are* locally s-equivalent *if no input sequence locally s-distinguishes s_1 and s_2. This is denoted $s_1 \approx s_2$ and otherwise $s_1 \not\approx s_2$.*

It is now possible to say what it means for M to be minimal within this architecture.

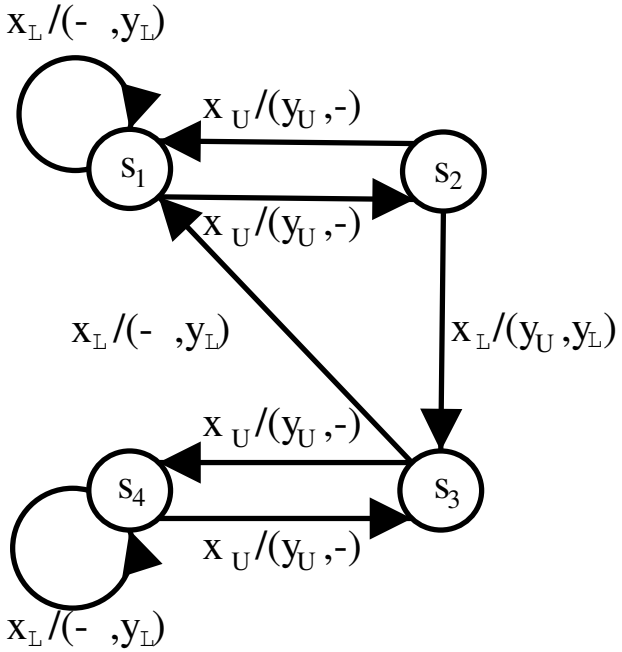

FIGURE 4. The FSM M_1

DEFINITION 3.5. *M is* locally s-minimal *if for every pair (s_1, s_2) of states of M ($s_1 \neq s_2$) we have that s_1 and s_2 are locally s-distinguishable.*

PROPOSITION 3.4. *A globally minimal FSM need not be locally s-minimal.*

Proof

Consider the FSM M_1 shown in Figure 4 in which there is one input x_U at U, one input x_L at L, one output y_U at U and one output y_L at L. It is clear that M_1 is globally minimal. However, any synchronized input sequence from s_1 or s_4 must be in either $\{x_U\}^*$ or $\{x_L\}^*$. By observing the corresponding output sequences, it is clear that s_1 and s_4 are not locally s-distinguishable at either U or L. Thus, M_1 is not locally s-minimal. □

If two states are locally s-equivalent then it is not possible to distinguish between them when testing in the distributed architecture without causing a controllability problem. The SUT may only be distinguished from an FSM M, by testing in the distributed test architecture without encountering controllability problems, if the initial states of the SUT and M are not locally s-equivalent. Thus, testing involves checking a weaker notion of correctness than that used when there is a global tester. For example, the FSM M_2 shown in Figure 5 is locally s-equivalent to M_1 but is not globally equivalent to M_1. When there is a global tester, testing can distinguish between M_1 and M_2. By contrast, in the distributed test architecture, a synchronized input sequence cannot distinguish between M_1 and M_2 since their initial states

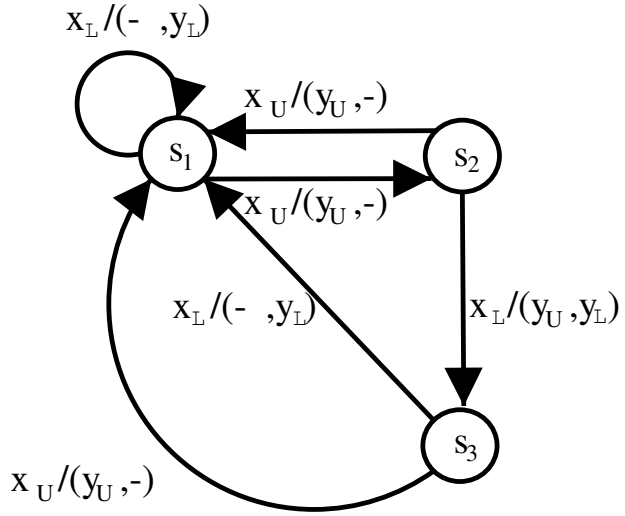

FIGURE 5. The FSM M_2

are locally s-equivalent. This is despite the fact that M_1 and M_2 are globally minimal and are not isomorphic.

In order to locally s-distinguish FSMs M and M' using an input sequence $\bar{x}$ we need to be able to apply $\bar{x}$ to both M and M' and a different sequence must be observed at one or more ports. We can thus see that two FSMs M and M' can be distinguished by a synchronized input sequence when testing within the distributed test architecture if and only if their initial states are locally s-distinguishable and so that this notion captures the power of testing in the distributed test architecture without causing controllability problems.

4. THE LENGTH OF SEQUENCES THAT LOCALLY S-DISTINGUISH STATES

This section contains a proof that if two states of an FSM M with n states and m ports are locally s-distinguished by an input sequence starting with input at a given port p_i then they are locally s-distinguished by an input sequence of length at most $m(n-1)$ that starts with input at p_i. It also proves that the bound is tight. Finally, we show that the result is not significantly altered if we allow the input sequence to start at any port. Section 5 gives a polynomial time algorithm that generates such sequences.

The following result shows that if an input sequence $\bar{x}$ of length greater than one locally s-distinguishes two states then a shorter input sequence locally s-distinguishes two states.

Lemma 4.1. *If $\bar{x}$ locally s-distinguishes states s_1 and s_2 of M at port $p_i \in P$ and $\bar{x} = \bar{x}_1\bar{x}_2$ then either*

1. *$\bar{x}_1$ locally s-distinguishes states s_1 and s_2 at p_i; or*
2. *$\bar{x}_2$ locally s-distinguishes states $\delta(s_1, \bar{x}_1)$ and $\delta(s_2, \bar{x}_1)$ at p_i.*

Proof

Proof by contradiction: let us suppose that $\bar{x}$ locally s-distinguishes states s_1 and s_2 at p_i but the above conditions do not hold. Since the first condition does not hold, $\pi_i(\gamma(s_1, \bar{x}_1)) = \pi_i(\gamma(s_2, \bar{x}_1))$. Since the second condition does not hold, $\pi_i(\gamma(\delta(s_1, \bar{x}_1), \bar{x}_2)) = \pi_i(\gamma(\delta(s_2, \bar{x}_1), \bar{x}_2))$. But $\pi_i(\gamma(s_j, \bar{x})) = \pi_i(\gamma(s_j, \bar{x}_1))\pi_i(\gamma(\delta(s_j, \bar{x}_1), \bar{x}_2))$ ($j \in \{1, 2\}$), providing a contradiction as required. □

The following result will be used in reasoning about the length of sequences that locally s-distinguish states.

Lemma 4.2. *Let us suppose that $\bar{x}_1\bar{x}_2$ is synchronized from states s_1 and s_2 of M, $\bar{x}_1$ does not locally s-distinguish s_1 and s_2 and $\bar{x}_2$ locally s-distinguishes $\delta(s_1, \bar{x}_1)$ and $\delta(s_2, \bar{x}_1)$ at port $p_i \in P$. Then $\bar{x}_1\bar{x}_2$ locally s-distinguishes s_1 and s_2 at port p_i.*

Proof

Since $\bar{x}_1$ does not locally s-distinguish s_1 and s_2, $\pi_i(\gamma(s_1, \bar{x}_1)) = \pi_i(\gamma(s_2, \bar{x}_1))$. Since $\bar{x}_2$ locally s-distinguishes $\delta(s_1, \bar{x}_1)$ and $\delta(s_2, \bar{x}_1)$ at p_i, $\pi_i(\gamma(\delta(s_1, \bar{x}_1), \bar{x}_2)) \neq \pi_i(\gamma(\delta(s_2, \bar{x}_1), \bar{x}_2))$. The result now follows from observing that since $\pi_i(\gamma(s_j, \bar{x})) = \pi_i(\gamma(s_j, \bar{x}_1))\pi_i(\gamma(\delta(s_j, \bar{x}_1), \bar{x}_2))$ ($j \in \{1, 2\}$), $\pi_i(\gamma(s_1, \bar{x})) \neq \pi_i(\gamma(s_2, \bar{x}))$. □

The following definition is inspired by Gill [27].

Definition 4.1. *States s_1, s_2 of M are* (k, i)-equivalent *if no input sequence of length k or less, that starts with input at port $p_i \in P$, locally s-distinguishes s_1 and s_2; otherwise they are* (k, i)-separable. *This defines a relation $\approx_k^i$: $s_1 \approx_k^i s_2$ if and only if s_1 and s_2 are (k, i)-equivalent. Further, s_1, s_2 are* locally s-equivalent at p_i *if no input sequence, that starts with input at port p_i, locally s-distinguishes s_1 and s_2. This is denoted $s_1 \approx^i s_2$.*

Proposition 4.1. *$s_1 \approx s_2$ if and only if for all $p_i \in P$ we have that $s_1 \approx^i s_2$.*

Let us suppose that an input sequence $\bar{x}$ is synchronized from one state s_1 of an FSM M but not from another state s_2 of M. The failure of $\bar{x}$ to be synchronized from s_2 must be due to some difference between the input/output sequences $\gamma(s_1, \bar{x})$ and $\gamma(s_2, \bar{x})$ and if we find the first such difference then the corresponding input sequence locally s-distinguishes s_1 and s_2.

Proposition 4.2. *For an FSM M and states s_1 and s_2 of M, if input sequence $\bar{x}$ is synchronized from s_1 but not from s_2 then there is a prefix of $\bar{x}$ that locally s-distinguishes s_1 and s_2.*

Proof

Proof by contradiction: let us suppose that $\bar{x} = x_1 \dots x_l$ is synchronized from s_1 but not from s_2 but no prefix of $\bar{x}$ locally s-distinguishes s_1 and s_2. Let $\bar{x}_1 = x_1 \dots x_k$ denote the longest prefix of $\bar{x}$ that is synchronized from both s_1 and s_2 and so $k \geq 1$ and

$\bar{x}_1 x_{k+1}$ is a prefix of $\bar{x}$. Let $\bar{x}_2 = x_1 \ldots x_{k-1}$ and so if $k = 1$ then $\bar{x}_2$ is the empty sequence.

Since $\bar{x}_1$ is synchronized from s_1 and s_2 but no prefix of $\bar{x}$ locally s-distinguishes s_1 and s_2, for all $p_i \in P$ we have that $\pi_i(\gamma(s_1, \bar{x}_1)) = \pi_i(\gamma(s_2, \bar{x}_1))$ and $\pi_i(\gamma(s_1, \bar{x}_2)) = \pi_i(\gamma(s_2, \bar{x}_2))$. Thus, since $\bar{x}_1 = \bar{x}_2 x_k$ and $\bar{x}_2$ does not locally distinguish s_1 and s_2, for all $p_i \in P$ we must have that $\pi_i(\gamma(\delta(s_1, \bar{x}_2), x_k)) = \pi_i(\gamma(\delta(s_2, \bar{x}_2), x_k))$.

Since $\bar{x}_1 x_{k+1}$ is synchronized from s_1 we must have that $\pi_i(\gamma(\delta(s_1, \bar{x}_2), x_k))$ involves the port p_j at which x_{k+1} is input and so $\pi_i(\gamma(\delta(s_2, \bar{x}_2), x_k))$ also involves port p_j. Thus, $\bar{x}_1 x_{k+1}$ is synchronized from s_2. This provides a contradiction as required. □

LEMMA 4.3. *The relation $\approx_k^i$ is an equivalence relation.*

Proof

It is clear that $\approx_k^i$ is symmetric and reflexive and it thus suffices to prove that $\approx_k^i$ is transitive. Proof by contradiction: let us suppose that there exists s_1, s_2, s_3 such that $s_1 \approx_k^i s_2$, $s_2 \approx_k^i s_3$ but $s_1 \not\approx_k^i s_3$. Thus there is an input sequence $\bar{x}$ of length at most k that starts with input at p_i, is synchronized from s_1 and s_3, and that locally s-distinguishes s_1 and s_3. Since $s_1 \approx_k^i s_2$, by Proposition 4.2 we know that $\bar{x}$ is synchronized from s_2 and does not locally s-distinguish s_1 and s_2. Similarly, since $s_2 \approx_k^i s_3$, $\bar{x}$ is synchronized from s_2 and does not locally s-distinguish s_2 and s_3. This provides a contradiction as required. □

Since $\approx_k^i$ is an equivalence relation it defines a set $\mathcal{P}_k^i$ of equivalence classes that partition the set of states. Specifically, two states s and s' are in the same set A from $\mathcal{P}_k^i$ if $s \approx_k^i s'$.

PROPOSITION 4.3. *$s_1 \approx^i s_2$ if and only if for all $k \geq 0$, $s_1 \approx_k^i s_2$.*

THEOREM 4.1. *The relation $\approx$ is an equivalence relation.*

This follows from Lemma 4.3 and Propositions 4.1 and 4.3. □

Since $\approx$ is an equivalence relation, it has a set of equivalence classes. It will transpire that these equivalence classes can be used to transform M into a locally s-minimal FSM.

The proofs of the following results use a similar logic to the proof that, given a single-port (globally) minimal FSM M with n states, there is an input sequence of length at most $n-1$ that (globally) distinguishes any two states s_1 and s_2 of M [27]. The results demonstrate that for all $s_1, s_2 \in S$, if s_1 and s_2 are locally s-distinguishable then they are locally s-distinguished by an input sequence of length at most $m(n-1)$. The proof will operate by considering the equivalence relations of the form $\approx_k^i$, using the fact that each can have at most n equivalence classes and that for all $k_1 > k_2$, $\approx_{k_1}^i$ has at least as many equivalence classes as $\approx_{k_2}^i$.

LEMMA 4.4. *Let us suppose that states s_1 and s_2 of M are locally s-distinguishable at $p_i \in P$ and they are (k, i)-equivalent. Then there exists a port $p_j \in P$ and states s_1', s_2' of M that are (k, j)-equivalent but $(k+1, j)$-separable.*

Proof

Let $\bar{x}$ denote a shortest input sequence that starts with input at p_i and locally s-distinguishes s_1 and s_2. Then $|\bar{x}| > k$. Let $\bar{x}'$ denote the prefix of $\bar{x}$ of length $|\bar{x}| - k - 1$ and thus $\bar{x} = \bar{x}'\bar{x}''$ for some $\bar{x}''$ with $|\bar{x}''| = k + 1$. Let p_j denote the port with the property that the first element of $\bar{x}''$ is from X_j. By the minimality of $\bar{x}$, $\bar{x}'$ does not locally s-distinguish s_1 and s_2.

Consider the states $s_l' = \delta(s_l, \bar{x}')$ $(1 \leq l \leq 2)$. Since $\bar{x}'\bar{x}''$ locally s-distinguishes s_1 and s_2, and $\bar{x}'$ does not locally s-distinguish s_1 and s_2, by Lemma 4.1 $\bar{x}''$ must locally s-distinguish s_1' and s_2'. By the minimality of $\bar{x}$ and Lemma 4.2, s_1' and s_2' are locally s-distinguished by no shorter input sequence that starts with input from X_j. But $|\bar{x}''| = k + 1$ and thus s_1' and s_2' are $(k+1, j)$-separable. As s_1' and s_2' are locally s-distinguished by no shorter sequence, that starts with input from X_j, s_1' and s_2' are (k, j)-equivalent, as required. □

THEOREM 4.2. *Let M denote an FSM with n states and m ports. Given states s_1, s_2 of M and port $p_i \in P$, if s_1 and s_2 are locally s-distinguished by an input sequence starting with an element of X_i then they are locally s-distinguished by an input sequence of length at most $m(n-1)$ that starts with an element of X_i.*

Proof

Let $\bar{x}$ be a shortest input sequence, starting with an element of X_i, that locally s-distinguishes s_1 and s_2 and $|\bar{x}| = k$. Proof by contradiction: assume that $k > m(n-1)$. Then for all $0 \leq q < k$, s_1 and s_2 are (q, i)-equivalent.

By Lemma 4.4, there is a function f such that for all $0 \leq q < k$, there is port $p_{f(q)} \in P$ and states s_1^q and s_2^q that are $(q, f(q))$-equivalent but $(q+1, f(q))$-separable. If for some $0 \leq q_1 < q_2 < k$ we have that $f(q_1) = f(q_2) = j$ then since there are states of M that are (q_1, j)-equivalent but $(q_1 + 1, j)$-separable, $\approx_{q_2}^j$ has more equivalence classes than $\approx_{q_1}^j$.

Since $k > m(n-1)$, there is a port $p_j \in P$ that appears more than $n-1$ times in the sequence $p_{f(0)} \ldots p_{p(k-1)}$. Let i denote the smallest non-negative integer such that $f(i) = j$. It is now sufficient to note that each $\approx_q^j$ has at most n equivalence classes and each also has at least one equivalence class. There can be at most $n-1$ (proper) refinements of $\approx_i^j$. This provides a contradiction as required. □

In order to show that this bound is tight we will define an FSM M_{mn} with n states and m ports such that in order to locally s-distinguish states s_0 and s_1 we must apply an input sequence that passes through each state and in order to change state we must apply input at

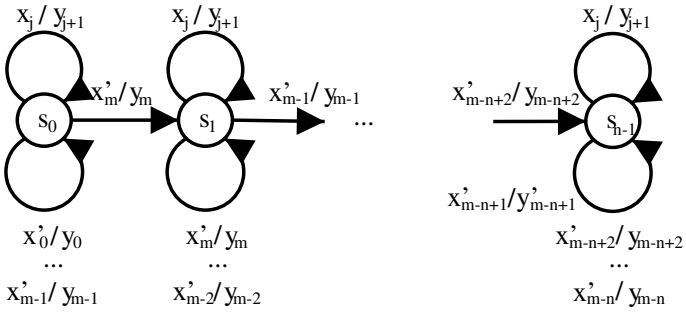

FIGURE 6. The FSM M_{mn}

every port. Throughout the rest of this section, where integer arithmetic is applied in a subscript of a port it is modulo m and integer arithmetic in subscripts of states is modulo n. Since it is normal to start the range at 0 in modular arithmetic, throughout the rest of this section the states are denoted $s_0, \ldots, s_{n-1}$ and the ports are denoted $p_0, \ldots, p_{m-1}$. The FSM M_{mn} is illustrated in Figure 6.

Definition 4.2. *Given $n > 1$ and $m > 1$ the FSM $M_{mn} = (S, X, Y, \delta, \lambda, s_0)$ is defined by the following in which $P = \{p_0, \ldots, p_{m-1}\}$ and $S = \{s_0, \ldots, s_{n-1}\}$. We have two inputs at each port and so for all $p_j \in P$ we set $X_j = \{x_j, x'_j\}$.*

For $0 \leq i \leq n-1$ and $0 \leq j \leq m-1$, input x_j in state s_i leads to no change in state and only output $y_{j+1} \in Y_{j+1}$. Thus, the (self loop) transition from s_i with input x_j can be followed by input at either p_j or p_{j+1}. Thus, transitions with input x_j cannot change the state but allow the next input to be at a different port (p_{j+1}).

Now consider the transitions with input of the form x'_j. All of these are self-loops with the following exceptions: for all $0 \leq i < n-1$ the transition from state s_i with input x'_{m-i} changes the state to s_{i+1}. Thus, all transitions in s_{n-1} are self-loops and a transition can only change the state from s_i ($0 \leq i < n-1$) if the input is x'_{m-i}. For the output there are two cases:

1. *The input of x'_{m-n+1} in state s_{n-1} leads only to output $y'_{m-n+1} \in Y_{m-n+1}$ with $y'_{m-n+1} \neq y_{m-n+1}$.*
2. *If $i \neq n-1$ or $j \neq m-n+1$ then the transition from s_i with input x'_j leads only to output y_j.*

Observe that the only way in which we can choose states s and s' and input x such that $\lambda(s,x) \neq \lambda(s',x)$ is to have exactly one of s and s' being s_{n-1} and x being x'_{m-n+1}.

We now prove some results regarding M_{mn}.

Lemma 4.5. *Let $\bar{x}$ denote an input sequence that is synchronized from states s_0 and s_1 of M_{mn} and $\bar{x} = \bar{x}'x$ for some $x \in X$. If $\bar{x}$ locally s-distinguishes states s_0 and s_1 of M_{mn} and no proper prefix of $\bar{x}$ locally s-distinguishes states s_0 and s_1 then $x = x'_{m-n+1}$ and either $\delta(s_0, \bar{x}') = s_{n-1}$ or $\delta(s_1, \bar{x}') = s_{n-1}$.*

Proof

This result follows immediately from the fact that the only opportunity for producing a different output from two states in response to an input is if one of the states is s_{n-1} and the input is x'_{m-n+1}.□

Lemma 4.6. *There is an input sequence of length $m(n-1)$ starting with input at port p_0 that locally s-distinguishes state s_0 and s_1 of M_{mn}.*

Proof

In order to locally s-distinguish the states it is sufficient to apply an input sequence that takes *exactly* one of these states to s_{n-1} and then applies input x'_{m-n+1}. We will build such an input sequence from $n-1$ subsequences $\bar{x}_1, \ldots, \bar{x}_{n-1}$. Each subsequence, when applied in the state s_j reached from s_1, will apply inputs of the form x_i a total of $m-1$ times in order to be able to apply the input x'_{m-j} to take the FSM to state s_{j+1}.

The first section is $\bar{x}_1 = x_0 x_1 \ldots x_{m-2} x'_{m-1}$. It is clear that this is synchronized from both s_0 and s_1. If $n = 2$ then this sequence locally s-distinguishes s_0 and s_1 as required and so we assume that $n > 2$. Thus, $\bar{x}_1$ leads to the same output sequences from s_0 and s_1, and in each case can be followed only by input at port p_{m-1}. In addition, $\delta(s_1, \bar{x}_1) = s_2$ and since $n > 1$ we have that either $\delta(s_0, \bar{x}_1) = s_0$ or $\delta(s_0, \bar{x}_1) = s_1$.

We now define the remaining subsequences in a similar manner, starting each with input at the port that received the final input in the previous subsequence. For $0 < j \leq n-1$, the subsequence $\bar{x}_j = x_{m-j+1} \ldots x_1 \ldots x_{m-j-1} x'_{m-j}$.

It is now sufficient to observe that $\bar{x} = \bar{x}_1 \ldots \bar{x}_{n-1}$ takes M_{mn} from state s_1 to s_{n-1} and applies x'_{m-n+1} and that it does not take M_{mn} from state s_0 to s_{n-1}. □

Lemma 4.7. *Let $\bar{x}x$ denote an input sequence that is synchronized from state s_j of M_{mn}, starts with input at p_{m-j+1} and ends with the input x'_{m-n+1}. If $\delta(s_j, \bar{x}) = s_{n-1}$ then $|\bar{x}x| \geq m(n-j)$.*

Proof

Proof by induction on $n-j$. The base case, where $j = n-1$, follows immediately. Inductive hypothesis: assume that the result holds for all $j > k$. Then it is sufficient to prove that the result holds for $j = k$.

In order to reach state s_{n-1} from state s_k we need to pass through state s_{k+1}. Let $\bar{x}_1$ denote the minimum length prefix of $\bar{x}$ such that $\delta(s_k, \bar{x}_1) = s_{k+1}$ and so $\bar{x} = \bar{x}_1 \bar{x}_2$ for some $\bar{x}_2$. Clearly $\bar{x}_1$ must end in input x'_{m-k}. However, by construction, for a sequence starting at s_k with input at p_{m-k+1} to end with input x'_{m-k} it must include $x_{m-k+1}, \ldots, x_{m-k-1}$ and so $|\bar{x}_1| \geq m$. By the inductive hypothesis, $|\bar{x}_2 x| \geq m(n-(k+1))$ and so the result follows. □

Theorem 4.3. *Given integers $n > 1$ and $m > 1$ there exists an FSM M with n states and m ports that has*

states s_0 and s_1 and port $p_j \in P$ such that s_0 and s_1 are locally s-distinguishable by an input sequence starting with input at p_j but cannot be locally s-distinguished by any input sequence of length less than $m(n-1)$ that starts with input at p_j.

Proof

It is sufficient to consider M_{mn}, states s_0 and s_1 and port p_0. By Lemma 4.6 it is possible to distinguish these states using an input sequence starting at p_0. By Lemma 4.5, the application of $\bar{x}$ from state s_1 must end in the application of x'_{m-n+1} in state s_{n-1}. The result thus follows from Lemma 4.7. □

The above result concerns the worst case when we have specified the port p at which the first input must be applied. The following shows that the result is not significantly different if we allow the sequence to start with input at any port.

THEOREM 4.4. *Given integers $n > 1$ and $m > 1$ there exists an FSM M with n states and m ports that has locally s-distinguishable states s_0 and s_1 that cannot be locally s-distinguished by any input sequence of length less than $m(n-2)+1$.*

Proof

If we consider states s_0 and s_1 of M_{mn} then the result follows from Lemma 4.7. □

5. GENERATING SEQUENCES THAT LOCALLY S-DISTINGUISH STATES

This section gives a polynomial time algorithm with the following property: if it is applied to FSM M then, given two states s_1 and s_2 of M and $p_i \in P$, if some input sequence starting with an element of X_i locally s-distinguishes s_1 and s_2 then the algorithm returns a minimal length input sequence that achieves this. Thus, as a side-effect, it decides which pairs of states of M can be locally s-distinguished by input sequences starting with an element of X_i. It may thus be used to determine which states of M are locally s-equivalent and so by considering $M_1 \oplus M_2$ it can also be used to decide whether FSMs M_1 and M_2 are locally s-equivalent.

Algorithm 1 is a variant on the classical algorithm for generating sequences that (globally) distinguish states of a single-port FSM. It works in the following manner. It builds a partition $\mathcal{P}_k^i$, $1 \leq i \leq m$, in which two states are in the same set of some $\mathcal{P}_k^i$ if and only if they are (k,i)-equivalent. It will be assumed that each set A in a partition $\mathcal{P}_k^i$ has a label that identifies it — this guarantees that A cannot be confused with a set from a partition $\mathcal{P}_k^j$, $j \neq i$. At each step, the partition $\mathcal{P}_{k+1}^i$ is built from the $\mathcal{P}_k^j$. Associated with each set $A \in \mathcal{P}_k^i$ is a set D_A of input sequences that start with elements of X_i and, between them, locally s-distinguish the states in A from those in each $A' \in \mathcal{P}_k^i$ with $A' \neq A$. Thus, in order to decide whether two states s_1 and s_2 are locally s-distinguished by an input sequence starting with an element of X_i it is sufficient to apply Algorithm 1 and determine whether s_1 and s_2 are in different sets in the final partition $\mathcal{P}_k^i$. If s_1 and s_2 are locally s-distinguished by an input sequence starting with an element of X_i and s_1 is in the set $A \in \mathcal{P}_k^i$ then D_A contains a minimum length input sequence that starts with an element of X_i and locally s-distinguishes s_1 and s_2.

ALGORITHM 1.
1. Input FSM M.
2. For $1 \leq i \leq m$ let $\mathcal{P}_0^i = \{S\}$.
3. Let $\mathcal{P}_1^i$ denote the partition of the states of M defined by $\approx_1^i$ $(1 \leq i \leq m)$.
4. For set $A \in \mathcal{P}_1^i$ let D_A denote a minimal subset of X_i such that: for all $s_1 \in A$ and $s_2 \in S \setminus A$, there is some $x \in D_A$ that locally s-distinguishes s_1 and s_2.
5. While there exists $1 \leq i \leq m$ such that $\mathcal{P}_k^i \neq \mathcal{P}_{k-1}^i$ do
6. Set $\mathcal{P}_{k+1}^i = \mathcal{P}_k^i$ $(1 \leq i \leq m)$.
7. While there is some $1 \leq i \leq m$ and $A \in \mathcal{P}_{k+1}^i$ such that at least one of the following holds:
   (a) there exists input $x \in X_i$ and states $s_1, s_2 \in A$ such that $\delta(s_1, x)$ and $\delta(s_2, x)$ are in different elements of $\mathcal{P}_k^i$ (choose two of these elements of $\mathcal{P}_k^i$ denoted B_1 and B_2); or
   (b) there exists input $x \in X_i$ and states $s_1, s_2 \in A$ such that the input of x in a state in A leads to an output at port $p_j \neq p_i$ and $\delta(s_1, x)$ and $\delta(s_2, x)$ are in different elements of $\mathcal{P}_k^j$ (choose two of these elements of $\mathcal{P}_k^j$ denoted B_1 and B_2).
8. Choose a minimal length input sequence $\bar{x}$ from D_{B_1} that locally s-distinguishes the elements of B_1 from those of B_2 and:
   (a) Partition A on the basis of $x\bar{x}$ forming new sets $A_1, \ldots, A_l$ and set $\mathcal{P}_{k+1}^i = (\mathcal{P}_{k+1}^i \setminus \{A\}) \cup \{A_1, \ldots, A_l\}$.
   (b) Set $D_{A_j} = D_A \cup \{x\bar{x}\}$ for all $1 \leq j \leq l$.
9. Endwhile
10. Set $k = k+1$
11. Endwhile
12. Output $\mathcal{P}_k^1, \ldots, \mathcal{P}_k^m$, and every D_A for $A \in \mathcal{P}_k^1 \cup \ldots \cup \mathcal{P}_k^m$.

FIGURE 7. Generating input sequences that locally s-distinguish states

Algorithm 1 is given in Figure 7. The following shows how Algorithm 1 works.

THEOREM 5.1. *States s_1 and s_2 of M are (k,i)-separable if and only if they are in different sets in the partition $\mathcal{P}_k^i$ produced in the application of Algorithm 1.*

Proof

Case 1: $\Rightarrow$. Proof by induction on k. The base cases,

in which $k = 0$ and $k = 1$, follow immediately. Inductive hypothesis: assume that this holds for all values of k less than a ($a > 1$).

Let us suppose that s_1 and s_2 are (a, i)-separable. From Lemma 4.4 it is clear that the algorithm cannot terminate with $\mathcal{P}_k^1 \dots \mathcal{P}_k^m$ for any $k < a$. It is sufficient to prove that s_1 and s_2 are in different sets from $\mathcal{P}_a^i$. Proof by contradiction: assume that s_1 and s_2 are in the same set from $\mathcal{P}_a^i$. If s_1 and s_2 are $(a-1, i)$-separable the inductive hypothesis gives a contradiction as required and so s_1 and s_2 must be $(a-1, i)$-equivalent. Let $x\bar{x}$ denote an input sequence of length a that locally s-distinguishes s_1 and s_2 ($x \in X_i$, $\bar{x} \in X^*$). Since x does not locally s-distinguish s_1 and s_2, by Lemma 4.1, $\bar{x}$ must locally s-distinguish states $s_1' = \delta(s_1, x)$ and $s_2' = \delta(s_2, x)$. Further, since $x\bar{x}$ is synchronized from s_1 and s_2, one of the following must be the case:

1. $\bar{x}$ starts with input from p_i. By the inductive hypothesis, s_1' and s_2' are in different sets in $\mathcal{P}_{a-1}^i$.
2. $\bar{x}$ starts with input from $p_j \neq p_i$ and $\lambda(s_1, x)$ and $\lambda(s_2, x)$ include output at p_j. By the inductive hypothesis, s_1' and s_2' are in different sets in $\mathcal{P}_{a-1}^j$.

In each case, in Algorithm 1, s_1 and s_2 will be locally s-distinguished in forming $\mathcal{P}_a^i$. This provides a contradiction and thus completes the inductive case.

Case 2: $\Leftarrow$. Proof by induction on k. The base cases, in which $k = 0$ and $k = 1$, follow immediately. Inductive hypothesis: assume that this holds for all values of k less than a ($a > 1$).

Let us suppose that s_1 and s_2 are in different sets from $\mathcal{P}_a^i$ and these sets are A_1 and A_2 respectively. It is sufficient to prove that s_1 and s_2 are (a, i)-separable. If s_1 and s_2 are in different sets for $\mathcal{P}_{a-1}^i$ the result follows from the inductive hypothesis and so assume that s_1 and s_2 are in the same set A of $\mathcal{P}_{a-1}^i$. Since s_1 and s_2 are in different sets from $\mathcal{P}_a^i$, there are two cases to consider.

1. There is an input x that takes s_1 and s_2 from A to different sets of $\mathcal{P}_{a-1}^i$. Let $s_l' = \delta(s_l, x)$ ($l \in \{1, 2\}$). By the inductive hypothesis, s_1' and s_2' are $(a-1, i)$-separable and so there is an input sequence $\bar{x}$ of length $a-1$ that locally s-distinguishes s_1' and s_2' such that $\bar{x}$ starts with input from X_i. Since $\bar{x}$ is synchronized from s_1' and s_2' and $\bar{x}$ starts with an input from X_i, $x\bar{x}$ is synchronized from s_1 and s_2. If $\lambda(s_1, x) \neq \lambda(s_2, x)$ then s_1 and s_2 are $(1, i)$-separable and the result follows. If $\lambda(s_1, x) = \lambda(s_2, x)$ then s_1 and s_2 are locally s-distinguished by $x\bar{x}$ and the result follows from observing that $x\bar{x}$ has length a.
2. There is an input x that takes s_1 and s_2 from A to different sets of $\mathcal{P}_{a-1}^j$, $j \neq i$, such that $\lambda(s_1, x)$ and $\lambda(s_2, x)$ have output at p_j. Let $s_l' = \delta(s_l, x)$ ($l \in \{1, 2\}$). By the inductive hypothesis, s_1' and s_2' are $(a-1, j)$-separable and so there is an input sequence $\bar{x}$ of length $a-1$ that locally s-distinguishes s_1' and s_2' such that $\bar{x}$ starts with input from X_j. $x\bar{x}$ is synchronized from s_1 and s_2. If $\lambda(s_1, x) \neq \lambda(s_2, x)$ then s_1 and s_2 are $(1, i)$-separable and the result follows. If $\lambda(s_1, x) = \lambda(s_2, x)$ then s_1 and s_2 are locally s-distinguished by $x\bar{x}$ and the result follows from observing that $x\bar{x}$ has length a.

The result thus follows. □

Theorem 5.2. *If there is an input sequence that locally s-distinguishes states s_1 and s_2 of M and starts with input at port $p_i \in P$ then Algorithm 1 produces a minimum length input sequence that locally s-distinguishes s_1 and s_2 and starts with input at p_i.*

Proof

Let us suppose that s_1 and s_2 are locally s-distinguished by an input sequence of length a but by no shorter input sequence. From Lemma 4.4 it is clear that the algorithm cannot terminate with $\mathcal{P}_k^1, \dots, \mathcal{P}_k^m$ for some $k < a$. Given a partition $\mathcal{P}_k^i$, $1 \leq i \leq m$, and set $A \in \mathcal{P}_k^i$, the input sequences in D_A have length at most k. The result thus follows from Theorem 5.1. □

Theorem 5.3. *Let us suppose that Algorithm 1, when applied to M, returns partitions $\mathcal{P}_k^1, \dots, \mathcal{P}_k^m$. Then states s_1 and s_2 of M are locally s-equivalent if and only if for all $1 \leq i \leq m$ there exists a set $A_i \in \mathcal{P}_k^i$ such that $s_1, s_2 \in A_i$.*

Proof

This follows from Theorem 5.1 and the fact that s_1 and s_2 are locally s-equivalent if and only if for all $1 \leq i \leq m$ they cannot be locally s-distinguished by an input sequence starting with input at p_i. □

From this it is clear that the intersections of the partitions produced by Algorithm 1 gives the equivalence classes of $\approx$. Section 6 gives an algorithm that, based on these equivalence classes, transforms M into a locally s-minimal FSM M'.

Algorithm 1 adapts an algorithm that produces input sequences to pairwise distinguish the states of a single-port FSM with n states and p inputs in $O(pn^2)$ time. It is clear that Algorithm 1 also takes time that is of $O(pn^2)$. An interesting question is whether Hopcroft's $O(np \log n)$ algorithm [28] can be adapted.

6. PRODUCING A LOCALLY S-MINIMAL FSM

This section gives an algorithm that takes the equivalence classes of $\approx$ for M produced by Algorithm 1 and returns a locally s-minimal FSM M'.

Algorithm 2 operates in the following way. Given two states s_1 and s_2 of M that are locally s-equivalent in M, s_1 and s_2 are merged. In order to merge two states s_1 and s_2 each transition of the form $(s, s_2, x/\hat{y})$ is rewritten to $(s, s_1, x/\hat{y})$ and then s_2 and all transitions leaving s_2 are deleted.

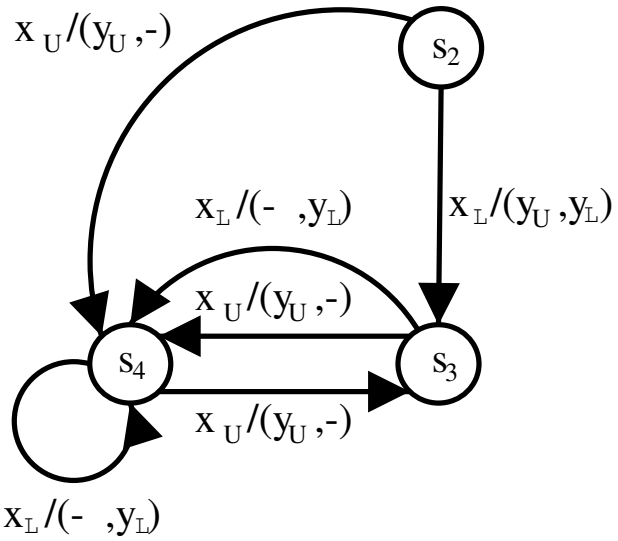

FIGURE 8. The FSM M_3

ALGORITHM 2.
1. Input M and the partition $\mathcal{P}$ of the state set S of M defined by $\approx$.
2. Let $M' = M$.
3. While there exists a set $A \in \mathcal{P}$ such that $|A| > 1$ do
4. Choose a set $A \in \mathcal{P}$ such that $|A| > 1$ and an ordered pair (s_1, s_2) of states with $s_1, s_2 \in A$ and $s_1 \neq s_2$.
5. If s_2 is the initial state of M' then make s_1 the initial state of M'.
6. Rewrite each transition of M' of the form $(s, s_2, x/\hat{y})$ to $(s, s_1, x/\hat{y})$.
7. Delete all transitions of M' that leave s_2 and then delete s_2 from M' and A.
8. endwhile
9. Output M'.

FIGURE 9. Generating a locally s-minimal FSM

Note that the order in which states are merged may affect the FSM produced. For example, the states s_1 and s_4 of FSM M_1 (Figure 4) may either be merged by deleting state s_4 to get M_2 (Figure 5) or may be merged by deleting state s_1 to get M_3 (Figure 8). Clearly M_2 and M_3 are not globally equivalent. However, the alternative locally s-minimal FSMs are locally s-equivalent (their initial states are locally s-equivalent) and thus are unique up to local s-equivalence.

LEMMA 6.1. *Let us suppose that states s_1 and s_2 of $M = (S, X, Y, \delta, \lambda, s_0)$ are locally s-equivalent and FSM $M' = (S', X, Y, \delta', \lambda', s'_0)$ is formed by taking a copy of M, in which the state names are primed, and rewriting every transition of the form $(s'_k, s'_2, x/\hat{y})$ to $(s'_k, s'_1, x/\hat{y})$. Then for every state $s_i \in S$, s_i and s'_i are locally s-equivalent.*

Proof

It is sufficient to prove that for all $\bar{x} \in X^*$ and $s_i \in S$, $\bar{x}$ does not locally s-distinguish s_i and s'_i. Proof will be by induction on the length of $\bar{x}$. The base case, in which $\bar{x} = \epsilon$, clearly holds. Inductive hypothesis: for every state s_i and input sequence $\bar{x}$ of length less than a ($a \geq 1$), $\bar{x}$ does not locally s-distinguish s_i and s'_i. Inductive case: let us suppose that $\bar{x}$ is an arbitrary input sequence of length a. It is sufficient to prove that for all $s_i \in S$, $\bar{x}$ does not locally s-distinguish s_i and s'_i.

There exists $x \in X$ and $\bar{x}_1 \in X^*$ such that $\bar{x} = x\bar{x}_1$. Consider a state $s_i \in S$. Clearly $\lambda(s_i, x) = \lambda'(s'_i, x)$. There are two cases.

1. Case 1: $\delta(s_i, x) \neq s_2$. Then $\delta(s_i, x) = s_j$ and $\delta'(s'_i, x) = s'_j$ for some $s_j \in S$. By the inductive hypothesis, $\bar{x}_1$ does not locally s-distinguish s_j and s'_j. Thus, since $\lambda(s_i, x) = \lambda'(s'_i, x)$, $\bar{x}$ does not locally s-distinguish s_i and s'_i.
2. Case 2: $\delta(s_i, x) = s_2$. Then $\delta(s_i, x) = s_2$ and $\delta'(s'_i, x) = s'_1$. By the inductive hypothesis, $\bar{x}_1$ does not locally s-distinguish s_1 and s'_1. Further, s_1 and s_2 are locally s-equivalent. Thus, $\bar{x}_1$ does not locally s-distinguish s_2 and s'_1. Thus, since $\lambda(s_i, x) = \lambda'(s'_i, x)$, $\bar{x}$ does not locally s-distinguish s_i and s'_i.

In each case $\bar{x}$ does not locally s-distinguish s_i and s'_i. The result thus follows. □

THEOREM 6.1. *If Algorithm 2 takes an FSM M and returns M' then M' is locally s-minimal and M and M' are locally s-equivalent.*

Proof

By Lemma 6.1, given states s'_1 and s'_2 of M' ($s'_1 \neq s'_2$), there are states s_1 and s_2 of M such that s_i and s'_i are locally s-equivalent ($i \in \{1, 2\}$) and s_1 and s_2 are locally s-distinguishable. Thus, s'_1 and s'_2 are locally s-distinguishable. Since, for every pair states s'_1 and s'_2 of M' with $s'_1 \neq s'_2$, s'_1 and s'_2 are locally s-distinguishable, M' is locally s-minimal. It thus remains to prove that M and M' are locally s-equivalent. This follows by observing that the process of generating M' from M proceeds through a sequence of transformations that, by Lemma 6.1, preserve local s-equivalence. □

THEOREM 6.2. *Algorithm 2 takes $O(pn^2)$ time, where p denotes the size of the input alphabet of the FSM M to which the algorithm is applied and n denotes the number of states of M.*

Proof

There are $O(n)$ iterations of the loop. Each iteration transforms each transition entering a particular state and there are at most pn such transitions. The result thus follows. □

7. CONCLUSIONS

When testing a distributed system under test (SUT), it may be necessary to have more than one tester: one at each port of the SUT. If these testers cannot directly communicate and there is no global clock then testing is taking place within the distributed test architecture. In this test architecture it is only possible to observe the sequences of interactions at each port locally and this can introduce controllability and observability problems.

When testing from a globally minimal (deterministic and completely specified) FSM M, if there is a global tester then testing is capable of distinguishing between M and any (minimal) SUT that is not isomorphic to M. This paper has shown that this is not the case in the distributed test architecture: it is possible to distinguish the SUT from M without introducing controllability problems if and only if the SUT is not locally s-equivalent to M.

We have defined a new form of equivalence called local s-equivalence that captures the power of testing within the distributed test architecture when we avoid controllability problems. In contrast to notions of equivalence that come from the literature on testing from labelled transition systems, this notion of equivalence is strictly weaker than isomorphism when testing from an FSM that is globally minimal, deterministic, initially connected, and completely specified.

This paper has proved that if an FSM M has n states and m ports, s_1 and s_2 are states of M, and p_i is a port then s_1 and s_2 are locally s-distinguishable using an input sequence starting at p_i if and only if they are locally s-distinguished by an input sequence starting at p_i that has length at most $m(n-1)$. It has also proved that this bound is tight. It has given a polynomial time algorithm that takes an FSM and returns a partition of the state set of M into sets of locally s-equivalent states. The algorithm also returns minimum length input sequences that locally s-distinguish the locally s-distinguishable states of M. The output of this algorithm is used in a second algorithm that transforms FSM M into a locally s-minimal FSM M'.

There may be cases where it is important that testing checks for global equivalence rather than local s-equivalence. Where this is the case, it is not sufficient to use a distributed test architecture, with no global clock, in which remote testers cannot directly communicate with one another. Instead one could consider introducing channels through which the local testers may send *coordination messages* to one another (see, for example, [9, 16, 18]).

The notion of local s-equivalence captures the power of testing in the distributed test architecture. There remain a number of open questions and challenges. The first challenge is to devise test generation algorithms that produce test sequences that, in the presence of a fault domain, determine whether the SUT is locally s-equivalent to the specification FSM M. By contrast, current algorithms aim to check equivalence and thus place very strong restrictions on M. These restrictions can be seen as conditions under which equivalence and local s-equivalence coincide for a given fault domain. Future research might also investigate conditions under which local s-equivalence and global equivalence are identical. Where for a specification FSM M there is a smaller locally s-minimal FSM M', if the distributed test architecture represents actual usage then we may be able to implement the smaller model M' rather than M. Where it is important to test for global equivalence, there is the question of how this can be achieved in a manner that minimizes either the number of channels required between testers or the number of coordination messages used. There may also be scope for using test sequences that introduce controllability problems but still achieve a given test objective. Finally, this work has not considered the situation in which a transition can be triggered by multiple input (see, for example [29]) and it would be interesting to extend it to such situations.

ACKNOWLEDGEMENTS

This work was supported in part by Leverhulme Trust grant number F/00275/D, Testing State Based Systems, Natural Sciences and Engineering Research Council (NSERC) of Canada grant number RGPIN 976, and Engineering and Physical Sciences Research Council grant number GR/R43150, Formal Methods and Testing (FORTEST).

REFERENCES

[1] Aho, A. V., Dahbura, A. T., Lee, D., and Uyar, M. U. (1988) An optimization technique for protocol conformance test generation based on UIO sequences and Rural Chinese Postman Tours. *Protocol Specification, Testing, and Verification VIII*, Atlantic City, New Jersey, USA, 7–10 June, pp. 75–86. Elsevier (North-Holland).

[2] Hierons, R. M. and Ural, H. (2006) Optimizing the length of checking sequences. *IEEE Transactions on Computers*, **55**, 618–629.

[3] Hong, H. S., Kim, Y. G., Cha, S. D., Bae, D. H., and Ural, H. (2000) A test sequence selection method for statecharts. *Journal of Software Testing, Verification and Reliability*, **10**, 203–227.

[4] Luo, G., Das, A., and von Bochmann, G. (1994) Generating tests for control portion of SDL specifications. *Protocol Test Systems VI*, Montreal, Quebec, Canada, 10–13 June, pp. 51–66. Elsevier (North-Holland).

[5] Tanenbaum, A. S. (1996) *Computer Networks*, 3rd edition. Prentice Hall International Editions, Prentice Hall, Hemel Hempstead, England.

[6] Farooqui, K., Logrippo, L., and de Meer, J. (1995) The ISO reference model for open distributed processing: an introduction. *Computer Networks and ISDN Systems*, **27**, 1215–1229.

[7] Wang, C. and Schwartz, M. (1993) Fault detection with multiple observers. *IEEE/ACM Transactions on Networking*, **1**, 48–55.

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

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

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

[11] Dssouli, R. and von Bochmann, G. (1985) Error detection with multiple observers. *Protocol Specification, Testing and Verification V*, Toulouse-Moissac, France, 10–13 June, pp. 483–494. Elsevier Science (North Holland).

[12] Dssouli, R. and von Bochmann, G. (1986) Conformance testing with multiple observers. *Protocol Specification, Testing and Verification VI*, Montreal, Quebec, Canada, 10–13 June, pp. 217–229. Elsevier Science (North Holland).

[13] Guyot, S. and Ural, H. (1995) Synchronizable checking sequences based on UIO sequences. *Protocol Test Systems, VIII*, Evry, France, 4–5 September, pp. 385–397. Chapman and Hall.

[14] Hierons, R. M. (2001) Testing a distributed system: generating minimal synchronised test sequences that detect output-shifting faults. *Information and Software Technology*, **43**, 551–560.

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

[16] Khoumsi, A. (2002) A temporal approach for testing distributed systems. *IEEE Transactions on Software Engineering*, **28**, 1085–1103.

[17] Luo, G., Dssouli, R., and von Bochmann, G. (1993) Generating synchronizable test sequences based on finite state machine with distributed ports. *The 6th IFIP Workshop on Protocol Test Systems*, Pau, France, 28–30 September, pp. 139–153. Elsevier (North-Holland).

[18] Rafiq, O. and Cacciari, L. (2003) Coordination algorithm for distributed testing. *The Journal of Supercomputing*, **24**, 203–211.

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

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

[21] Ural, H. and Wang, Z. (1993) Synchronizable test sequence generation using UIO sequences. *Computer Communications*, **16**, 653–661.

[22] Young, Y.-C. and Tai, K.-C. 26–28 March Observational inaccuracy in conformance testing with multiple testers. *IEEE 1st Workshop on Application-Specific Software Engineering and Technology*, Washington, DC, USA, 26–28 March.

[23] Brinksma, E. (1988) A theory for the derivation of tests. *Protocol Specification, Testing, and Verification VIII*, Atlantic City, 7–10 June, pp. 63–74. North-Holland.

[24] Tretmans, J. (1996) Conformance testing with labelled transitions systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, **29**, 49–79.

[25] ITU-T (1997) *Recommendation Z.500 Framework on formal methods in conformance testing*. International Telecommunications Union, Geneva, Switzerland.

[26] ISO/IEC (1995) *Information technology — Open Systems Interconnection, 9646 Parts 1-7*.

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

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

[29] Haar, S., Jard, C., and Jourdan, G.-V. (2007) Testing input/output partial order automata. *19th IFIP TC6/WG6.1 International Conference on Testing of Software and Communicating Systems (TestCom/FATES 2007)*, Tallinn, Estonia, 26–29 June, Lecture Notes in Computer Science, **4581**, pp. 171–185. Springer.