Distinguishing Sequences for Distributed Testing: Adaptive Distinguishing Sequences

Robert M. Hierons[1] and Uraz Cengiz Türker[2]

[1]*Department of Computer Science, Brunel University London, UK.*
[2]*Faculty of Engineering and Natural Sciences, Sabanci University, Turkey*
Email: rob.hierons@brunel.ac.uk, urazc@sabanciuniv.edu

This paper concerns the problem of testing from a finite state machine (FSM) M modelling a system that interacts with its environment at multiple physically distributed interfaces, called ports. We assume that the distributed test architecture is used: there is a local tester at each port, the tester at port p only observes events at p, and the testers do not interact during testing. This paper formalises the notion of an adaptive test strategy and what it means for an adaptive test strategy to be controllable. We provide algorithms to check whether a global strategy is controllable and to generate a controllable adaptive distinguishing sequence (ADS). We prove that controllable ADS existence is PSPACE-Hard and that the problem of deciding whether M has a controllable ADS with length ℓ is NP-Hard. In practice, there is likely to be a polynomial upper bound on the length of ADS in which we are interested and for this case the decision problem is NP-Complete.

Keywords: Finite state machine, distributed test architecture, distinguishing sequences

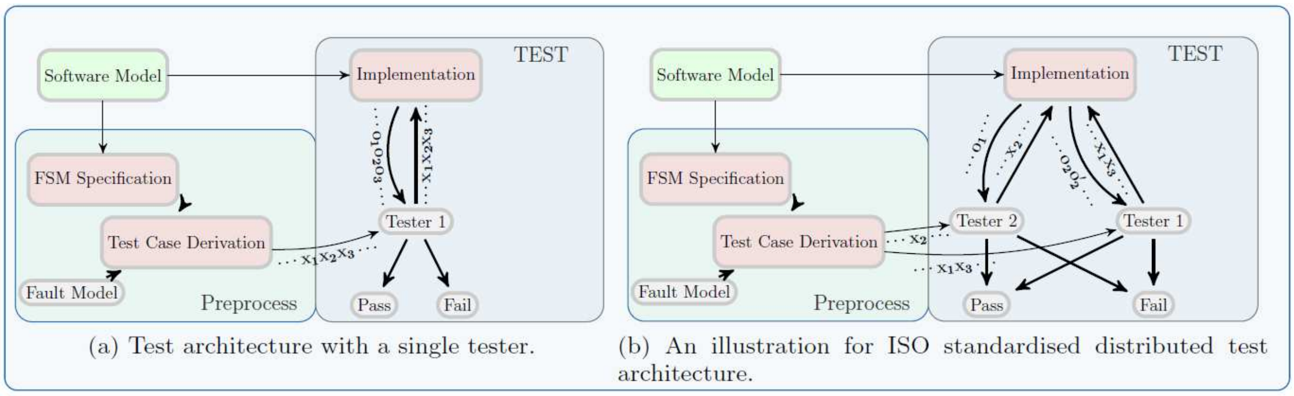
Received 00 January 2009; revised 00 Month 2009

1. INTRODUCTION

Software testing is an important part of the software development process but is typically expensive, manual, and error-prone. This has led to interest in automating parts of testing. One of the most promising approaches to automation is *model based testing (MBT)* in which automation is based on a model [1–8]. In MBT, test cases are typically automatically generated from a model. This model is an abstraction of the specification of the system under test (SUT) or some aspect of interest to the tester. Given a model M, there is the potential to automatically analyse M to generate test cases, direct testing, and check if the observed output is acceptable (solve the Oracle problem). Thus, the manual element in MBT is the production of a model and possibly an adapter that translates between values (inputs and outputs) in the model and those for the SUT. Evidence gathered in an industrial project involving hundreds of testers suggests that the use of MBT can lead to significant benefits [5]. MBT usually utilises state-based models, where the semantics of the model is described in terms of states and transitions between states. There has thus been much interest in testing from finite state machines (FSMs) (see, for example, [9–13]) and input output transition systems (IOTSs) (see, for example, [7, 8]). While test tools might allow the use of richer languages, the models are typically mapped to FSMs or IOTSs for analysis.

FSMs and IOTSs are used to model state-based systems, in which an event (input or output) can lead to a change of state. The response to an input depends on the state of the system/model and so it is necessary to use sequences of inputs in testing. A test case might thus be a sequence of inputs, which is also called an input sequence. However, in some situations there are benefits to using test cases that are adaptive and for such test cases the choice of next input depends on the outputs produced in response to the previous inputs in the test case. For example, a test case might apply input $?i$ and then choose to either apply input $?i$ if output $!o$ is observed or to apply input $?i'$ if output $!o'$ is observed. Such test cases can be represented by trees and will be called strategies.

Most MBT techniques assume that the SUT interacts synchronously with one tester (Figure 1a). However, many systems interact with their environment at multiple physically distributed interfaces, called *ports* (Figure 1b) [14]. Examples of such systems include communications protocols, cloud systems, web services, and wireless sensor networks [15, 16]. In such situations, we might place a separate tester at each port and allow these testers to act independently. When there



(a) Test architecture with a single tester. (b) An illustration for ISO standardised distributed test architecture.

FIGURE 1: Illustrations for different test architectures.

is no global clock and the testers do not synchronise during testing, we are testing in the ISO standardised *distributed test architecture* [17]. Note that the testers might be humans or automated systems.

This paper focuses on testing from a deterministic FSM in the distributed test architecture. The formalism used is multi-port FSMs. In a multi-port FSM, a transition is triggered by an input and can send output to more than one port. This formalism was initially introduced in the context of testing a communications protocol. Here one tester (called the upper tester) acts as the layer above the SUT, while the other tester (called the lower tester) resides on a different machine. Most MBT work on distributed testing has used multi-port FSMs (see, for example, [18–27]).

The use of the distributed test architecture introduces additional challenges. For example, normally we want a test case to be structured so that each tester knows when to supply its inputs and for such a decision to be based on the observations the tester made. Test cases that do not have this property introduce *controllability problems*. Further, the set of observations made by the local testers need not uniquely define the sequence of inputs and outputs (the trace) produced by the SUT and thus testing has reduced observational power (*observability problems*). There are many techniques for generating test cases from an FSM (see, for example, [9–13]) and it is desirable to adapt these techniques for use in distributed testing. Most FSM test techniques use *strategies* (parts of test cases) to distinguish states of the FSM specification M and this motivates our interest in the problem of producing strategies that distinguish states of a multi-port FSM: techniques that produce such strategies have the potential to be used in test case generation methods for distributed testing.

Most FSM-based testing techniques use sequences that distinguish the states of the specification FSM M. Some use separating sequences (input sequences that distinguish two or more states of M) but such techniques may require many test sequences to identify a state and so to test a transition. As a result, many methods instead use a single sequence or adaptive process (*strategy*) to distinguish states of M. Such a strategy is a distinguishing sequence and might be a *preset distinguishing sequence* (PDS) (a single input sequence) or an *adaptive distinguishing sequence* (ADS) (where the next input supplied depends on the response to previous inputs). When compared to separating sequences, the use of PDSs and ADSs can lead to smaller tests that contain fewer test sequences (see, for example, [28]). There has been particular interest in ADSs as the problem of deciding whether a single-port FSM M has a PDS is PSPACE-Complete and the problem of deciding whether a single-port FSM M has an ADS is polynomial time solvable [29].

The above observations motivate the problem investigated in this paper: producing controllable ADSs for distributed testing. There are two reasons for our interest in *controllable* ADSs. First, deciding whether there is a separating sequence for two states is generally undecidable [25]; this problem can be solved in low-order polynomial time if we restrict attention to controllable sequences [24]. Second, if a test case is controllable then the order in which the inputs arrive is predictable, making it easier to trace failures to requirements and simplifying debugging. Interestingly, previous work showed how one can construct test cases from a multi-port FSM M when there are known PDSs [30], although the notion of a PDS used in this previous work potentially required some adaptivity and so might be seen as being a special type of ADS. It should be straightforward to adapt this previously developed approach to use an ADS as defined in this paper. More generally, if we can produce ADSs from multi-port FSMs then there is the potential to adapt current FSM-based test generation methods for use in distributed testing.

This paper makes the following contributions.

1. We discuss adaptive test cases, which we call *test strategies*, for FSM based testing and define both a global strategy (corresponding to there being a single tester that observes and controls all of the

local testers) and a distributed strategy.

2. We define what it means for a global strategy to be controllable and also for a distributed strategy to be deterministic.

3. We prove that a controllable global strategy can be mapped to a deterministic distributed strategy such that these strategies define the same set of possible behaviours. Further, if a global strategy is not controllable then we cannot create a corresponding deterministic distributed strategy.

4. We define what it means for a distributed strategy to be an ADS for all of the states of FSM M and for a subset of its states.

5. We prove that the problem of deciding whether there is a controllable ADS that distinguishes the states of an FSM M is NP-Hard but we leave decidability open.

6. We prove that deciding whether there is an ADS of height at most ℓ is NP-Hard. Where bound ℓ is a polynomial in terms of the size of M this problem is NP-Complete.

7. We provide algorithms to check whether a global strategy is controllable, generate a controllable ADS, and map a controllable ADS to local test cases.

These contributions address several practical issues and have corresponding consequences. First, having formalised ADSs for distributed testing there is the potential to adapt corresponding test case generation algorithms for use in distributed testing or to use the previously defined test case generation method [30]. Second, the results show that the problems of generating ADSs are significantly harder for distributed testing. The problems are more tractable when we have a polynomial upper bound on the lengths of the ADSs and, as explained later, a polynomial upper bound might be provided by the sum of the lengths of a set of separating sequences that distinguish the states of M. There might therefore be value in having test case generation algorithms that use ADSs, where there are ADSs that are sufficiently short, and also more general methods that use separating sequences. Finally, conditions under which ADS existence become simpler have the potential to feed into notions of testability for distributed testing.

The paper is structured as follows. We start in Section 2 by reviewing related work and then, in Section 3, by discussing multi-port FSMs and the distributed test architecture. In Section 4 we formally define the notion of a global ADS and what it means for such an ADS to be controllable and prove that a controllable ADS can be implemented using a set of distributed testers. Section 5 then examines complexity issues related to ADSs. Finally, Section 6 draws conclusions and discusses possible future work. All proofs can be found in the Appendix.

2. RELATED WORK

2.1. Testing from single-port FSMs

There has been much work on generating test sequences from single-port FSMs [9–13]. These approaches often use sequences that force (single-port FSM) M into a particular state and also sequences that distinguish states of M. Between them, these sequences can be used to check the response of M to an input x when in a state s. This is achieved by first moving M into state s, then applying input x, and finally checking that the resultant state is as expected. These are the basis for techniques that take an FSM M and return an m-complete test suite: a test suite that is guaranteed to determine correctness as long as the SUT has no more than m states [9–13]. The benefit of using m-complete test suites is that they provide some guarantees; even if the tester cannot determine a useful upper bound on the number of states of the SUT, the choice of m provides a trade-off between test suite size/cost and test suite effectiveness.

There are several approaches to distinguishing states of an FSM [11, 29]. The weakest approach is to use *separating sequences*, where a separating sequence distinguishes two states of M but may achieve no more than this. The strongest approach is to use a *distinguishing sequence* (DS), which distinguishes all of the states of M. For single-port testing, if M is minimal[3] then for every pair s, s' of states of M there must be a separating sequence for s, s' and, in addition, such a sequence can be generated in low-order polynomial time [31]. If M is not minimal then we can apply a polynomial time algorithm to convert M into an equivalent minimal single-port FSM. Unfortunately, test case generation approaches that use separating sequences require many test sequences to check a single transition. In contrast, we can test a transition through one test sequence if there is a preset distinguishing sequence (PDS): an input sequence that distinguishes all of the states of M. However, the problem of deciding whether a single-port FSM M has a PDS is PSPACE-Complete [29]. This has led to interest in *adaptive distinguishing sequences* (ADSs), where the next input supplied depends on the response to previous inputs. If there is a PDS then there is also an ADS but the converse is not the case. In addition, ADSs can be produced in low-order polynomial time [29]. These results demonstrate the benefits of ADSs: there are efficient algorithms to decide whether a single-port FSM has an ADS and, where an ADS exists, we can use a single test sequence to check a transition. Where there

[3]An FSM M is minimal if no FSM equivalent to M has fewer states.

is no ADS, we can instead use separating sequences.

2.2. Distributed testing from multi-port FSMs

The early work in the area of distributed testing explored the situation in which there is a separate independent (local) tester at each port. If there is no global clock and the local testers do not synchronise during testing then we are testing in the ISO standardised *distributed test architecture* [17]. Sometimes, we allow the testers to exchange coordination messages through a network in order to synchronise their actions (see, for example, [32–35]). However, this can make testing more expensive, since it requires us to establish a network to connect the local testers, and may not be feasible where there are timing constraints. In addition, the message exchange may use the same network as the SUT and so change the behaviour of the SUT. As a result, there has been much interest in testing in the distributed test architecture (see, for example, [18–27]).

Previous work identified two main problems introduced by distributed testing [21, 22, 27]. First, there might be a *controllability problem* in which a local tester, at a port p, cannot determine when to supply an input. Let us consider the interaction given in Figure 2a. The tester at port 1 should start by sending input x_1, this is expected to lead to output o_1 at port 1, and the tester at port 2 should then send input x_2. The problem here is that the tester at port 2 does not observe the earlier input or output and so cannot know when to send its input. Controllability problems lead to non-determinism in testing and so there has been interest in the problem of generating test sequences that do not cause controllability problems [18, 20, 23, 36–39].

Observability problems refer to the fact that, since we only make local observations, we may not be able to distinguish between two different behaviours (global traces). Consider the interaction between two testers and the SUT given in Figure 2b. The specification says that the input of x_1 at port 1 should lead to output o_1 at port 1 and that if we apply x_1 again then we should get o_1 at port 1 and o_2 at port 2. This defines the allowed global trace $x_1/\langle o_1, \varepsilon\rangle\, x_1/\langle o_1, o_2\rangle$ in which ε denotes null output at a particular port. Here the tester at port 1 expects to observe $x_1o_1x_1o_1$ and the tester at port 2 expects to observe o_2. If instead the SUT produced $x_1/\langle o_1, o_2\rangle\, x_1/\langle o_1, \varepsilon\rangle$ then the SUT produced a global trace not allowed by the specification but the local testers made the expected observations: the tester at port 1 observed $x_1o_1x_1o_1$ and the tester at port 2 observed o_2 (Figure 2c). Observability problems can reduce the effectiveness of a test sequence and so there has been interest in producing test sequences that do not suffer from such observability problems [19, 22, 40–42].

Recent work has shown that a number of core testing problems change significantly when we are using the distributed test architecture. For example, if we are testing from an FSM M then it is undecidable whether there is a test case that is guaranteed to move M to a particular state or to distinguish two states s and s' [25]. If we restrict attention to controllable test sequences[4] then there are low-order polynomial time algorithms to decide whether there is a separating sequence for two states [24] and to decide whether there is a controllable sequence that forces M into a particular state [43]. Further, recent work has shown how such separating sequences can be used in test case generation [44]. However, as noted above, if we use separating sequences then we require many test sequences to test a single transition. There is a known method for constructing test cases from a multi-port FSM M when there are PDSs [30], although the notion of a PDS used in this previous work potentially required some adaptivity and so might be seen as being a special type of ADS. The hope is that if we can produce ADSs for distributed testing then these can form the basis for FSM-based test case generation methods and diagnostic methods.

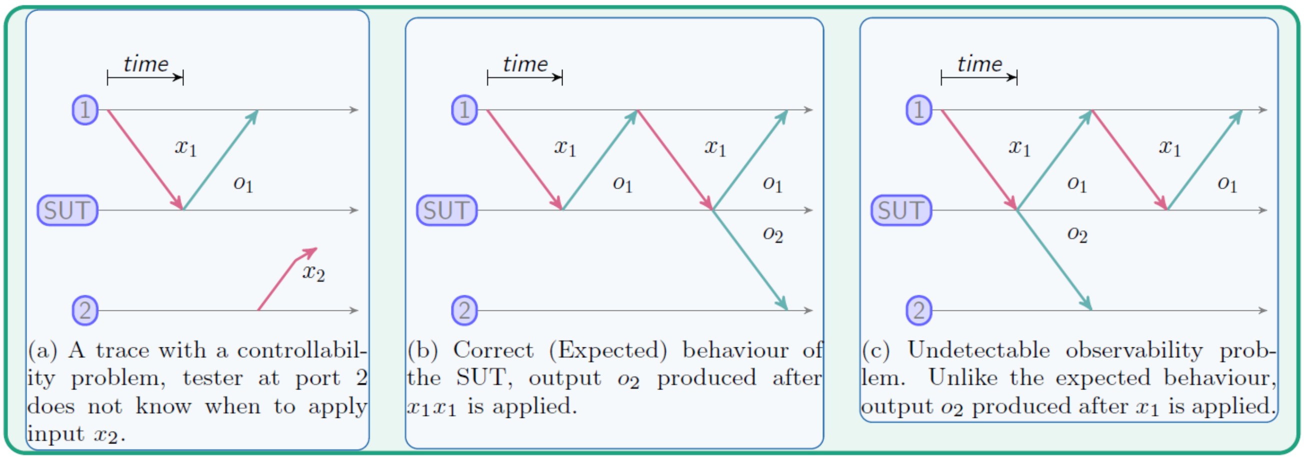
2.3. Adaptive random testing

This subsection briefly reviews another form of adaptivity in testing: adaptive random testing (ART) [45, 46]. This approach is motivated by the belief that one wants a highly diverse test suite that provides an even spread over the input domain. In ART, test case generation is iterative, starting with the empty test suite and adding (and executing) test cases one at a time. In each iteration, there is a current test suite T and a set of candidate test cases C, C being randomly generated. The test case $t \in C$ that is chosen, to be added to T, is one that maximises a notion of 'spread' or 'diversity'. Typically, there is a metric that gives the distance between pairs of test cases and the test case t chosen in the one that has maximal distance from the closest element of T that does not cause a failure. Thus, if d is the metric used and T_C is the set of test cases from T that do not lead to failure, then the test case t chosen maximises the following value.

$$min_{t \in T_C} d(t, t')$$

A number of studies have found that ART outperforms uniform random testing (see, for example, [45]) and ART has been further developed in several directions. For example, ART has been refined so that the next test case is randomly chosen from a restricted domain, an approach that has been called Restricted Random Testing (RRT) [47, 48]. RRT restricts the input domain, from which the next test case is chosen, by excluding regions around the test cases in T_C; those that do not cause a failure. It is also possible to combine ART and partition testing [49] and this at least partially

[4]A test sequence is controllable if it causes no controllability problems. This is formally defined in Section 4.



(a) A trace with a controllability problem, tester at port 2 does not know when to apply input x_2.

(b) Correct (Expected) behaviour of the SUT, output o_2 produced after x_1x_1 is applied.

(c) Undetectable observability problem. Unlike the expected behaviour, output o_2 produced after x_1 is applied.

FIGURE 2: Controllability and observability problems.

addresses one weakness of ART, which is the time taken to choose a next test case. However, although ART and RRT are adaptive, the form of adaptivity used is different from that discussed in this paper.

3. PRELIMINARIES

A multi-port deterministic finite state machine (FSM)[5] has a set $\mathcal{P}$ of ports at which it interacts with its environment. The ports are physically distributed and each has its own input/output alphabet. For each port $p \in \mathcal{P}$ there is a separate local tester that applies the inputs to p and observes the outputs produced at p.

Definition 3.1. *An FSM is defined by a tuple $M = (\mathcal{P}, S, s_0, X, Y, \delta, \lambda)$ where:*

- *$\mathcal{P} = \{1, \dots, k\}$ is the finite set of ports.*
- *S is the finite set of states and $s_0 \in S$ is the initial state.*
- *X is the finite set of inputs and $X = X_1 \cup X_2 \cup \dots \cup X_k$ where X_p $(1 \leq p \leq k)$ is the input alphabet for port p. Given input $x \in X$, $inport(x) = p$ if $x \in X_p$. We consider the projection of an input onto a port and define it as $\pi_p(x) = x$ if $x \in X_p$, and $\pi_p(x) = \varepsilon$ if $x \notin X_p$. We use "ε" to denote an empty/null input or output and also the empty sequence.*
- *$Y = \prod_{p=1}^{k}(Y_p \cup \{\varepsilon\})$ is the set of outputs where Y_p is the output alphabet for port p. An output $y \in Y$ is a vector $\langle o_1, o_2, \dots, o_k \rangle$ where $o_p \in Y_p \cup \{\varepsilon\}$ for all $1 \leq p \leq k$. The notation $\pi_p(y)$ is used to denote the projection of y onto port p, which is simply the p^{th} component of the output vector y. We define $outport(y) = \{p \in \mathcal{P} \mid \pi_p(y) \neq \varepsilon\}$, which is the set of ports at which an output is produced.*
- *δ is the state transfer function of type $S \times X \to S$. If an input $x \in X$ is applied when M is in state s then M changes its state to $\delta(s, x)$.*
- *During a state transition M produces an output vector. The output function $\lambda : S \times X \to Y$ gives the output vector produced in response to an input.*

We assume that all of the alphabets are disjoint and so: for all $p, p' \in \mathcal{P}$, such that $p \neq p'$, we have $X_p \cap X_{p'} = \emptyset$ and $Y_p \cap Y_{p'} = \emptyset$; and X is disjoint from $\cup_{1 \leq i \leq k} Y_k$. This is not a significant restriction since we can label inputs and outputs with port numbers in order to ensure that these conditions hold.

An FSM receives only one input at a time but a transition can produce more than one output. There has been some interest in models (partial order automata) where transitions can be triggered by multiple inputs [50, 51]. It would be interesting to extend the notion of ADSs to such models.

We will use the following terminology in which $M = (\mathcal{P}, S, s_0, X, Y, \delta, \lambda)$.

Definition 3.2. *Given state s and input x, if $\delta(s, x) = s'$ and $\lambda(s, x) = y$ then $\tau = (s, s', x/y)$ is a* transition *of M with* starting state *s,* ending state *s', and* label *x/y. If $(s, s', x/y)$ is a transition of M then the input of x in state s leads to M moving to state s' and producing output y. Given transition $\tau = (s, s', x/y)$ we define $inport(\tau) = inport(x/y) = inport(x)$ and we also define $outport(\tau) = outport(x/y) = outport(y)$ and finally we define $ports(\tau) = ports(x/y) = \{inport(x)\} \cup outport(y)$ to denote the ports used in the transition.*

An FSM can be represented by a directed graph. Figure 3 gives an example of a 2-port FSM M_1 with port set $\{1, 2\}$, state set $\{s_1, s_2, s_3\}$, initial state s_1, inputs $\{x_1\}$ at port 1 and $\{x_2\}$ at port 2, and outputs $\{o_1, o_1'\}$ at port 1 and $\{o_2, o_2'\}$ at port 2. A node represents a state and a directed edge from a node labeled by s to a node labeled by s' with label x/y represents the

[5] In the context of this work, we always use the term FSM to refer to multi-port deterministic finite state machines with more than one port. A classical finite state machine with one port only will be explicitly referred to as a *single-port FSM*.

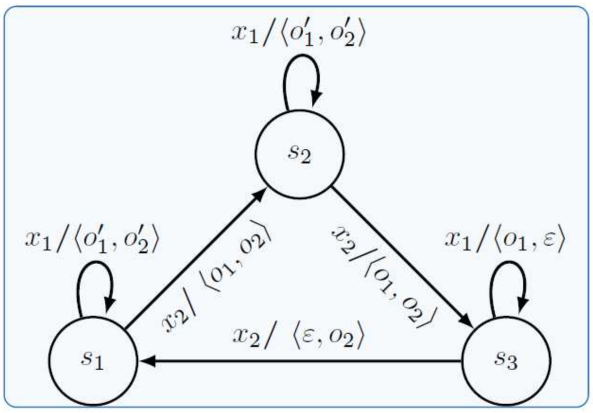

$x_1/\langle o'_1, o'_2\rangle$
s_2
$x_1/\langle o'_1, o'_2\rangle$
$x_2/\langle o_1, o_2\rangle$
$x_2/\langle o_1, o_2\rangle$
$x_1/\langle o_1, \varepsilon\rangle$
$x_2/\ \langle \varepsilon, o_2\rangle$
s_1
s_3

FIGURE 3: Example FSM M_1.

transition $(s, s', x/y)$.

The output and state transfer functions can be extended to input sequences as usual: if we use ε to represent the empty sequence and if $x \in X$ and $\bar{x} \in X^*$ then $\delta(s, \varepsilon) = s$, $\delta(s, x\bar{x}) = \delta(\delta(s, x), \bar{x}))$, $\lambda(s, \varepsilon) = \varepsilon$, $\lambda(s, x\bar{x}) = \lambda(s, x)\lambda(\delta(s, x), \bar{x}))$. The behaviour of FSM M is defined in terms of the labels of walks of M.

Definition 3.3. *A* walk *in M is a sequence $(s_1, s_2, x_1/y_1) \ldots (s_m, s_{m+1}, x_m/y_m)$ of consecutive transitions. This walk has* starting state *s_1, ending state s_{m+1}, and* label *$x_1/y_1\ x_2/y_2 \ldots x_m/y_m$. $x_1/y_1\ x_2/y_2 \ldots x_m/y_m$ is an* input/output sequence*, also called a* global trace*, and $x_1x_2 \ldots x_m$ is the* input portion *and $y_1y_2 \ldots y_m$ is the* output portion *of this global trace.*

Definition 3.4. *An FSM M defines the language $L(M)$ of labels of walks with starting state s_0. Likewise, $L_M(s)$ denotes the set of labels of walks of M with starting state s. Given $S' \subseteq S$, $L_M(S') = \cup_{s \in S'} L_M(s)$ denotes the set of labels of walks of M with starting state in S'. States s, s' are* equivalent *if $L_M(s) = L_M(s')$ and FSMs M and N are* equivalent *if $L(M) = L(N)$. An FSM M is* minimal *if there is no equivalent FSM that has fewer states. An FSM M is* strongly connected *if for every ordered pair (s, s') of states of M there is a walk that has starting state s and ending state s'; a strongly connected FSM M is minimal if and only if $L_M(s) \neq L_M(s')$ for all $s, s' \in S$ with $s \neq s'$.*

An example of a walk in the FSM M_1 from Figure 3 is $(s_3, s_1, x_2/\langle \varepsilon, o_2\rangle)(s_1, s_2, x_2/\langle o_1, o_2\rangle)$, and this has starting state s_3 and ending state s_2. Its label is $x_2/\langle \varepsilon, o_2\rangle\, x_2/\langle o_1, o_2\rangle$, which has input portion x_2x_2 and output portion $\langle \varepsilon, o_2\rangle\langle o_1, o_2\rangle$. $L(M)$ contains the global trace $x_2/\langle o_1, o_2\rangle\ x_1/\langle o'_1, o'_2\rangle$ and $L_M(s_3)$ contains the global trace $x_2/\langle \varepsilon, o_2\rangle\ x_2/\langle o_1, o_2\rangle$.

We use *pre* to denote a function that takes a set of sequences and returns the set of prefixes of these sequences. If $x_1/y_1\ x_2/y_2 \ldots x_m/y_m$ is a trace then its prefixes are of the form $x_1/y_1 x_2/y_2 \ldots x_n/y_n$ for $n \leq m$. We restrict attention to minimal FSMs; an FSM can be rewritten to form a minimal FSM in polynomial time [52].

Assumption 1. We are testing from minimal FSM $M = (\mathcal{P}, S, s_0, X, Y, \delta, \lambda)$.

Since we assume that the ports are physically distributed, no tester observes a global trace: the tester connected to port p will observe only the inputs and outputs at p. We use Σ to denote the set of global observations (inputs and outputs) that a *hypothetical global tester* can observe and Σ_p to denote the set of observations that can be made at port p. Thus, $\Sigma = X \cup Y$ contains all inputs and vectors of outputs while $\Sigma_p = X_p \cup Y_p$ contains only inputs and outputs at p. Let $\sigma \in \Sigma^*$ be a global trace, then $\pi_p(\sigma)$ is the *local trace* at p: a sequence of inputs and outputs at port p (the projection of σ at p).

Definition 3.5. *Given port p, function π_p is defined by the following rules.*
$\pi_p(\varepsilon) = \varepsilon$
$\pi_p((x/\langle o_1, o_2, \ldots, o_m\rangle)\sigma) = \pi_p(\sigma)$ *if* $x \notin X_p \wedge o_p = \varepsilon$
$\pi_p((x/\langle o_1, o_2, \ldots, o_m\rangle)\sigma) = x\pi_p(\sigma)$ *if* $x \in X_p \wedge o_p = \varepsilon$
$\pi_p((x/\langle o_1, o_2, \ldots, o_m\rangle)\sigma) = o_p\pi_p(\sigma)$ *if* $x \notin X_p \wedge o_p \neq \varepsilon$
$\pi_p((x/\langle o_1, o_2, \ldots, o_m\rangle)\sigma) = xo_p\pi_p(\sigma)$ *if* $x \in X_p \wedge o_p \neq \varepsilon$

Since the local testers observe only the local projections of global traces, these testers can only distinguish two global traces if one or more of their local projections differ.

Definition 3.6. *Two global traces σ_1, σ_2 are* indistinguishable*, written $\sigma_1 \sim \sigma_2$, if for all $p \in \mathcal{P}$ we have that $\pi_p(\sigma_1) = \pi_p(\sigma_2)$.*

For instance, let us consider FSM M_1 given in Figure 3, and global traces $\sigma_1 = x_2/\langle o_1, o_2\rangle\ x_2/\langle \varepsilon, o_2\rangle$, $\sigma_2 = x_2/\langle \varepsilon, o_2\rangle\ x_2/\langle o_1, o_2\rangle$, then $\pi_1(\sigma_1) = o_1$, $\pi_2(\sigma_1) = x_2o_2x_2o_2$, $\pi_1(\sigma_2) = o_1$ and $\pi_2(\sigma_2) = x_2o_2x_2o_2$ and so $\sigma_1 \sim \sigma_2$. Finally we use $|.|$ to denote the cardinality of a set or the length of a sequence i.e. continuing the above example we have $|\mathcal{P}| = 2$ and $|\pi_2(\sigma_1)| = 4$.

Table 1 summarises some of the terminology introduced.

4. TEST STRATEGIES FOR DISTRIBUTED TESTING

4.1. Test Strategies

This section defines test strategies for use in distributed testing and explores properties of such strategies. Essentially, a test strategy is similar to a strategy in game theory [53] and is a process that determines the actions of a tester. Test strategies have been discussed for testing from a single-port FSM [53] and multi-ports FSMs (where a local tester has its own strategy) [25]. However, as we explain below, the formalisation in this section is, by necessity, different. In addition, the notion of a strategy (for testing from an FSM) being controllable has not previously been discussed

TABLE 1: Terminology

Notation/terminology	Meaning
test sequence	a sequence of inputs
adaptive test case/strategy	a process that determines the next input to apply on the basis of previous input/output
separating sequence	an input sequence that leads to different output sequences for two given states
controllable test case	a test case in which a local tester can determine when to supply its inputs
$\delta(s,x)$	the state reached if input x is received when M is in state s
$\lambda(s,x)$	the output produced if input x is received when M is in state s
$L(M)$	the set of input/output sequences that label walks from the initial state of M
$pre(\sigma)$	the set of prefixes of σ
$\pi_p(\sigma)$	the projection of trace σ onto port p
$\sigma_1 \sim \sigma_2$	for all $p \in \mathcal{P}$, $\pi_p(\sigma_1) = \pi_p(\sigma_2)$

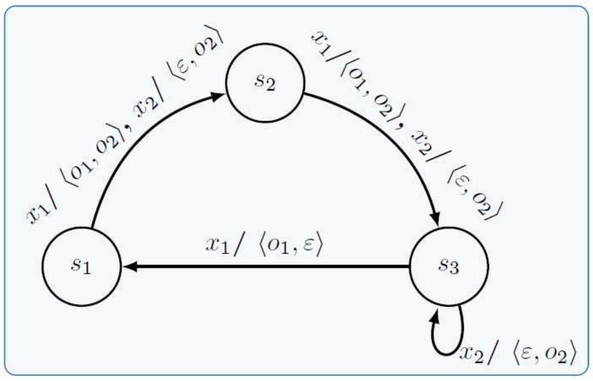

s_1
s_2
s_3
$x_1/\langle o_1,o_2\rangle, x_2/\langle\varepsilon,o_2\rangle$
$x_1/\langle o_1,o_2\rangle, x_2/\langle\varepsilon,o_2\rangle$
$x_1/\langle o_1,\varepsilon\rangle$
$x_2/\langle\varepsilon,o_2\rangle$

FIGURE 4: FSM M_2 for Example 1.

and previous work has not considered the problem of using a strategy to distinguish more than two states in distributed testing.

The main result in this section is that if a global strategy μ is controllable then it can be mapped to a set of local test cases (a distributed strategy) that implement μ. In Section 5 this will allow us to focus on problems associated with generating controllable global strategies that achieve certain objectives. Some readers might want to skip much of Section 4, although in Section 5 we will use the notion of the set of evolutions of a strategy μ (the set of traces that can result from applying μ) and the evolutions of μ from a state set S' (the set of traces that can result from applying μ in states from S'). Table 2 provides a list of terms introduced in this section, along with corresponding definition numbers.

A *global strategy* will be a test strategy in which a single tester observes all inputs and outputs. Thus, since the hypothetical global observer receives the global trace, the usual definition of an ADS applies. We will call these *traditional* ADSs. However, observability problems can reduce their effectiveness.

Example 1. Consider the FSM given in Figure 4. We have that x_1x_1 is a traditional ADS (also a traditional PDS) since it leads to different global traces from the states: from s_1 we have $x_1/\langle o_1,o_2\rangle\, x_1/\langle o_1,o_2\rangle$; from s_2 we have $x_1/\langle o_1,o_2\rangle\, x_1/\langle o_1,\varepsilon\rangle$; and from s_3 we have $x_1/\langle o_1,\varepsilon\rangle\, x_1/\langle o_1,o_2\rangle$. However, if we consider the local traces we find that x_1x_1 does not distinguish states s_2 and s_3 in distributed testing since in each case the projection at port 1 is $x_1o_1x_1o_1$ and the projection at port 2 is o_2.

We therefore have the following.

Proposition 1. Given FSM M, a traditional ADS of M might fail to distinguish some states of M when only local observations are made.

We will define the notion of a *distributed test strategy*, which is a tuple of local strategies; a local strategy μ_p is assigned to a port p and a tester at p applies μ_p.

In Section 4.2 we define what we mean by a global strategy and what it means for such a strategy to be controllable. We also define what it means for a global strategy to be an adaptive distinguishing sequence. In Section 4.3 we then consider distributed test strategies and show that a controllable global strategy for FSM M can be mapped to a distributed test strategy that is equally effective in testing from M. Since we are interested in controllable testing, this will show that it is sufficient to generate global strategies that achieve given objectives and produce distributed strategies from these. This is useful since typical testing objectives, such as reaching a state of M or distinguishing states of M, are described in terms of the global behaviour of M. The focus of the paper will then be on generating controllable global strategies.

4.2. Global Strategies

In this section we define the notion of a global strategy μ and the set of behaviours (evolutions) that can occur when using μ. Based on this, we define what it means for a global strategy to be controllable and to be an ADS.

When testing from an FSM, an observation is a trace and the global tester makes a decision, regarding what to do next, on the basis of such a trace. A global

TABLE 2: Definitions

Notation/terminology	Corresponding definition
An evolution and the sets of evolutions of global strategy μ ($Ev(\mu), Ev(\mu, M, s), Ev(\mu, M, S')$)	Definition 4.2
A strategy being controllable	Definition 4.3
A global strategy being an ADS	Definition 4.5
A local strategy	Definition 4.6
Distributed strategy (a tuple $(\mu_1, \ldots, \mu_k)$ in which $\mu_1, \ldots, \mu_k$ are local strategies)	Definition 4.7
A distributed strategy being deterministic	Definition 4.8
Evolutions of a deterministic distributed strategy	Definition 4.10
Projection of a global strategy ($\pi_p(\mu)$ and $\pi_p^{S'}(\mu)$)	Definitions 4.11 and 4.12
A distributed strategy being an ADS	Definition 4.13

strategy μ is therefore a partial function from $(X/Y)^*$ to X, where $(X/Y)^*$ denotes the set of traces. Thus, if the SUT produces trace σ then $\mu(\sigma)$ determines what the tester does next: if $\mu(\sigma) = x$ ($x \in X$) then the tester applies x and otherwise μ is not defined on σ and testing ends. We include a finiteness requirement to ensure that testing terminates.

DEFINITION 4.1. *A* global strategy *μ is a partial function from $(X/Y)^*$ to X such that only finitely many traces from $(X/Y)^*$ are mapped to elements of X.*

When the global tester applies a strategy μ one obtains an *evolution*[6] of μ [25]. We can restrict the set of evolutions if we start testing an FSM M when it is in state s since we must observe a trace from $L_M(s)$. Similarly, we can define the set of evolutions when applying a global strategy from some set S' of states.

DEFINITION 4.2. *Trace σ is an* evolution *of global strategy μ if the following hold.*

1. *If $\sigma' x/y$ is a prefix of σ for $x \in X$ and $y \in Y$ then $\mu(\sigma') = x$.*
2. *If σ' is a prefix of σ and $\mu(\sigma') = x$ then there exists $y \in Y$ such that $\sigma' x/y$ is prefix of σ.*

Given global strategy μ, we let $Ev(\mu)$ denote the set of evolutions of μ. Given FSM M and state s of M, we let the set of evolutions of μ from s be $Ev(\mu, M, s) = Ev(\mu) \cap L_M(s)$. Further, given FSM M with set of states S and $S' \subseteq S$ we let the set of evolutions of μ from S' be $Ev(\mu, M, S') = \bigcup_{s \in S'} Ev(\mu, M, s)$. We say that the length of the longest evolution in $Ev(\mu, M, S)$ is the height *of μ.*

This states that an input will only be applied after σ' if this is specified by the strategy and also that whenever the strategy can apply an input it does so. We will assume that a global strategy μ is not defined on traces that cannot occur when μ is applied and so $\sigma \notin pre(Ev(\mu))$ implies that μ is not defined on σ. Clearly, this does not reduce the effectiveness of the global strategies we consider; it simply avoids some redundancy.

[6]Previous work concerned a single (local) tester and so strategies were mappings from Σ_p^*.

ASSUMPTION 2. Global strategy μ is only defined on traces in $pre(Ev(\mu))$.

While executing a global strategy a controllability problem may arise.

EXAMPLE 2. Let $x_1, x_1' \in X_1$ be different inputs at port 1, $x_2 \in X_2$, and let us suppose that there are traces $x_2/\langle o_1, o_2\rangle\ x_1/\langle y, \varepsilon\rangle$ (Figure 5a) and $x_2/\langle \varepsilon, o_2\rangle x_2/\langle o_1, \varepsilon\rangle x_1'/\langle y', \varepsilon\rangle$ (Figure 5b) in $pre(Ev(\mu))$. Since $\pi_1(x_2/\langle o_1, o_2\rangle) = \pi_1(x_2/\langle \varepsilon, o_2\rangle\ x_2/\langle o_1, \varepsilon\rangle) = o_1$, tester 1 cannot differentiate between $x_2/\langle o_1, o_2\rangle$ and $x_2/\langle \varepsilon, o_2\rangle\, x_2/\langle o_1, \varepsilon\rangle$, and so it cannot know which input (x_1 or x_1') to send after observing o_1.

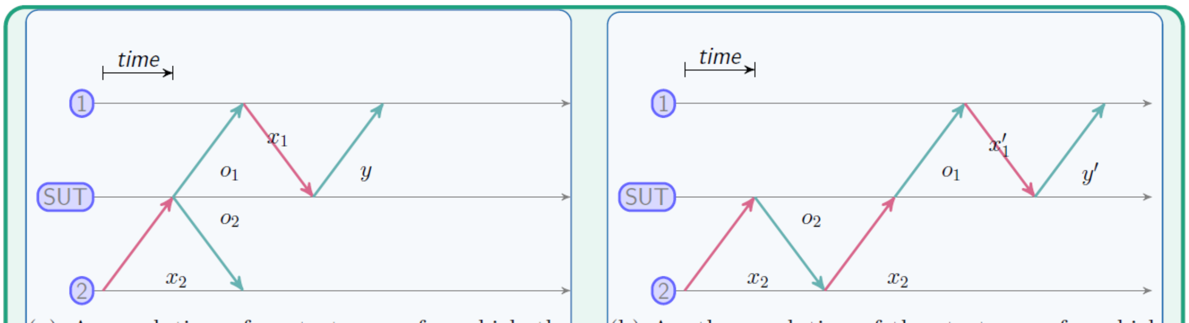
For a global strategy to be controllable, the actions at a port p should only depend on the observations made at p and so if there is a local tester at p then this local tester knows when to supply input. This is characterised by saying that if two evolutions σ and σ' have the same observations at p ($\pi_p(\sigma) = \pi_p(\sigma')$) then the tester at p performs the same action after each.

DEFINITION 4.3. *Global strategy μ is* controllable *if for all $\sigma, \sigma' \in pre(Ev(\mu))$, if there exists port p such that $\pi_p(\sigma) = \pi_p(\sigma')$ and $\mu(\sigma) \in X_p$ then $\mu(\sigma') = \mu(\sigma)$.*

We can now adapt the notion of controllability to the case where we have a strategy and a set S' of states from which we might apply this. To do this it is sufficient to restrict attention to evolutions that can occur from S'.

DEFINITION 4.4. *Given set S' of states of M, strategy μ is* controllable for S' *if for all $\sigma, \sigma' \in pre(Ev(\mu, M, S'))$, if there exists port p such that $\pi_p(\sigma) = \pi_p(\sigma')$ and $\mu(\sigma) \in X_p$ then $\mu(\sigma') = \mu(\sigma)$.*

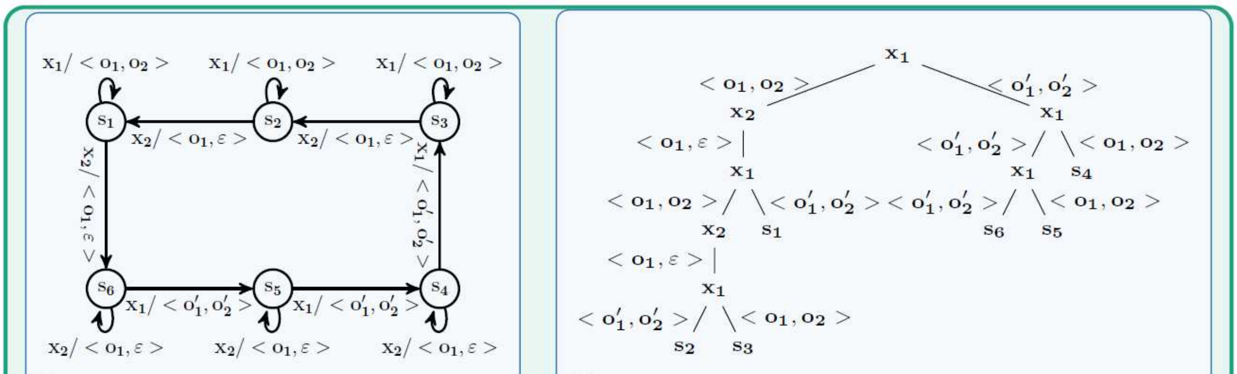
The following shows that this is less restrictive than controllability: a strategy might be controllable for a given M and S' but not controllable in general. The key point is that if we fix M and S' then we only need to consider the behaviour that can result from applying



(a) An evolution of a strategy μ for which the tester at port 1 observes o_1 and applies input x_1.

(b) Another evolution of the strategy μ for which the tester at port 1 observes o_1 and applies input x_1'.

FIGURE 5: A simple controllability problem. Strategy μ can lead to two different evolutions such that the tester at port 1 receives o_1 but supplies different inputs (x_1, x_1').



(a) An FSM M with input x_1 at port 1, and x_2 at port 2, output alphabet $Y_1 = \{o_1, o_1'\}$ at port 1, and $Y_2 = \{o_2, o_2'\}$ at port 2.

(b) A global controllable ADS μ for M given in Figure 6a.

FIGURE 6: An FSM M and a controllable global ADS for M.

the strategy μ from states in S'.

Proposition 2. If a strategy μ is controllable then for every FSM M and set S' of states of M, we have that μ is controllable for S'. However, it is possible that strategy μ is controllable for S' for some FSM M and state set S' and yet μ is not controllable.

We now consider the problem of checking whether a strategy μ is controllable for an FSM M and below we give an algorithm that achieves this. Note that previous work has explored the problem of deciding whether a global test case, for testing from an input output transition system (IOTS), is controllable [54]. The algorithm given in this paper is similar to that previously defined for IOTSs, although it differs slightly as a result of the differences in the formalisms. In particular, for FSMs we know that input and output alternate but this is not the case for IOTSs; by taking advantage of this property we can produce a simpler algorithm.

For a strategy μ, Algorithm 1 retrieves a set SEQ of (maximal) sequences that can result from applying μ to M i.e. $SEQ = Ev(\mu, M, S)$. Afterwards it checks if these sequences cause a controllability problem.

Recall that we have a controllability problem if and only if there are different prefixes σ, σ' such that the following hold:

1. there exists a port p such that $\pi_p(\sigma) = \pi_p(\sigma')$
2. μ returns input at port p after at least one of σ and σ'
3. $\mu(\sigma) \neq \mu(\sigma')$

The first condition says that σ and σ' look the same to the tester at port p ($\pi_p(\sigma) = \pi_p(\sigma')$). Thus, there is a controllability problem if μ behaves differently at port p after σ and σ'. The second and third conditions check this: if the second condition holds then μ applies input at p after at least one of σ and σ' and so there is a controllability problem if it does not apply the same input after these ($\mu(\sigma) \neq \mu(\sigma')$).

We now explain how Algorithm 1 operates and use σ_k^p to denote the trace formed by taking the prefix of σ with $k-1$ inputs and then taking the projection at port

p. The algorithm first generates the set SEQ (Line 1) of sequences that can result from applying σ in M and initiates a boolean variable C (Line 2); C will be set to false if the strategy μ is not controllable. For each sequence σ in SEQ and each $1 < k \leq \ell$, where ℓ is the length of σ, we retrieve the port p that applies the kth input x_k of σ (we do not need to consider the case $k = 1$ since the first input cannot cause a controllability problem). We also compute the prefix of σ of length $k-1$ (i.e. the observations that precede the input of x_k) and take its projection σ_k^p at port p. We then check whether the port that applies x_k is in the set $ports(x_{k-1}/y_{k-1})$ of ports that observe input/output in the previous transition (Lines 7–8); if this is not the case then σ is not a trace produced by a controllable strategy since the tester at p cannot know when to send input x_k to ensure that it arrives after x_{k-1}. Finally for each $\sigma' \in SEQ \setminus \{\sigma\}$, we first check whether σ_k^p is a prefix of the projection $\pi_p(\sigma')$ of σ' on port p. If this is the case then for μ to be controllable we require that μ applies input x_k as soon as σ_k^p is observed and so after the shortest prefix of σ' whose projection at p is σ_k^p; this is checked in Line 12.

We now consider the complexity of Algorithm 1 and let n denote the number of states of M and ℓ denote the height of μ. Note that the algorithm uses three nested loops. As the cardinality of SEQ is bounded above by n, the outer loop iterates at most n times. At each iteration the algorithm retrieves the prefix with k inputs and as k varies from 1 to ℓ, the number of steps executed by this loop is of $O(n\ell)$. The innermost loop iterates $O(n)$ times and on each iteration it considers a sequence of $O(\ell)$ length. Thus, the number of steps used by the algorithm is of $O(n^2\ell^2)$ and so we can conclude that Algorithm 1 is a polynomial time algorithm with respect to the size of its input.

A global strategy μ is an ADS if it produces different observations from the states being considered and so distinguishes them[7].

DEFINITION 4.5. *Global strategy μ is an* adaptive distinguishing sequence (ADS) *for state set $S' \subseteq S$ of M if for all $s, s' \in S'$ with $s' \neq s$, $\sigma \in Ev(\mu, M, s)$, and $\sigma' \in Ev(\mu, M, s')$, we have that $\sigma \not\sim \sigma'$. Further, μ is an* adaptive distinguishing sequence (ADS) *for M if it is an adaptive distinguishing sequence for S.*

The difference, when compared to ADSs for testing from a single-port FSM, is that we compare global traces using $\sim$ rather than equality, this being an inevitable consequence of the reduced observational power of distributed testing.

We now provide a recursive algorithm that constructs a controllable global ADS for an FSM M, if such an ADS exists. But before going further we introduce the notion of a test case being *weakly controllable* that will be used in this algorithm.

Recall that whenever traces σ, σ' are indistinguishable at port p, we need that $\mu(\sigma)$ and $\mu(\sigma')$ are 'consistent' at p: either neither returns input at p or $\mu(\sigma) = \mu(\sigma')$ (see Line 12 of Algorithm 1). However, while constructing a strategy μ we should allow the case where $\mu(\sigma) = x$, $x \in X_p$ and $\mu(\sigma') = \varepsilon$ even though such a strategy is not controllable. This stems from the following observation: we may extend such a strategy μ that is not controllable to form one in which $\mu(\sigma) = x$ and $\mu(\sigma') = x$. That is to say, a recursive algorithm might produce a strategy μ that is not controllable but that can be extended to form a controllable strategy as the algorithm progresses. So while constructing a strategy if we declare that μ is not controllable, whenever we have $\mu(\sigma) = x$ and $\mu(\sigma') = \varepsilon$, we may not be able to find the desired final strategy. Thus, we should not discard such strategies. In order to achieve this we use another version of Algorithm 1 that checks whether a test case is weakly controllable. This algorithm can be obtained from Algorithm 1 by changing line 12 as follows.

```
12   $\mu(\sigma'') \neq x_k$ and $\mu(\sigma'') \neq \varepsilon$
```

If a strategy fails this condition then we have traces σ and σ' that are indistinguishable at port p such that: the behaviour of μ is different at p after σ and σ' and $\mu(\sigma)$ and $\mu(\sigma')$ both define inputs. Importantly, such a strategy (one that is not weakly controllable) cannot be extended to form a controllable strategy.

The pseudocode for constructing a controllable ADS is given in Algorithm 2. The algorithm receives an FSM (M), a tuple of input symbols (Υ) a non-negative integer (h) and a global strategy (μ) as its inputs. At each recursive step, the algorithm forms and extends a strategy in a depth-first search manner. The algorithm terminates if it finds a controllable ADS or the strategies being constructed reach a preset height ℓ.

The inputs are M, $\Upsilon = \emptyset$, $h = 0$, and $\mu = \emptyset$. During a recursive step the algorithm first initialises a set of sets of sequences (*traces set*) $\Pi = \{\{\varepsilon\}\}$. Set Π is constructed so that it contains the equivalence classes of $Ev(\mu, M, S)$, under equivalence relation $\sim$: a controllable strategy must behave in the same way after two traces that are equivalent under $\sim$. Set Π is formed by considering the traces in $Ev(\mu, M, S)$ one at a time, with two traces σ and σ' being in the same set Ψ_i if and only if $\sigma \sim \sigma'$ (Lines 2–7).

After the set Π has been formed, for every $\Psi_i \in \Pi$ the algorithm retrieves the ith symbol x from the tuple Υ. Then for each trace σ_s in Ψ_i the algorithm updates the strategy to $\mu \leftarrow \mu \cup \{(\sigma_s, x)\}$ (Lines 8–11) or makes no change if $x = \varepsilon$.

When μ is updated the set Π is formed again, with the algorithm executing the instructions provided on Lines 1–7 (Line 12). Afterwards, the algorithm checks whether the cardinality of set Π is equal to the number

[7]If an FSM does not have an ADS then there may still be value in producing ADSs for subsets of states, a problem that has been explored for single-port FSMs [55]

ALGORITHM 1: A routine to check whether a given global strategy μ is controllable.

```
Input: An FSM $M$, strategy $\mu$
Output: True or False
begin
1     Construct set $SEQ \leftarrow \{\sigma_1, \sigma_2, \dots, \sigma_{|SEQ|}\}$
2     $C \leftarrow True$// C will set to false if $\mu$ is not controllable for $M$.
3     foreach $\sigma \in SEQ$ do
4         Let $\sigma$ be the sequence $x_1/y_1, x_2/y_2, \dots, x_\ell/y_\ell$
5         foreach $1 < k \leq \ell$ do
6             Let $p \leftarrow port(x_k)$ and $\sigma_k^p \leftarrow \pi_p(x_1/y_1, x_2/y_2, \dots, x_{k-1}/y_{k-1})$
7             if $p \notin ports(x_{k-1}/y_{k-1})$ then
8                 $C \leftarrow False$
9             foreach $\sigma' \in SEQ \setminus \{\sigma\}$ do
10                if $\sigma_k^p$ is a prefix of $\pi_p(\sigma')$ then
11                    Let $\sigma''$ be the shortest prefix of $\sigma'$ such that $\pi_p(\sigma'') = \sigma_k^p$
12                    if $\mu(\sigma'') \neq x_k$ then
13                        $C \leftarrow False$
14    Return $C$
```

ALGORITHM 2: Generating a controllable ADS for FSM M.

```
Input: An FSM $M$, tuple of input symbols $\Upsilon$, non-negative integer $h$, strategy $\mu$
Output: An ADS for $M$
begin
1     $\Pi \leftarrow \{\{\varepsilon\}\}$// Initialise the trace set.
2     foreach $s \in S$ do
3         $\sigma_s \leftarrow Ev(\mu, M, s)$// Retrieve the longest trace obtained from state $s$ by strategy $\mu$.
4         if there exist no set $\Psi$ in $\Pi$ such that $\sigma_{s'} \in \Psi$ and $\sigma_s \sim \sigma_{s'}$ then
5             $\Psi \leftarrow \emptyset$// Let $\Psi$ be an empty set of sequences.
6             $\Pi \leftarrow \Pi \cup \Psi$// Introduce set $\Psi$.
7         $\Psi \leftarrow \Psi \cup \{\sigma_s\}$.// Add corresponding trace to set $\Psi$ such that $\sigma_s$ and $\sigma_{s'}$ are in same set if
            $\sigma_s \sim \sigma_{s'}$.
8     foreach $\Psi \in \Pi$ and $\Upsilon \neq \emptyset$ do
9         $x \leftarrow \Upsilon(\Psi)$// Pick input for $\Psi$
10        foreach $\sigma_s \in \Psi$, $x \neq \varepsilon$ do
11            $\mu \leftarrow \mu \cup \{(\sigma_s, x)\}$
12    Repeat instructions between Lines 1–7 // Retrieve traces and update $\Pi$ such that $\sigma_s$, $\sigma_{s'}$ are in same set
        if $\sigma_s \sim \sigma_{s'}$.
13    if $|\Pi| = |S|$ then
14        if $\mu$ is controllable then
15            Return $\mu$
16    if $h < \ell$ then
17        if $\mu$ is weakly controllable then
18            foreach different $|\Pi|-$tuples of set $X \cup \{\varepsilon\}$ do
19                Return $ADS(M, \Upsilon, h \leftarrow h+1, \mu)$// For each different $|\Pi|$-tuples of inputs, construct $\Upsilon$
                    and call algorithm.
20    Return $null$
```

of states of the FSM (Line 13). If the cardinality of Π is equal to the number of states of the FSM, and the new strategy is controllable (Line 14), then the strategy is a controllable ADS and this is returned.

If the cardinality of Π is less than n or the strategy is not controllable then the algorithm checks whether h has reached the preset bound (Line 16) and returns null if this is the case (Line 20). Otherwise, the algorithm checks if the strategy is weakly controllable (Line 17). If the strategy is not weakly controllable then the algorithm returns null (Line 20). Otherwise, for each different $|\Pi|$-tuple of the set $X \cup \{\varepsilon\}$, the algorithm generates Υ, which is a mapping from each equivalence class to a next input or ε (representing no input being added in this iteration of the loop). It then calls itself with M, Υ, $h = h + 1$, and μ (Lines 18–19). The role of Υ in this recursive step ensures that the algorithm considers all ways of extending the current strategy. We include ε as an option so that evolutions of a strategy can vary in length.

As we generate different $|\Pi|$-tuples of set $X \cup \{\varepsilon\}$ (Line 18), the time complexity of the algorithm is exponential in the size of M. However, since at each recursive call the algorithm processes M, Υ, h and μ, the proposed ADS construction algorithm requires polynomial space if the preset bound ℓ can be described by a polynomial function in the size of M. Moreover, the algorithm is guaranteed to return a controllable ADS if there exists a controllable ADS of height ℓ or less.

The following shows how the global ADS in Figure 6 can be generated from the FSM in the same figure when using upper bound $\ell = 5$.

Example 3. Let us suppose that we provide M (given in Figure 6a) along with $\emptyset$, $\Upsilon = \{x_1, x_2\}$ and $h = 0$ to Algorithm 2 as inputs and we assume that $\ell = 5$. Note that h is provided as input in order to allow recursion; if an ADS of height h is not found and $h < \ell$ then the recursive step is applied. Initially, $\mu = \{\}$ and so if we apply μ in a state of M then the resultant trace is the empty sequence ε. Thus, after executing Lines $2 - 7$ we obtain $\Pi = \{\Psi_0\}$ with $\Psi_0 = \{\varepsilon\}$. The algorithm will then select an input for Ψ_0 (Lines 8-11) and let us suppose that x_1 is chosen.

At this point the algorithm has generated a strategy μ of depth 1; one that initially applies x_1. It updates Π with the equivalence classes of the set of resultant traces. There are two traces that can be produced using this strategy: from states s_1, s_2, and s_3 there is output $\langle o_1, o_2\rangle$ (and so $\sigma_1 = \sigma_2 = \sigma_3 = x_1/\langle o_1, o_2\rangle$) and from states s_4, s_5, and s_6 there is output $\langle o'_1, o'_2\rangle$ (and so $\sigma_4 = \sigma_5 = \sigma_6 = x_1/\langle o'_1, o'_2\rangle$). The set of traces is thus $\{x_1/\langle o_1, o_2\rangle, x_1/\langle o'_1, o'_2\rangle\}$ and we obtain two equivalence classes: $\Psi_1 = \{x_1/\langle o_1, o_2\rangle\}$ and $\Psi_2 = \{x_1/\langle o'_1, o'_2\rangle\}$. The algorithm assigns $\{\Psi_1, \Psi_2\}$ to Π (line 12).

The algorithm now checks whether μ distinguishes the states of M (whether $|\Pi| = |S|$). Since this is not the case, the algorithm checks whether the bound on the depth has been reached. Since $\ell = 5$ and the current value of h is 1, the algorithm then checks whether μ is weakly controllable (it clearly is). The algorithm is then recursively called with μ, Υ, and $h = 2$.

In the next phase, the algorithm repeats the above procedure for each Ψ_i. In order to produce the ADS in Figure 6b, it chooses input x_2 for Ψ_1 and input x_1 for Ψ_2. The resultant set of traces is $\{x_1/\langle o_1, o_2\rangle x_2/\langle o_1, \varepsilon\rangle, x_1/\langle o'_1, o'_2\rangle\ x_1/\langle o_1, o_2\rangle, x_1/\langle o'_1, o'_2\rangle\ x_1/\langle o'_1, o'_2\rangle\}$. We therefore have three sets of traces: $\Psi_1 = \{x_1/\langle o_1, o_2\rangle\ x_2/\langle o_1, \varepsilon\rangle\}$, $\Psi_2 = \{x_1/\langle o'_1, o'_2\rangle\ x_1/\langle o_1, o_2\rangle\}$, and $\Psi_3 = \{x_1/\langle o'_1, o'_2\rangle x_1/\langle o'_1, o'_2\rangle\}$. It is straightforward to see that this global strategy is weakly controllable. Since since $|\Pi|$ is 3, and so is less than $|S|$, the process continues. In the next iteration the algorithm chooses input x_1 for Ψ_1, ε (no input) for Ψ_2, and x_1 for Ψ_3. The process continues, potentially returning $True$ and the global ADS shown in Figure 6b.

We now show how this global ADS can be used to identify a state (in this case s_3).

Example 4. From the ADS given in Figure 6b, initially the strategy μ returns value $\mu(\varepsilon) = x_1$. Let us suppose that when the tester applies input x_1 and the FSM produces output $\langle o_1, o_2\rangle$. The global strategy states that the next input to apply is $\mu(x_1/\langle o_1, o_2\rangle) = x_2$. Let us suppose that in response to input x_2, M produces $\langle o_1, \varepsilon\rangle$. This pattern repeats until the global strategy μ produces ε as follows: $\mu(x_1/\langle o_1, o_2\rangle\ x_2/\langle o_1, \varepsilon\rangle) = x_1$, $\mu(x_1/\langle o_1, o_2\rangle\ \ x_2/\langle o_1, \varepsilon\rangle\ \ x_1/\langle o_1, o_2\rangle) = x_2$, $\mu(x_1/\langle o_1, o_2\rangle\ x_2/\langle o_1, \varepsilon\rangle\ x_1/\langle o_1, o_2\rangle\ x_2/\langle o_1, \varepsilon\rangle) = x_1$, $\mu(x_1/\langle o_1, o_2\rangle x_2/\langle o_1, \varepsilon\rangle x_1/\langle o_1, o_2\rangle x_2/\langle o_1, \varepsilon\rangle x_1/\langle o_1, o_2\rangle) = \varepsilon$. Testing then ends and from the global trace we know that initially M must have been in state s_3.

4.3. Local and Distributed Test Strategies

This section defines what we mean by local and distributed strategies. The main results are the definition of projection functions that map global strategies to local strategies and the proof that if a global strategy is controllable then we can implement it using a distributed strategy. This shows the value of generating controllable global strategies for use in testing, which is the problem we investigate in this paper.

Recall that a global strategy is a partial function from $(X/Y)^*$ to X and so it might appear that a local strategy for port p will be a partial function from $(X_p/Y_p)^*$ to X_p. However, the observations made by the tester at port p need not alternate between inputs and outputs. For example, there might be a trace such as $\sigma = x_1/\langle \varepsilon, o_2\rangle\ x_1/\langle o_1, o_2\rangle\ x_2/\langle o_1, \varepsilon\rangle$ and here $\pi_2(\sigma) = o_2 o_2 x_2$. As a result, a local strategy will be a partial function from sequences of observations at p (elements of Σ_p^*) and not from $(X_p/Y_p)^*$. We include ε in the set of values that can be returned by the local strategy, with this denoting the case where the tester waits to observe further output. In contrast to global strategies, if the local tester at port p chooses to not send an input then it might not have terminated since input can be supplied by other local testers and this can result in additional observations at p.

Definition 4.6. *A local strategy μ_p for port p is a function from Σ_p^* to $X_p \cup \{\varepsilon\}$ such that only finitely many traces from Σ_p^* are mapped to X_p.*

In distributed testing the overall test system can be seen as a set of local testers and thus can be represented by a tuple of local strategies.

Definition 4.7. *Given port set $\mathcal{P} = \{1, 2, \ldots, k\}$, a* distributed strategy *μ is a tuple $(\mu_1, \mu_2, \ldots, \mu_k)$ such that for all $p \in \mathcal{P}$ we have that μ_p is a local strategy for p. Further, $\sigma \in \Sigma_p^*$ is an evolution of local strategy μ_p if the following hold:*

1. *If $\sigma' x$ is a prefix of σ for $x \in X_p$ then $\mu_p(\sigma') = x$.*
2. *If σ' is a prefix of σ and $\mu_p(\sigma') = x$ then $\sigma' x$ a prefix of σ.*

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

Given distributed strategy $\mu = (\mu_1, \mu_2, \ldots, \mu_k)$, if σ is a global trace and $\mu_p(\pi_p(\sigma)) = x \in X_p$ then the tester at port p applies input x if it observes $\pi_p(\sigma)$. When

μ is applied, a local tester makes decisions regarding when to supply input on the basis of its observations. In defining the evolutions of μ we need to consider a situation that could not occur with global strategies: we may get a point where more than one local strategy wants to supply the next input and so this can cause a race (see Figure 7 for an example). If the aim is to implement a global strategy then such scenarios are undesirable since there can be more than one possible next input after a trace (global strategies do not allow such situations to occur). As a result, we define what it means for a distributed strategy to be deterministic (to not have such races). The following requires that if trace σ can occur in testing using distributed strategy $\mu = (\mu_1, \mu_2, \ldots, \mu_k)$ (condition 1) then at most one μ_p can supply input after σ; later we will prove that we can map a controllable global strategy to a deterministic distributed strategy.

DEFINITION 4.8. *Given state set S', a distributed strategy $\mu = (\mu_1, \mu_2, \ldots, \mu_k)$ is* deterministic *for S' if there does not exist $\sigma \in L_M(S')$ such that the following hold:*

1. *for all $p \in \mathcal{P}$ we have that $\pi_p(\sigma) \in pre(Ev(\mu_p))$; and*
2. *there exist $p, p' \in \mathcal{P}$, $p \neq p'$, such that $\mu_p(\pi_p(\sigma)) \in X_p$ and $\mu_{p'}(\pi_{p'}(\sigma)) \in X_{p'}$.*

Further, μ is deterministic *if there does not exist a trace $\sigma \in (X/Y)^*$, such that the following hold:*

1. *for all $p \in \mathcal{P}$ we have that $\pi_p(\sigma) \in pre(Ev(\mu_p))$; and*
2. *there exists $p, p' \in \mathcal{P}$, $p \neq p'$, such that $\mu_p(\pi_p(\sigma)) \in X_p$ and $\mu_{p'}(\pi_{p'}(\sigma)) \in X_{p'}$.*

Consequently, we cannot have a trace σ that can occur when applying μ (one where all the projections of σ are prefixes of evolutions of the local strategies) after which two different testers can supply the next input. Thus, since the μ_p are functions, any next input after a global trace σ is uniquely defined. It is straightforward to define evolutions of deterministic distributed strategies and this will allow us to reason about the observations that can be made when a distributed strategy interacts with the SUT.

DEFINITION 4.9. *Trace $\sigma \in (X/Y)^*$ is an* evolution *of a deterministic distributed strategy $\mu = (\mu_1, \mu_2, \ldots, \mu_k)$ if the following hold:*

1. *If $\sigma' x/y$ is a prefix of σ for $x \in X_p$ then $\mu_p(\pi_p(\sigma')) = x$; and*
2. *If σ' is a prefix of σ and $\mu_p(\pi_p(\sigma')) = x$, $x \in X_p$, then there exists $y \in Y$ such that $\sigma' x/y$ is a prefix of σ.*

We let $Ev(\mu)$ denote the set of evolutions of μ.

This can be extended to the case where we have a set $S' \subseteq S$ of states from which a deterministic distributed strategy might be applied.

DEFINITION 4.10. *Given state set $S' \subseteq S$ of M and distributed strategy $\mu = (\mu_1, \mu_2, \ldots, \mu_k)$ that is deterministic for S', trace $\sigma \in (X/Y)^*$ is an* evolution *of μ from S' if $\sigma \in L_M(S')$ and the following hold:*

1. *If $\sigma' x/y$ is a prefix of σ for $x \in X_p$ then $\mu_p(\pi_p(\sigma')) = x$; and*
2. *If σ' is a prefix of σ and $\mu_p(\pi_p(\sigma')) = x$, $x \in X_p$, then there exists $y \in Y$ such that $\sigma' x/y$ is a prefix of σ.*

We let $Ev(\mu, M, S')$ denote the set of such evolutions of μ from S'.

We now consider the problem of mapping a global strategy μ to a distributed strategy. Given port p we can define the projection of μ at p and, in an abuse of notation, call this $\pi_p(\mu)$. The essential idea is that if μ supplies input x at port p after σ ($\mu(\sigma) = x$) then $\mu_p = \pi_p(\mu)$ can supply x after the observation $\pi_p(\sigma)$ it makes in this scenario. We initially define $\pi_p(\mu)$ to be a *relation* between Σ_p^* and $X_p \cup \{\varepsilon\}$: for some $\sigma_p \in \Sigma_p^*$ we may have that $\pi_p(\mu)$ maps σ_p to more than one element of $X_p \cup \{\varepsilon\}$.

DEFINITION 4.11. *Given a global strategy μ and port $p \in \mathcal{P}$, $\pi_p(\mu)$ is a relation μ_p between Σ_p^* and $X_p \cup \{\varepsilon\}$ defined by: $x \in \mu_p(\sigma_p)$ for $x \in X_p \cup \{\varepsilon\}$ and $\sigma_p \in \Sigma_p^*$ if and only if there exists some $\sigma \in Ev(\mu)$ such that $x = \mu(\sigma)$ and $\pi_p(\sigma) = \sigma_p$.*

We have the important property that the projections of a controllable global strategy form a deterministic distributed strategy and so are suitable for use in distributed testing.

PROPOSITION 3. If global strategy μ is controllable for S then the distributed strategy $(\pi_1(\mu), \pi_2(\mu), \ldots.\ \pi_k(\mu))$ is deterministic.

This tells us that if we have a controllable global strategy μ then the set of evolutions is defined for the distributed strategy $(\pi_1(\mu), \pi_2(\mu), \ldots, \pi_k(\mu))$ since evolutions are defined for deterministic distributed strategies.

When considering a set $S' \subseteq S$ of states we restrict attention to traces in $L_M(S')$. As a result, one might expect that if global strategy μ is controllable for S' then the distributed strategy $(\pi_1(\mu), \pi_2(\mu) \ldots \pi_k(\mu))$ is deterministic for S'. However, this is not necessarily the case since in forming the local strategies we consider all evolutions of a global strategy, not only those allowed from S'.

PROPOSITION 4. It is possible that a global strategy μ is controllable for $S' \subseteq S$ but the distributed strategy $(\pi_1(\mu), \pi_2(\mu), \ldots \pi_k(\mu))$ is not deterministic for S'.

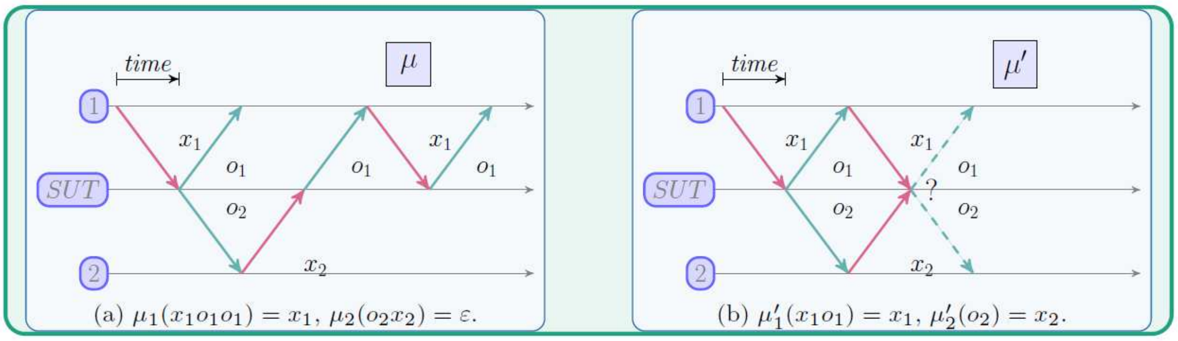

time
μ
1
SUT
2
x_1
o_1
o_2
o_1
x_1
o_1
x_2
(a) $\mu_1(x_1o_1o_1) = x_1$, $\mu_2(o_2x_2) = \varepsilon$.
time
μ'
1
SUT
2
x_1
o_1
o_2
x_1
?
o_1
o_2
x_2
(b) $\mu'_1(x_1o_1) = x_1$, $\mu'_2(o_2) = x_2$.

FIGURE 7: Two distributed strategies: $\mu = (\mu_1, \mu_2)$ and $\mu' = (\mu'_1, \mu'_2)$. Local strategies μ_1 and μ_2 are deterministic. Local strategies μ'_1 and μ'_2 require testers to supply input at the same time, causing nondeterminism.

This leads us to define the projection of a global strategy in the presence of a set $S' \subseteq S$ of states; we simply restrict attention to evolutions of μ that can occur from S'.

DEFINITION 4.12. *Given global strategy μ, port $p \in \mathcal{P}$ and set S' of states of M, $\pi_p^{S'}(\mu)$ is a relation μ_p between Σ_p^* and $X_p \cup \{\varepsilon\}$ defined by: $x \in \mu_p(\sigma_p)$ for $x \in X_p \cup \{\varepsilon\}$ and $\sigma_p \in \Sigma_p^*$ if and only if there exists $\sigma \in Ev(\mu, M, S')$ such that $x = \mu(\sigma)$ and $\pi_p(\sigma) = \sigma_p$.*

We can now generalise Proposition 3 for the case where we have a set $S' \subseteq S$ of states.

PROPOSITION 5. Given state set $S' \subseteq S$ of M, if global strategy μ is controllable for S' then the distributed strategy $(\pi_1^{S'}(\mu), \pi_2^{S'}(\mu), \ldots \pi_k^{S'}(\mu))$ is deterministic for S'.

We have shown that the projections of a controllable global strategy define a deterministic distributed strategy. If we take the projections of a controllable global strategy then the resultant distributed strategy the same set of evolutions and so we can implement a controllable global strategy using distributed testers.

PROPOSITION 6. Let us suppose that μ is a controllable global strategy and for all $p \in \mathcal{P}$ we have that $\mu_p = \pi_p(\mu)$. Then the distributed strategy $\mu' = (\mu_1, \mu_2, \ldots, \mu_k)$ is such that $Ev(\mu) = Ev(\mu')$.

PROPOSITION 7. Let us suppose that μ is a controllable global strategy for set $S' \subseteq S$ of states of FSM M and for all $p \in \mathcal{P}$ we have that $\mu_p = \pi_p^{S'}(\mu)$. Then the distributed strategy $\mu' = (\mu_1, \mu_2, \ldots, \mu_k)$ is such that $Ev(\mu, M, S') = Ev(\mu', M, S')$.

We now define what it means for a deterministic distributed strategy to be an ADS.

DEFINITION 4.13. *A distributed strategy μ is an* adaptive distinguishing sequence *for state set S' of FSM M, $S' \subseteq S$, if μ is deterministic for S' and for all $s, s' \in S'$ with $s' \neq s$, $\sigma \in Ev(\mu, M, s)$, and $\sigma' \in Ev(\mu, M, s')$ we have that $\sigma' \not\sim \sigma$. Further, μ is an* adaptive distinguishing sequence *for M if μ is an adaptive distinguishing sequence for S.*

We now have the main result in this section, which is that if μ is a controllable ADS for M then the distributed strategy obtained by taking the projections of μ is also an ADS. This allows us to construct global controllable strategies that are ADSs, knowing that we can take their projections to form suitable distributed strategies. The rest of the paper will therefore focus on generating suitable global strategies.

THEOREM 4.1. *If μ is controllable and is an adaptive distinguishing sequence for FSM M then $\mu' = (\pi_1(\mu), \pi_2(\mu), \ldots, \pi_k(\mu))$ is an adaptive distinguishing sequence for M.*

THEOREM 4.2. *Given $S' \subseteq S$, if μ is controllable for S' and is an adaptive distinguishing sequence for S' then $\mu' = (\pi_1^{S'}(\mu), \pi_2^{S'}(\mu), \ldots, \pi_k^{S'}(\mu))$ is an adaptive distinguishing sequence for M from S'.*

We now provide an algorithm (Algorithm 3) that receives a global strategy μ and a port p and returns the projection of the global strategy at p. Recall that a global strategy is a partial function from traces to inputs. In forming the local strategy μ_p we extract the evolutions of μ since μ is defined on its evolutions and their prefixes. Let us suppose that σ' is a prefix of an evolution σ of μ. There are two cases to consider. If $\mu(\sigma') = x$ and $x \in X_P$ then μ_p should apply x after $\pi_p(\sigma')$ and so we add to μ_p the pair $(\pi_p(\sigma'), \mu(\sigma'))$ that represents μ_p mapping $\pi_p(\sigma')$ to x (Line 6). Alternatively, if $\mu(\sigma') = x$ and $x \notin X_P$ then μ_p should not supply input after $\pi_p(\sigma')$ and so we add the pair $(\pi_p(\sigma'), \varepsilon)$ that represents μ_p mapping $\pi_p(\sigma')$ to ε (and so μ_p does not supply input after $\pi_p(\sigma')$) (Line 7).

For example consider the global ADS strategy given in Figure 6b. For Port 1, Algorithm 3 returns μ_1 such that $Ev(\mu_1, M, s_1) = x_1o_1o_1x_1o'_1$ and $Ev(\mu_1, M, s_6) = x_1o'_1x_1o'_1x_1o'_1$ etc.

These results are important since they tell us that as long as we restrict attention to controllable strategies, it is safe to generate global strategies that are ADSs and

ALGORITHM 3: Generating a projection of a global controllable strategy μ on a given port $p \in \mathcal{P}$.

```
Input: $\mu$, $p \in \mathcal{P}$
Output: The projection of $\mu$ on given port $p$
begin
    $\mu_p \leftarrow \emptyset$
    foreach $\sigma \in Ev(\mu, M, S)$ do
        Let $\sigma$ is the sequence $x_1/y_1, x_2/y_2, \dots, x_\ell/y_\ell$
        foreach $1 \leq k \leq \ell$ do
            Let $\sigma_k$ is the sequence $x_1/y_1, x_2/y_2, \dots, x_k/y_k$
            if $\mu(\sigma_k) \in X_p$ then
                $\mu_p = \mu_p \cup \{(\pi_p(\sigma_k), \mu(\sigma_k))\}$
            else
                $\mu_p = \mu_p \cup \{(\pi_p(\sigma_k), \varepsilon)\}$
    Return $\mu_p$
```

then take their projections. If a global strategy μ is not controllable then there must be a port p and traces σ and σ' such that $\pi_p(\sigma) = \pi_p(\sigma')$, $\mu(\sigma) \in X_p$ and $\mu(\sigma') \neq \mu(\sigma)$. As a result, the set of projections of μ do not define a distributed strategy in which the local testers are deterministic: $\pi_p(\mu)$ must relate $\pi_p(\sigma)$ to more than one element of $X_p \cup \{\varepsilon\}$. Thus, a local tester cannot determine what to do next based on its observations and, in addition, μ and $(\pi_1(\mu), \pi_2(\mu) \dots, \pi_k(\mu))$ will have different sets of evolutions. As a result, we know that it is safe to consider controllable global strategies and also that if a global strategy μ is not controllable then we cannot implement it using distributed testers (using the resultant distributed testers can lead to a race and so behaviours not allowed under μ).

5. GENERATING A CONTROLLABLE ADS

In Section 4, we showed that when using the distributed test architecture it is sufficient to consider global controllable ADSs. We now show that the problem of deciding whether an FSM M has a global controllable ADS that distinguishes all of its states is PSPACE-Hard.

We will rely on the following lemma that tells us that the problem of deciding whether a single-port FSM has a PDS is PSPACE-hard even if all transitions have non-empty output.

Lemma 5.1. *Given a single-port FSM M in which no transition produces empty output, checking the existence of a preset distinguishing sequence is* PSPACE-Complete.

Theorem 5.1. *Given an FSM M, checking the existence of a controllable ADS that distinguishes all of the states of M is* PSPACE-Hard. *In addition, this result still holds if we restrict attention to FSMs that have two ports.*

Clearly, not every FSM has a global controllable ADS, therefore as discussed for single-port FSMs [55], under such circumstances one may wish to construct global controllable ADSs for subset of states. However, we show that this is also a hard problem.

Theorem 5.2. *The following problem is* PSPACE-Hard*: given an FSM M, find a controllable* ADS *μ and state set S' where μ is a controllable* ADS *for S' and μ and S' are such that S' has maximal size.*

By the virtue of the reduction used in the proof of the above, one can deduce that there are FSMs such that the shortest evolution[8] in $Ev(\mu, M, S)$ is of exponential length.

Theorem 5.3. *There is a class of FSMs that have* ADSs *such that the shortest evolution is of exponential length.*

Finally, since existence is PSPACE-Hard so are the corresponding optimisation problems.

Theorem 5.4. *The following problem is* PSPACE-Hard*: given an FSM M, what is the smallest value of ℓ such that M has an* ADS *of height ℓ?*

These results show that ADS generation problems are computationally hard, in contrast to the situation with single-port FSMs, but leave open the question of whether these problems are decidable. We now show that we can reduce these negative results to some extent when we limit the height of the ADS and we are therefore interested in the following problem.

Definition 5.1. *Given an FSM M and natural number ℓ, the* Exact Height *problem for M is to determine whether M has a controllable* ADS *with height ℓ.*

Naturally, this corresponds also to the problem of deciding whether there is a controllable ADS with height at most ℓ and it is straightforward to adapt the results in this section to the case where we have a bound on the height of an ADS. We therefore focus on the exact height problem. This problem is motivated by the fact that it is possible to use a set

[8]Definition 4.2 defines the notion of an evolution

of separating sequences instead of an ADS or PDS in order to identify states. It is known that for any two states s_i and s_j of an FSM M with n states, k ports and m inputs, we can decide in $O(mn^2)$ time whether there is a controllable separating sequence that distinguishes s_i and s_j and if so there is such a sequence of length at most $k(n-1)$. Thus, one can construct a set of controllable separating sequences to form a characterisation set of polynomial size and this can be achieved in polynomial time.

We will show that the Directed Hamiltonian Path (DHP) problem for strongly connected directed graphs, which is NP-Complete [56] [57], is polynomial time reducible to the exact height problem. An instance of a DHP problem can be defined as follows, where a walk is said to *visit* a vertex v if v is the starting vertex or the ending vertex of at least one edge in the walk.

Definition 5.2. *Consider a strongly connected directed graph $G = (V, E)$ with vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and edge set $E = \{e_1, e_2, \ldots, e_m\}$. We say that walk ρ of G is a Hamiltonian path if and only if the walk visits each vertex of G exactly once. The DHP problem is to decide whether a strongly connected directed graph G has a Hamiltonian path.*

Given an instance G of the DHP problem, we will construct an FSM $M(G) = (\mathcal{P}, S, s_0, X, Y, \delta, \lambda)$. The aim will be to construct the transition structure of the FSM in such a way that an ADS μ simulates the rules that govern the DHP problem. We will then prove that if G has n vertices then there is an ADS μ for M whose longest evolution in $Ev(\mu, M, S)$ is of length $n-1$ if and only if the corresponding sequence of symbols constitutes a solution to the DHP problem for G. We now show how we construct $M(G)$.

Definition 5.3. *We represent vertex v_i of G by a state s_i of M and add an additional state s_e and so $S = \{s_1, s_2, \ldots, s_n\} \cup \{s_e\}$. For each edge e_i of G there is a corresponding port i of M and unique input x_i at port i. There is an extra port 0, the port set of M is $\mathcal{P} = \{0\} \cup \{1, 2, \ldots, m\}$. There are no inputs at port 0. The output alphabets are: $Y_0 = \{1, 2, \ldots, n\}$ and for all $1 \le i \le m$, $Y_i = \{o_i\}$.*

If e_i is an edge from vertex v_j to vertex v_r then the state changes associated with transitions of $M(G)$ that have input x_i are defined by the following:

1. *We include in $M(G)$ a transition from s_j to s_r with input x_i.*
2. *From every state $s \in S$ with $s \neq s_j$ there is a self-loop transition from s with input x_i. These are included to make $M(G)$ completely specified and all transitions from s_e are of this form.*

Thus, for each state s of $M(G)$, a walk in G has a corresponding walk in $M(G)$ that starts at s. We define the output in response to input x_i in order to ensure that in controllable testing x_i can only be followed by an input $x_j \neq x_i$ if e_i can be followed by e_j in G. As a result, controllable walks through $M(G)$ correspond to walks of G. Let us suppose that e_i is an edge from vertex v_j to vertex v_r and in G the edges that leave v_r (and so can follow e_i) are are those in $E' \subseteq E$. Then for port p, $1 \le p \le m$, the transitions with input x_i produce output o_p at p if $e_p \in E'$ and otherwise they produce no output at p. As a result, input x_i can be followed by input $x_j \neq x_i$ in controllable testing if and only if e_i can be followed by e_j in G. At port 0 there are two cases: x_i leads to output j at 0 if x_i is input when M is in state s_j (recall that e_i leaves vertex v_j) and otherwise, if x_i is input when $M(G)$ is in a state $s_{j'} \neq s_j$, it leads to no output at 0.

As a result of this construction, any input of x_i leads to the same output at all ports in $\{1, 2, \ldots, m\}$ irrespective of the state in which it is applied. Thus, only the output at port 0 can be used to distinguish states. In addition, no output can be produced at 0 when an input sequence is applied from s_e. For example consider an instance G of the DHP problem given in Figure 8a and corresponding FSM $M(G)$ given in Figure 8b.

In terms of distinguishing states there is no value in following an input immediately by itself (e.g. having a subsequence of the form $x_i x_i$). We will say that a strategy for $M(G)$ is *non-redundant* if it cannot lead to an input being immediately followed by itself and results will restrict attention to non-redundant strategies.

A controllable global strategy of $M(G)$ has a particular form.

Proposition 8. Given a directed graph G and FSM $M(G)$ with state set S, if μ is a non-redundant controllable global strategy for $M(G)$ then all traces in $Ev(\mu, M(G), S \setminus \{s_e\})$ have the same input portion $x_{i_1} x_{i_2} \ldots x_{i_r}$ and this has the property that $e_{i_1} e_{i_2} \ldots e_{i_r}$ is a walk of G.

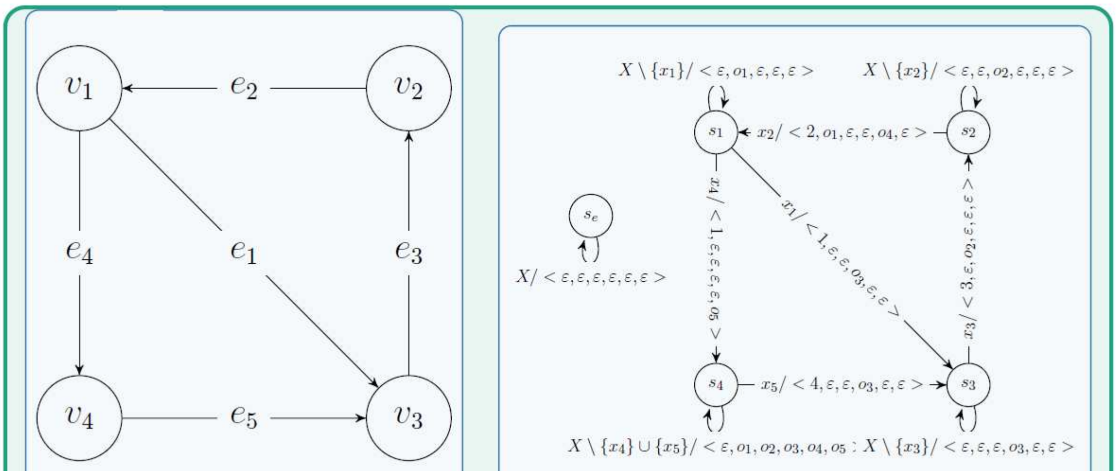
We now show how the DHP problem for strongly connected G relates to the existence of an ADS μ for $M(G)$ whose longest evolution has length $\ell = n$.

Proposition 9. Strongly connected directed graph G has a Hamiltonian path if and only if $M(G)$ has a controllable ADS that distinguishes all of the state of $M(G)$ and whose longest evolution has length $\ell = n$.

We can now prove that the problem of deciding whether an FSM has an ADS whose longest evolution has length ℓ is NP-Hard. The following holds when we are interested in distinguishing all states in S or some subset S' of these.

Theorem 5.5. *The exact height problem is in EXPSPACE and is NP-Hard.*

Finally, if we suitably bound ℓ then the problem is NP-Complete.


v_1
v_2
v_3
v_4
e_1
e_2
e_3
e_4
e_5
$X \setminus \{x_1\}/ < \varepsilon, o_1, \varepsilon, \varepsilon, \varepsilon >$
$X \setminus \{x_2\}/ < \varepsilon, \varepsilon, o_2, \varepsilon, \varepsilon, \varepsilon >$
s_1
$x_2/ < 2, o_1, \varepsilon, \varepsilon, o_4, \varepsilon >$
s_2
s_e
$X/ < \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon >$
$x_4/ < 1, \varepsilon, \varepsilon, \varepsilon, \varepsilon, o_5 >$
$x_1/ < 1, \varepsilon, \varepsilon, o_3, \varepsilon, \varepsilon >$
$x_3/ < 3, \varepsilon, o_2, \varepsilon, \varepsilon, \varepsilon >$
s_4
$x_5/ < 4, \varepsilon, \varepsilon, o_3, \varepsilon, \varepsilon >$
s_3
$X \setminus \{x_4\} \cup \{x_5\}/ < \varepsilon, o_1, o_2, o_3, o_4, o_5$
$X \setminus \{x_3\}/ < \varepsilon, \varepsilon, \varepsilon, o_3, \varepsilon, \varepsilon >$

(a) Original DHP problem instance G and the specified vertex is v_4.

(b) Constructed FSM $M(G)$ at which the specified vertices are s_4, s_e.

FIGURE 8: Construction of an FSM from a given DHP problem instance.

Theorem 5.6. *The exact height* ADS *problem for an FSM M is* NP-Complete *if ℓ is defined by a polynomial in term of the size of M.*

As noted before, in testing it is possible to use a characterisation set containing controllable separating sequences. Thus, in practice we are unlikely to be interested in ADSs that are significantly longer than the sum of the lengths of the sequences in such a characterisation set. This motivates the above result: in practice we are likely to have a polynomial upper bound on the height of an ADS that we are ready to use.

6. CONCLUSIONS

Many automated test case generation algorithms for testing from a single-port FSM M use test cases that distinguish states of M and there has been particular interest in adaptive distinguishing sequences (ADSs). There has been interest in ADSs as they can be shorter, are computationally less expensive to produce, and there are FSMs that have ADSs but no PDS. This has led to the development of many automated test case generation algorithms for FSMs that use ADSs. However, such algorithms return test cases that are designed for the case where there is a single tester that interacts with the SUT and so need not be suitable for distributed testing.

This paper has extended the concepts of ADSs to distributed testing. We showed that if an ADS is controllable then it can be implemented by a set of distributed local testers but otherwise this is not possible. It also demonstrated how these local testers can be devised.

We then considered complexity related problems for ADSs. We proved that the problem of deciding whether an FSM has an ADS is PSPACE-Hard but left decidability open. This situation is significantly different from the single-port FSM problem, which can be solved in low-order polynomial time. We also showed that the problem of deciding whether there is an ADS with height ℓ (or height at most ℓ) is NP-Hard. In practice we are likely to have upper bound ℓ that is a polynomial in the size of M and for this case the problem is NP-Complete.

There are several lines of future work. First, it is still open whether the ADS existence problem is decidable. There is also the problem of finding realistic conditions under which the problems studied can be solved in polynomial time. There may also be scope to develop heuristics for devising controllable ADSs and finally, there is the potential to develop new automated test case generation algorithms for distributed testing.

APPENDIX

Proposition 2. If a strategy μ is controllable then for every FSM M and set S' of states of M, we have that μ is controllable for S'. However, it is possible that strategy μ is controllable for S' for some FSM M and state set S' and yet μ is not controllable.

Proof. The first part is immediate from the definitions. For the second part consider a global strategy μ that:

- starts by supplying input x_1 at port 1;
- if $\langle o_1, \varepsilon \rangle$ is output then the strategy supplies input x_2 at port 2 and terminates;
- for every other output the strategy terminates.

This strategy is not controllable since it should send input x_2 to port 2 after $x_1/\langle o_1, \varepsilon \rangle$ but not after the empty sequence ε, even though $x_1/\langle o_1, \varepsilon \rangle$ and ε have the same projections at port 2. However, μ is controllable for any S' from which the input of x_1 cannot lead to output $\langle o_1, \varepsilon \rangle$. The result thus follows. □

Proposition 3. If global strategy μ is controllable for S then the distributed strategy $(\pi_1(\mu), \pi_2(\mu), \ldots.\ \pi_k(\mu))$ is deterministic.

Proof. We are required to prove that $\mu' = (\pi_1(\mu), \pi_2(\mu) \ldots.\pi_k(\mu))$ is deterministic. We will use proof by contradiction and assume that μ' is non-deterministic. By Definition 4.8 there therefore exists global trace $\sigma_1 \in (X/Y)^*$ such that for all $p \in \mathcal{P}$ we have that $\pi_p(\sigma_1) \in Ev(\mu_p)$ and ports p, p' with $p \neq p'$ such that $\mu_p(\pi_p(\sigma_1)) \in X_p$ and $\mu_{p'}(\pi_{p'}(\sigma_1)) \in X_{p'}$. By Definition 4.11, there exist global traces σ and σ' in $Ev(\mu)$ such that:

- $\pi_p(\sigma) = \pi_p(\sigma_1)$ and $\mu(\sigma) = x$, $x \in X_p$; and
- $\pi_{p'}(\sigma') = \pi_{p'}(\sigma_1)$ and $\mu(\sigma') = x'$, $x' \in X_{p'}$.

By Definition 4.3, since μ is controllable and $\pi_p(\sigma) = \pi_p(\sigma_1)$ we must have that μ supplies input x after σ_1. Similarly, by Definition 4.3, since $\pi_{p'}(\sigma') = \pi_{p'}(\sigma_1)$ we must have that μ supplies input x' after σ_1. This contradicts the definition of a global strategy, since μ applies two different inputs after σ_1, and so the result follows. □

Proposition 4. It is possible that a global strategy μ is controllable for $S' \subseteq S$ but the distributed strategy $(\pi_1(\mu), \pi_2(\mu), \ldots \pi_k(\mu))$ is not deterministic for S'.

Proof. Consider a global strategy μ that initially supplies input x_1 at port 1 and only supplies another input if the output is $\langle 1, \varepsilon \rangle$, in which case the input is x_2 at port 2. Further let S' be some set of states from which the input of x_1 cannot lead to $\langle 1, \varepsilon \rangle$. Then clearly μ is controllable for S' since from S' it simply supplies x_1, observes an output, and then terminates. However, if we take the projections we find that $\pi_1(\mu)$ starts by supplying input x_1 and $\pi_2(\mu)$ can initially supply input x_2 (since μ can supply x_2 after $x_1/\langle 1, \varepsilon \rangle$ and $\pi_2(x_1/\langle 1, \varepsilon \rangle) = \varepsilon$). Thus, the distributed strategy $(\pi_1(\mu), \pi_2(\mu))$ is not deterministic for S' as required. □

Proposition 6. Let us suppose that μ is a controllable global strategy and for all $p \in \mathcal{P}$ we have that $\mu_p = \pi_p(\mu)$. Then the distributed strategy $\mu' = (\mu_1, \mu_2, \ldots, \mu_k)$ is such that $Ev(\mu) = Ev(\mu')$.

Proof. First we prove that $Ev(\mu) \subseteq Ev(\mu')$. Proof by contradiction: assume that there is some $\sigma \in Ev(\mu) \setminus Ev(\mu')$. Let σ' denote the longest prefix of σ that is in $pre(Ev(\mu'))$ and so there exists an input/output pair x/y such that $\sigma' x/y$ is a prefix of σ and $\sigma' x/y \notin pre(Ev(\mu'))$. Let p be such that $x \in X_p$. Since $\sigma' x/y \in Ev(\mu)$ we have that $\mu(\sigma') = x$ and so $\mu_p(\pi_p(\sigma')) = x$. Thus, μ' can supply input x after σ' and so $\sigma' x/y \in pre(Ev(\mu'))$, providing a contradiction as required.

Now we prove that $Ev(\mu') \subseteq Ev(\mu)$. Proof by contradiction: assume that there is some $\sigma \in Ev(\mu') \setminus Ev(\mu)$. Let σ' denote the longest prefix of σ that is in $pre(Ev(\mu))$ and so there exists an input/output pair x/y such that $\sigma' x/y$ is a prefix of σ and $\sigma' x/y \notin pre(Ev(\mu))$. Let p be such that $x \in X_p$. Since $\sigma' x/y \in Ev(\mu')$ we have that $\mu_p(\pi_p(\sigma')) = x$. Thus, by the definition of $\pi_p(\mu)$, there exists some $\sigma'' \in Ev(\mu, M, S'))$ such that $\pi_p(\sigma'') = \pi_p(\sigma')$ and $\mu(\sigma'') = x$. However, since $\pi_p(\sigma'') = \pi_p(\sigma')$ and μ is controllable we must have that $\mu(\sigma') = \mu(\sigma'')$. Thus, $\mu(\sigma') = x$ and so $\sigma' x/y \in pre(Ev(\mu))$, providing a contradiction as required. □

Theorem 4.1. *If μ is controllable and is an adaptive distinguishing sequence for FSM M then $\mu' = (\pi_1(\mu), \pi_2(\mu), \ldots, \pi_k(\mu))$ is an adaptive distinguishing sequence for M.*

Proof. By Proposition 3, since μ is controllable we know that μ' is deterministic. In addition, by Proposition 6 we know that $Ev(\mu) = Ev(\mu')$ and so for every state s of M we have that $Ev(\mu, M, s) = Ev(\mu', M, s)$. The result now follows from μ being an adaptive distinguishing sequence for M. □

Theorem 4.2. *Given $S' \subseteq S$, if μ is controllable for S' and is an adaptive distinguishing sequence for S' then $\mu' = (\pi_1^{S'}(\mu), \pi_2^{S'}(\mu), \ldots, \pi_k^{S'}(\mu))$ is an adaptive distinguishing sequence for M from S'.*

Proof. By Proposition 5, since μ is controllable for S' we know that μ' is deterministic for S'. In addition, by Proposition 7, for every state s of M we have that $Ev(\mu, M, s) = Ev(\mu', M, s)$. The result now follows from μ being an adaptive distinguishing sequence for S'. □

Lemma 5.1. *Given a single-port FSM M in which no transition produces empty output, checking the existence of a preset distinguishing sequence is* PSPACE-Complete.

Proof. The problem being in PSPACE is a consequence of the general PDS existence problem for single-port FSMs being in PSPACE and so we focus on proving that the problem is PSPACE-Hard. We will show that any algorithm that can solve this problem can also solve the general problem of deciding whether a single-port FSM has a distinguishing sequence. Let $M = (S, s_0, X, Y, \delta, \lambda)$ be a single-port FSM in which some transitions may have output ε. We construct an FSM $M' = (S, s_0, X, Y \cup \{y\}, \delta, \lambda')$ where $y \notin Y$ is a new output and the function λ' is defined by: given $s \in S$ and $x \in X$, if $\lambda(s, x) \neq \varepsilon$ then $\lambda'(s, x) = \lambda(s, x)$ and otherwise $\lambda'(s, x) = y$. It is now sufficient to observe that an input sequence is a distinguishing sequence for M if and only if it is a distinguishing sequence for M'. The result now follows from the problem of deciding whether a single-port FSM has a distinguishing sequence being PSPACE-complete [29]. □

Theorem 5.1. *Given an FSM M, checking the existence of a controllable ADS that distinguishes all* *of the states of M is* PSPACE-Hard. *In addition, this result still holds if we restrict attention to FSMs that have two ports.*

Proof. Assume that we have been given a single-port FSM $M_1 = (S, s_0, X, Y, \delta, \lambda)$ such that all of the transitions of M_1 have non-empty output. We will construct an FSM M that has two ports 1 and 2. The state set of M will be S and the initial state will be s_0. Port 1 will have input alphabet $X_1 = X$ and output alphabet $Y_1 = \emptyset$. Port 2 will have input alphabet $X_2 = \emptyset$ and output alphabet $Y_2 = Y$. Given state s and input x such that $\delta(s,x) = s'$ and $\lambda(s,x) = y$, we will include in M the transition from s to s' that has input $x \in X_1$ and produces output $\langle \varepsilon, y \rangle$.

Now consider controllable global adaptive strategy for M. Since all inputs are at port 1 and no outputs are produced at port 1, there is no opportunity for a controllable global adaptive strategy to lead to different input sequences from different states: the tester choosing the next input will have observed no output irrespective of the state that the adaptive strategy was applied in. Thus, all controllable adaptive strategies for M correspond to fixed input sequences. In addition, since every transition produces non-empty output at port 2 and no output at port 1, if the tester applies an input sequence $x_1, x_2, \ldots, x_m$ at port 1 and the tester at port 2 observes output sequence $y_1, y_2, \ldots, y_m$ then we know that for all $1 \leq i \leq m$, y_i was produced in response to input x_i. Thus, there are no observability problems.

To summarise, a controllable global adaptive strategy for M corresponds to a fixed input sequence $\bar{x}$ and the output sequence observed at port 2 when $\bar{x}$ is applied from state $s \in S$ is exactly $\lambda(s, \bar{x})$. Thus, an adaptive strategy for M is a controllable ADS for M if and only if it corresponds to an input sequence $\bar{x}$ such that for all $s, s' \in S$ with $s \neq s'$ we have that $\lambda(s, \bar{x}) \neq \lambda(s', \bar{x})$. This is the case if and only if $\bar{x}$ is a distinguishing sequence for M_1. The result now follows from Lemma 5.1. □

Theorem 5.2. *The following problem is* PSPACE-Hard*: given an FSM M, find a controllable* ADS *μ and state set S' where μ is a controllable* ADS *for S' and μ and S' are such that S' has maximal size.*

Proof. If we have an algorithm that solves this problem and are given FSM M, then M has a controllable ADS if and only if the algorithm returns such an ADS. The result thus follows from Theorem 5.1. □

Theorem 5.3. *There is a class of FSMs that contain* ADS *such that the shortest evolution is of exponential length.*

Proof. Consider a single-port FSM that has a PDS with exponential length [29], now reapply the reduction given in Theorem 5.1. □

Theorem 5.4. *The following problem is* PSPACE-Hard*: given an FSM M, what is the smallest value of ℓ such that M has an* ADS *of height ℓ?*

Proof. An FSM has an ADS if and only if it has a minimum height ADS. Thus, any algorithm that returns the smallest ℓ such that M has an ADS of height ℓ also decides whether M has an ADS. The result thus follows from the existence problems being PSPACE-Hard. □

Proposition 8. Given a directed graph G and FSM $M(G)$ with state set S, if μ is a non-redundant controllable global strategy for $M(G)$ then all traces in $Ev(\mu, M(G), S \setminus \{s_e\})$ have the same input portion $x_{i_1} x_{i_2} \ldots x_{i_r}$ and this has the property that $e_{i_1} e_{i_2} \ldots e_{i_r}$ is a walk of G.

Proof. First observe that all transitions of $M(G)$ with input x_i, $1 \leq i \leq m$, produce the same output at all ports of $M(G)$ except 0. In addition, $M(G)$ has no inputs at port 0. We will prove that the input portions are the same for all traces in $Ev(\mu, M(G), S \setminus \{s_e\})$ and will use proof by contradiction: assume that the input portions of $Ev(\mu, M(G), s)$ and $Ev(\mu, M(G), s')$ are different for some states $s, s' \in S \setminus \{s_e\}$. Let $\bar{x}$ denote the longest common prefix of the input portions of $Ev(\mu, M(G), s)$ and $Ev(\mu, M(G), s')$. Without loss of generality, assume that $Ev(\mu, M(G), s)$ has an input portion that follows $\bar{x}$ with input x_p at port p. However, since $M(G)$ has no input at port 0 we have that $p \neq 0$ and so the responses to $\bar{x}$ in states s and s' have the same outputs at p. Thus, since μ is controllable, $Ev(\mu, M, s')$ must have an input portion that follows $\bar{x}$ with input x_p. However, this contradicts the definition of $\bar{x}$ as required. Thus, all traces in $Ev(\mu, M(G), S \setminus \{s_e\})$ have the same input portion $x_{i_1} x_{i_2} \ldots x_{i_r}$. Further, by the definition of $M(G)$, in a non-redundant controllable global strategy an input x_i can only be followed by input x_j if in G we have that e_i can be followed by e_j. The result therefore follows. □

Proposition 9. Strongly connected directed graph G has a Hamiltonian path if and only if $M(G)$ has a controllable ADS that distinguishes all of the state of $M(G)$ and whose longest evolution has length $\ell = n$.

Proof. First we prove that if G has a Hamiltonian path $\rho = e_1 e_2 \ldots e_{n-1}$ then $M(G)$ has an ADS whose longest evolution has length n. Choose an edge e_n of G that can follow e_{n-1} in G: since G is strongly connected there must be some such edge. By the definition of $M(G)$, the input sequence $x_1 x_2 \ldots x_n$ defines a controllable global strategy for $M(G)$. In addition, since ρ is a Hamiltonian path, for every state s_i of $M(G)$, $s_i \neq s_e$, the application of input sequence $x_1 x_2 \ldots x_n$ from s_i includes an input that corresponds to an edge with starting vertex v_i and so leads to an output sequence at port 0 that starts with i. Finally, the application of $x_1 x_2 \ldots x_n$ in state s_e leads to no output being produced at 0. Thus, $x_1x_2 \dots x_n$ defines an ADS and its longest evolution has length n as required.

Now we assume that $M(G)$ has a controllable ADS whose longest evolution has length $\ell = n$ and we are required to prove that G has a Hamiltonian path. By Proposition 8 we know that there is some input sequence $x_1x_2 \dots x_n$ such that all traces of $Ev(\mu, M(G), S \setminus \{s_e\})$ have input portion $x_1x_2 \dots x_n$. Further, since μ is an ADS for $M(G)$ we must have that for every state $s_i \neq s_e$, $x_1x_2 \dots x_n$ contains an input x_j such that v_i is the starting vertex of e_j. In addition, since μ is controllable we must have that $e_1e_2 \dots e_n$ is a walk of G. To conclude, all vertices of G start edges in walk $e_1e_2 \dots e_n$ of G and so $e_1e_2 \dots e_{n-1}$ is a Hamiltonian path of G. □

Theorem 5.5. *The exact height problem is in* EXPSPACE *and is* NP-Hard.

Proof. We will first show that a non-deterministic Turning machine T can decide the exact height ADS problem using exponential space. We can allow T to initially guess an ADS μ with height at most ℓ. Since this defines a finite tree with at most n leaves there is an upper bound on the size of the tree that is polynomial in terms of n and ℓ and so this take space that is polynomial in ℓ and n.

In order to check whether μ is controllable it is sufficient to compute the traces that can be produced by applying μ from states of M and for any two traces σ and σ' check whether there are corresponding controllability problems. There are corresponding controllability problems if there are prefixes σ_1 and σ'_1 of σ and σ' respectively that have the same projection at a port p such that after σ and σ' the behaviour of the tester at p differs. Thus, the Turing machine can check this in polynomial time. Finally, the Turing machine can check in polynomial time whether μ distinguishes the states of M. The Turing machine takes space that is polynomial in n and ℓ and so exponential in the description of the problem (since ℓ can be described in $O(\log_2 \ell)$ space). Thus, we have that a non-deterministic Turing machine can solve the problem in exponential space. Finally, using the Savitch's Theorem [58] we know that a deterministic Turing machine can also solve the problem in exponential space. We therefore have that the problem is in EXPSPACE.

The problem being NP-Hard follows from Proposition 9 and the fact that the DHP problem with strongly connected directed graphs is NP-Hard. □

REFERENCES

[1] Barnett, M., Grieskamp, W., Nachmanson, L., Schulte, W., Tillmann, N., and Veanes, M. (2003) Towards a tool environment for model-based testing with AsmL. *Formal Approaches to Software Testing*, Montreal, Canada, October 6th, Lecture Notes in Computer Science, **2931**, pp. 252–266. Springer-Verlag.

[2] Cartaxo, E. G., Machado, P. D. L., and Neto, F. G. O. (2011) On the use of a similarity function for test case selection in the context of model-based testing. *Software Testing, Verification and Reliability*, **21**, 75–100.

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

[4] Garousi, V., Briand, L. C., and Labiche, Y. (2008) Traffic-aware stress testing of distributed real-time systems based on UML models using genetic algorithms. *Journal of Systems & Software*, **81**, 161–185.

[5] Grieskamp, W., Kicillof, N., Stobie, K., and Braberman, V. A. (2011) Model-based quality assurance of protocol documentation: tools and methodology. *Software Testing, Verification and Reliability*, **21**, 55–71.

[6] Pickin, S., Jard, C., Jeron, T., Jezequel, J.-M., and Le Traon, Y. (2007) Test synthesis from UML models of distributed software. *IEEE Transactions on Software Engineering*, **33**, 252–269.

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

[8] Tretmans, J. (2008) Model based testing with labelled transition systems. In Hierons, R. M., Bowen, J. P., and Harman, M. (eds.), *Formal Methods & Testing, An Outcome of the FORTEST Network, Revised Selected Papers*, Lecture Notes in Computer Science, Springer, Verlag.

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

[10] Hennie, F. C. (1964) Fault-detecting experiments for sequential circuits. *Proceedings of Fifth Annual Symposium on Switching Circuit Theory & Logical Design*, Princeton, New Jersey, November 11-13, pp. 95–110., The inst. of elec. and elect. eng.

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

[12] Moore, E. P. (1956) Gedanken-experiments. In Shannon, C. and McCarthy, J. (eds.), *Automata Studies*, **34**, pp. 129–153.

[13] Petrenko, A. and Yevtushenko, N. (2005) Testing from partial deterministic FSM specifications. *IEEE Transactions on Computers*, **54**, 1154–1165.

[14] Walter, T., Schieferdecker, I., and Grabowski, J. (1998) Test architectures for distributed systems: state of the art & beyond. *Testing of Communicating Systems IFIP*, Italy, Trento, September 11, **3**, pp. 149–174. Springer US.

[15] Sunyé, G., Cunha De Almeida, E., Le Traon, Y., Baudry, B., and Jézéquel, J.-M. (2014) Model-Based Testing of Global Properties on Large-Scale Distributed Systems. *Information & Software Technology* , **56-7**, Pages 749–762.

[16] Ernits, J. P., Roo, R., Jacky, J., and Veanes, M. (2009) Model-based testing of web applications using NModel. *TestCom/FATES*, Eindhoven, The Netherlands, November 2-4 pp. 211–216.

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

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

[19] Chen, J., Hierons, R. M., and Ural, H. (2004) Conditions for resolving observability problems in distributed testing. *24rd IFIP International Conference on Formal Techniques for Networked & Distributed Systems (FORTE 2004)*, Madrid, Spain, September 27-30, Lecture Notes in Computer Science, **3235**, pp. 229–242. Springer-Verlag.

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

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

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

[23] Guyot, S. and Ural, H. (1995) Synchronizable checking sequences based on UIO sequences. *Protocol Test Systems, VIII*, Evry, France, September 4-6, pp. 385–397. Chapman & Hall.

[24] Hierons, R. M. and Ural, H. (2008) The effect of the distributed test architecture on the power of testing. *The Computer Journal*, **51**, 497–510.

[25] Hierons, R. M. (2010) Reaching and distinguishing states of distributed systems. *SIAM Journal on Computing*, **39**, 3480–3500.

[26] Jourdan, G. V., Ural, H., and Yenigun, H. (2006) Minimizing coordination channels in distributed testing. *26th IFIP International Conference on Formal Techniques for Networked & Distributed Systems (FORTE 2006)*, France, September 26-29, Lecture Notes in Computer Science, **4229**, pp. 451–466. Springer-Verlag.

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

[28] Dorofeeva, R., El-Fakih, K., Maag, S., Cavalli, A. R., and Yevtushenko, N. (2010) FSM-based conformance testing methods: A survey annotated with experimental evaluation. *Information & Software Technology*, **52**, 1286–1297.

[29] Lee, D. and Yannakakis, M. (1994) Testing finite-state machines: State identification and verification. *IEEE Transactions on Computers*, **43**, 306–320.

[30] Hierons, R. M. and Ural, H. (2008) Checking sequences for distributed test architectures. *Distributed Computing*, **21**, 223–238.

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

[32] de Almeida, E. C., Marynowski, J. E., Suny, G., Traon, Y. L., and Valduriez, P. (2010) Efficient distributed test architectures for large-scale systems. *Testing Software & Systems - 22nd IFIP WG 6.1 International Conference, ICTSS 2010*, Natal, Brazil, November 8-10, Lecture Notes in Computer Science, **6435**, pp. 174–187. Springer, Verlag.

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

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

[35] Lucas, C., Elbaum, S., and Rosenblum, D. S. (2012) Detecting problematic message sequences and frequencies in distributed systems. *SIGPLAN Not.*, **47**, 915–926.

[36] Hierons, R. M., Merayo, M., and Nunez, M. (2008) Controllable test cases for the distributed test architecture. *Automated Technology for Verification & Analysis*, Lecture Notes in Computer Science, Seoul, Korea, October 20-23 , pp. 201–215. Springer Berlin / Heidelberg.

[37] Hierons, R. M. (2011) Controllable testing from nondeterministic finite state machines with multiple ports. *IEEE Transactions on Computers*, **60**, 1818 –1822.

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

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

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

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

[42] Young, Y. C. and Tai, K. C. (1998) Observational inaccuracy in conformance testing with multiple testers. *IEEE 1st workshop on application-specific software engineering and technology*, March 26-28, 1998, Clarion Hotel and University of Texas at Dallas, Richardson, Texas, pp. 80–85., IEEE Computer Society Press

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

[44] Hierons, R. M. (–) Generating complete controllable test suites for distributed testing. *IEEE Transactions on Software Engineering*, DOI 10.1109/TSE.2014.2364035, In Press.

[45] Chen, T. Y., Leung, H., and Mak, I. K. (2004) Adaptive random testing. *9th Asian Computing Science Conference*, Chiang Mai, China, December 14–16, Lecture Notes in Computer Science, **3321**, pp. 320–329. Springer.

[46] Ciupa, I., Leitner, A., Oriol, M., and Meyer, B. (2008) ARTOO: adaptive random testing for object-oriented software. *30th International Conference on Software Engineering (ICSE 2008)*, Leipzig, Germany, May 10 - 18, pp. 71–80. ACM.

[47] Chan, K. P., Chen, T. Y., and Towey, D. (2002) Restricted random testing. *7th International Conference on Software Quality (ECSQ 2002)*, Portland, Oregon, USA, October 11 - 12, Lecture Notes in Computer Science, **2349**, pp. 321–330. Springer.

[48] Chan, K. P., Chen, T. Y., and Towey, D. (2006) Restricted random testing: Adaptive random testing by exclusion. *International Journal of Software Engineering and Knowledge Engineering*, **16**, 553–584.

[49] Lv, J., Hu, H., Cai, K., and Chen, T. Y. (2014) Adaptive and random partition software testing. *IEEE Transactions on Systems, Manufacturing, and Cybernetics: Systems*, **44**, 1649–1664.

[50] Haar, S., Jard, C., and Jourdan, G.-V. (2007) Testing input/output partial order automata. *19th IFIP TC 6/WG 6.1 International Conference on Testing of Software & Communicating Systems (TestCom/FATES)*, Tallin, Estonia, June 26-29, Lecture Notes in Computer Science, **4581**, pp. 171–185. Springer, Verlag.

[51] v. Bochmann, G., Haar, S., Jard, C., and Jourdan, G.-V. (2008) Testing systems specified as partial order input/output automata. *20th IFIP TC 6/WG 6.1 International Conference on Testing of Software & Communicating Systems (TestCom/FATES)*, Tokyo, Japan, June 10-13, Lecture Notes in Computer Science, **5047**, pp. 169–183. Springer, Verlag.

[52] 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 & Computation*, pp. 189–196. Academic Press.

[53] Alur, R., Courcoubetis, C., and Yannakakis, M. (1995) Distinguishing tests for nondeterministic and probabilistic machines. *27th ACM Symposium on Theory of Computing*, Las Vegas, Nevada, May 29-June 1, pp. 363–372. ACM New York, NY.

[54] Hierons, R. M., Merayo, M. G., and Núñez, M. (2008) Controllable test cases for the distributed test architecture. *6th International Symposium on Automated Technology for Verification & Analysis (ATVA 2008)*, Seoul, Korea, October 20-23, pp. 201–215, Lecture Notes in Computer Science, Springer-Verlag.

[55] Hierons, R. M. and Türker, U. C. (2015) Incomplete distinguishing sequences for finite state machines. *The Computer Journal*, doi: 10.1093/comjnl/bxv041, In Press.

[56] Garey, M. R. and Johnson, D. S. (1979) *Computers & Intractability.* W. H. Freeman and Company, New York.

[57] Karp, R. M. (1972) Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W. (eds.), *Complexity of Computer Computations.* Plenum Press, New York-London. 85–103.

[58] Savitch, W. J. (1970) Relationships between non-deterministic and deterministic tape complexities. *Journal of Computer & System Sciences*, **4**, 177 – 192.