Distinguishing Sequences for Distributed Testing: Preset 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

There has been long-standing interest in automatically generating test sequences from a finite state machine (FSM) and more recently this has been extended to the case where there are multiple physically distributed testers and so we are testing from a multi-port FSM. This paper explores the problem of generating a controllable preset distinguishing sequence (PDS) from a multi-port FSM, motivated by the fact that many FSM-based test generation algorithms use PDSs. We prove that it is generally undecidable whether a multi-port FSM has a controllable PDS but provide a class of multi-port FSMs for which the problem is decidable. We also consider the important case where there is an upper bound ℓ on the length of PDSs of interest, proving that controllable PDS existence is PSPACE-hard and in EXPSPACE. In practice the upper bound ℓ is likely to be a polynomial in terms of the size of the multi-port FSM and in this case controllable PDS existence is NP-Complete.

Keywords: Finite state machine, distributed test architecture, preset distinguishing sequence

Received 00 January 2009; revised 00 Month 2009

1. INTRODUCTION

Testing is an important part of the software development process but is typically manual, expensive, and error prone. This has led to significant interest in automating parts of testing, with many approaches being examples of model-based testing (MBT) in which automation is based on a model. Many MBT methods take as input a finite state machine (FSM) (see, for example, [1–5]) or input output transition system (IOTS) (see, for example, [6, 7]). The FSM or IOTS that is used in test automation might have been produced by a tool that analyses the semantics of a model, allowing testers to use more expressive languages such as state charts and SDL [3, 8, 9]. This paper focuses on the problem of generating test sequences from an FSM. This problem was initially described in the seminal paper by Moore [4] in 1956 and in 1964 Hennie [2] provided the first FSM-based test generation algorithm that can be automated.

Traditionally, software testing has been seen as a process in which the system under test (SUT) and tester interact synchronously. However, this does not reflect how many systems interact with their environment and, in particular, a distributed system might interact with its environment at a number of physically distributed locations (ports). In such a situation we might have a separate local tester at each port. If the local testers do not interact with one another during testing and there is no global clock then testing is distributed and we are testing in the (ISO standardised [10]) distributed test architecture. There has been significant interest in distributed testing (see, for example, [11–20]), with the initial motivation being protocol conformance testing. In this case an implementation N of a layer of the protocol stack is tested through having one local tester acting as the layer above N and another local tester sitting on a different machine [14, 15, 20].

A separate line of research has explored the situation in which the local testers can synchronise their actions by communicating through a network (see, for example, [21–24]). There are situations in which such an approach is entirely appropriate and one can then use traditional FSM-based test sequence generation algorithms. However, it may not be possible to synchronise the testers if there are timing constraints[3]. In addition, the exchange of messages between the local testers either requires an external network to be established, which can increase the cost of testing, or uses the same network as the SUT and so can change

[3]Although our models do not include time, we might use them to test the functional aspects of a system that has timing constraints such as timeouts.

the behaviour of the SUT.

The initial work on distributed testing found that there can be additional *controllability problems* in which a local tester cannot determine when to supply an input since it only observes the events at its port [14, 20]. Consider, for example, the interaction shown in Figure 1a. In this the tester at port 1 should send input x_1 and as a result we expect o_1 to be output at port 1. The tester at port 2 should then send input x_2. However, the tester at port 2 does not observe the previous interactions, which were at port 1, and so does not know when to send the input. Controllability problems are typically seen as being undesirable since the tester cannot know the order in which inputs are received: this makes it difficult to determine whether a test objective was achieved and also to trace failures to requirements. The focus of test generation has thus been on techniques that return *controllable* test sequences; test sequences that do not cause controllability problems [11, 13, 16, 25–28]. Note also that recent work showed that it is undecidable whether there is a test case that is guaranteed to distinguish two states s and s' of an FSM [18] but this problem can be solved in low-order polynomial time if we restrict attention to controllable test sequences [17].

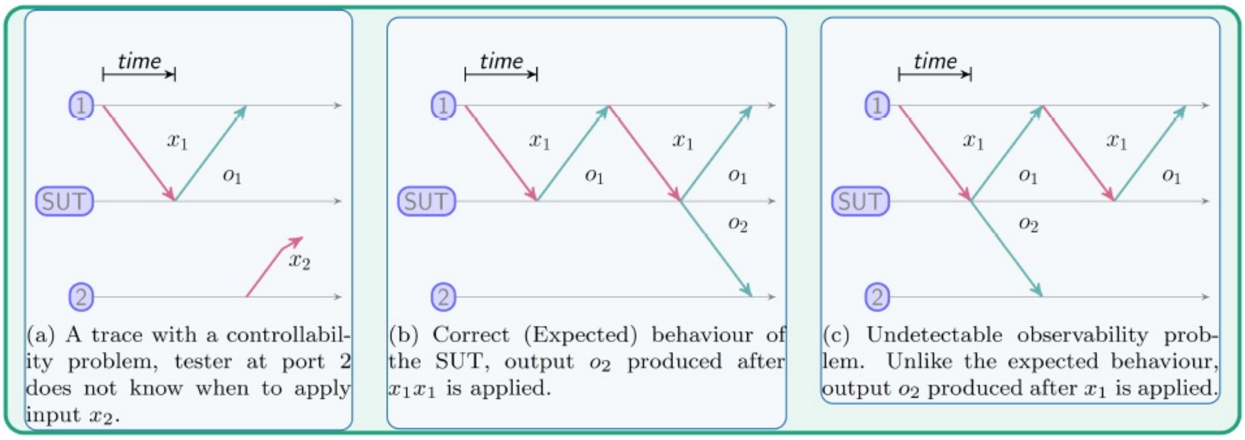
In distributed testing a local tester observes the events at its ports and so a projection of the global trace (sequence of inputs and outputs) that occurred. Thus, the overall observation is a set of local traces rather than a global trace. Since a set of local traces need not uniquely define the global trace that occurred, there are additional *observability problems* in distributed testing [15]. To see this, consider the interaction given in Figure 1b. In the specification the input of x_1 at port 1 should lead to output o_1 at port 1 and if we apply x_1 again then o_1 should be output at port 1 and o_2 should be output at port 2. The expected global trace is therefore $x_1/\langle o_1, \varepsilon\rangle x_1/\langle o_1, o_2\rangle$ in which ε denotes null output at a particular port. As a result, the tester at port 1 should observe $x_1o_1x_1o_1$ and the tester at port 2 should observe o_2. These local observations are made if instead the SUT produces $x_1/\langle o_1, o_2\rangle\, x_1/\langle o_1, \varepsilon\rangle$ (Figure 1c) despite this global trace not being allowed by the specification. Observability problems can reduce test effectiveness and so there has been interest in producing test sequences that do not suffer from observability problems [12, 15, 29–31].

Most FSM-based test generation techniques use sequences that distinguish the states of the FSM M from which test sequences are being generated (see, for example, [1, 2, 32–36]). It has been found that distinguishing sequences, where they exist, lead to shorter tests [37]. There are two types of distinguishing sequences: adaptive distinguishing sequences (ADSs) and preset distinguishing sequences (PDSs). A PDS for state set S' is an input sequence that leads to different outputs from all of the states in S'. An ADS for S' also leads to different outputs from all of the states in S' but is adaptive: instead of being a fixed input sequences, it is like a decision tree, the next input to be applied being decided on the basis of the output observed. Hierons proved that when the FSM has multiple ports, it is undecidable whether a set S' of states has an ADS or a PDS and this is the case even when we restrict attention to sets containing only two states [33]. Despite these negative results, recently Hierons and Türker investigated the problem of deriving *controllable* ADSs from a multi-port FSM. Although they did not provide an answer to the question of whether constructing an ADS is decidable, they proved that ADS existence is PSPACE-hard [39]. In contrast, ADS existence can be decided in polynomial time for single-port FSMs [40].

In this paper, we concentrate on deriving controllable PDSs from multi-port FSMs. One motivation is that many FSM-based test sequence generation techniques use PDSs (see, for example, [2, 34, 41–44]); if we can devise approaches that generate controllable PDSs then there is the potential to extend these test generation techniques to distributed testing. In addition, there is a test sequence generation algorithm for multi-port FSMs that uses PDSs [45].

This paper makes the following contributions. First, we define what it means for an input sequence to be a controllable PDS for an FSM M or some subset of its set of states. We prove that (controllable) PDS existence is undecidable in general but we also provide a class of FSMs for which it is decidable. We also prove that if we have an upper bound ℓ on the length of PDSs in which we are interested then PDS existence is decidable, PSPACE-hard, and in EXPSPACE. In practice, instead of using a PDS we might generate a set of controllable input sequences that pairwise distinguish the states of the FSM M and such input sequences can be generated in low-order polynomial time [17]. Thus, in practice we might expect an upper bound ℓ to be a polynomial in terms of the size of the FSM M and we prove that controllable PDS existence is then NP-Complete. As far as we are aware there is no corresponding result for single-port FSMs but it is not hard to show that the general bounded case is PSPACE-Complete for such FSMs. Finally, we give a class of FSMs for which controllable PDS existence is decidable.

This paper is structured as follows. We provide preliminary material in Section 2 and also define controllable PDSs. In Section 3 we then prove that PDS existence is undecidable in general and provide the complexity results for the bounded cases. Section 4 then considers a special class of FSM, which we call C-FSMs, and proves that controllable PDS existence is decidable for this. Finally, in Section 5 we conclude and discuss potential future work.



(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 1: Controllability and observability problems.

2. PRELIMINARIES

In this section we explore the formalisation used, provide definitions of concepts we require, and define what it means for an input sequence to be a controllable PDS for an FSM M. We will let $\mathcal{P}$ denote the set of ports; for all $p \in \mathcal{P}$ there is a local tester at port p. We will use the term FSM to denote a multi-port finite state machine (FSM): an FSM in which there are multiple ports. We will use the term single-port FSM for classical FSMs. We now give standard definitions of FSMs and associated notation (see, for example, [39]).

DEFINITION 2.1. *An FSM is defined by a tuple* $M = (\mathcal{P}, S, s_0, X, Y, \delta, \lambda)$ *where:*

- $\mathcal{P} = \{1, 2, \ldots, k\}$ *is the set of ports.*
- S *is the finite set of states and* $s_0 \in S$ *is the initial state. We let* n *denote the number of states.*
- X *is the finite set of inputs and* $X = X_1 \cup X_2 \cup \cdots \cup X_k$ *where* X_p $(1 \leq p \leq k)$ *is the input alphabet for port* p. *We assume that the input alphabets of the ports are disjoint: for all* $p, p' \in \mathcal{P}$, *such that* $p \neq p'$, *we have* $X_p \cap X_{p'} = \emptyset$. *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. *We assume that the output alphabets of the ports are disjoint: for two ports* $p, p' \in \mathcal{P}$, *such that* $p \neq p'$, *we have* $Y_p \cap Y_{p'} = \emptyset$. *An output* $y \in Y$ *is a vector* $\langle o_1, o_2, \ldots, o_k \rangle$ *where* $o_p \in Y_p \cup \{\varepsilon\}$ *for all* $1 \leq p \leq k$. *We also assume that* X *is disjoint from* $\cup_{1 \leq p \leq k} Y_p$. *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* $S \times X \rightarrow S$. *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* $\lambda : S \times X \rightarrow Y$ *gives the output vector produced in response to an input.*

An FSM processes inputs one at a time but is able to produce more than one output in a transitions (at most one output for each port). We will use the following terminology in which $M = (\mathcal{P}, S, s_0, X, Y, \delta, \lambda)$.

DEFINITION 2.2. *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.*

We will use directed graphs to represent FSMs. As an example, Figure 2 describes a 2-port FSM M that has port set $\{1, 2\}$, state set $\{s_1, s_2, s_3, s_4\}$, 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 of the FSM and a directed edge between two nodes represents a transition: an edge with label x/y from a node that has label s to a node that has label s' represents the transition $(s, s', x/y)$.

We use ε to represent the empty sequence and juxtaposition to denote concatenation. Thus, for example, xx' represents a sequence of length 2 whose first element is x and whose second element is x'. The output and state transfer functions can be extended to input sequences in the usual way: if $x \in X$ and $\bar{x} \in X^*$ then $\delta(s, \varepsilon) = s$, $\delta(s, x\bar{x}) = \delta(\delta(s, x), \bar{x})$, $\lambda(s, \varepsilon) = \varepsilon$, and $\lambda(s, x\bar{x}) = \lambda(s, x)\lambda(\delta(s, x), \bar{x})$. We now define some standard terminology.

DEFINITION 2.3. *A sequence of transitions* $\rho =$

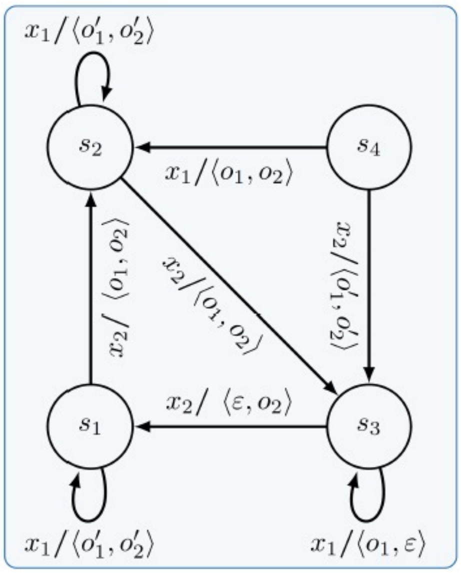

$x_1/\langle o_1', o_2' \rangle$
s_2
s_4
$x_1/\langle o_1, o_2 \rangle$
$x_2/\langle o_1, o_2 \rangle$
$x_2/\langle o_1, o_2 \rangle$
$x_2/\langle o_1', o_2' \rangle$
$x_2/\ \langle \varepsilon, o_2 \rangle$
s_1
s_3
$x_1/\langle o_1', o_2' \rangle$
$x_1/\langle o_1, \varepsilon \rangle$

FIGURE 2: Example FSM M

$(s_1, s_2, x_1/y_1)(s_2, s_3, x_2/y_2)\dots(s_m, s_{m+1}, x_m/y_m)$ *is a* walk *that has* starting state s_1, ending state s_{m+1}, *and* label $x_1/y_2\ x_2/y_2\dots x_m/y_m$. *Here* $x_1/y_2x_2/y_2\dots x_m/y_m$ *is an* input output sequence, *also called a* trace, *whose* input portion *is* $\bar{x} = x_1\dots x_m$ *and whose* output portion *is* $\bar{y} = y_1\dots y_m$. *We also use* $\bar{x}/\bar{y}$ *to represent this trace. An FSM* M *defines the language* $L(M)$ *of traces that label walks with starting state* s_0 *and this is the behaviour of* M. *Thus,* $L(M) = \{\bar{x}/\lambda(s_0, \bar{x})|\bar{x} \in X^*\}$ *and* $L_M(s)$ *denotes the set of labels of walks of* M *with starting state* s. *Given* $S' \subseteq S$, *we let* $L_M(S') = \cup_{s\in S'} L_M(s)$ *denote the set of labels of walks of* M *that have starting state in* S'. *States* s, s' *are* equivalent *if* $L_M(s) = L_M(s')$ *and FSMs* M *and* N *are* equivalent *if* $L(M) = L(N)$. *FSM* M *is* minimal *if there is no FSM that is equivalent to* M *and has fewer states. Further,* M *is* strongly connected *if for all* $(s, s') \in S \times S$ *we have that there is a walk with starting state* s *and ending state* s'. *If* M *is strongly connected then* M *is minimal if and only if* $L_M(s) \neq L_M(s')$ *for all* $s, s' \in S$ *with* $s \neq s'$. *We use pre to denote a function that takes a set of sequences and returns the set of prefixes of these sequences. If* $x_1/y_1\ x_2/y_2\dots x_m/y_m$ *is a trace then its prefixes are of the form* $x_1/y_1\ x_2/y_2\dots x_i/y_i$ *for* $i \leq m$.

If we consider the FSM M given in Figure 2 then $(s_3, s_1, x_2/\langle\varepsilon, o_2\rangle)(s_1, s_2, x_2/\langle o_1, o_2\rangle)$ is a walk that has starting state s_3 and ending state s_2. The label of this walk is $x_2/\langle\varepsilon, o_2\rangle\ x_2/\langle o_1, o_2\rangle$ and this 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 $x_2/\langle o_1, o_2\rangle\ x_1/\langle o_1, o_2\rangle$, which can also be represented as $x_2x_1/\langle o_1, o_2\rangle\langle o_1, o_2\rangle$, and $L_M(s_3)$ contains the global trace $x_2/\langle\varepsilon, o_2\rangle\ x_2/\langle o_1, o_2\rangle$. As normal, we restrict attention to minimal FSMs but this is not a significant restriction since it is possible to convert an FSM into an equivalent minimal FSM in low-order polynomial time [46].

We will sometimes use the term *global trace* to denote a trace in order to distinguish this from the local traces observed by the (local) testers. Since the ports are distributed, no tester observes a global trace: the tester at port p only observes the inputs and outputs at p. If σ is a global trace, then we use $\pi_p(\sigma)$ to denote the *local trace* at p: a sequence of inputs and outputs at port p (the projection of σ at p).

DEFINITION 2.4. *Given port* p, *function* π_p *is defined by the following rules.*
$\pi_p(\varepsilon) = \varepsilon$
$\pi_p((x/\langle o_1, o_2, \dots, o_k\rangle)\sigma) = \pi_p(\sigma)$ *if* $x \notin X_p \wedge o_p = \varepsilon$
$\pi_p((x/\langle o_1, o_2, \dots, o_k\rangle)\sigma) = x\pi_p(\sigma)$ *if* $x \in X_p \wedge o_p = \varepsilon$
$\pi_p((x/\langle o_1, o_2, \dots, o_k\rangle)\sigma) = o_p\pi_p(\sigma)$ *if* $x \notin X_p \wedge o_p \neq \varepsilon$
$\pi_p((x/\langle o_1, o_2, \dots, o_k\rangle)\sigma) = xo_p\pi_p(\sigma)$ *if* $x \in X_p \wedge o_p \neq \varepsilon$

A local tester only observes the local projections of the global trace that occurred and so the set of local testers can only distinguish two global traces if there is a port p such that the local projections at p differ.

DEFINITION 2.5. *Global traces* σ_1, σ_2 *are* indistinguishable, *written* $\sigma_1 \sim \sigma_2$, *if for all* $p \in \mathcal{P}$ *we have that* $\pi_p(\sigma_1) = \pi_p(\sigma_2)$.

Consider, for instance, global traces $\sigma_1 = x_2/\langle o_1, o_2\rangle x_2/\langle\varepsilon, o_2\rangle$ and $\sigma_2 = x_2/\langle\varepsilon, o_2\rangle x_2/\langle o_1, o_2\rangle$. We have that $\pi_1(\sigma_1) = o_1$, $\pi_2(\sigma_1) = x_2o_2x_2o_2$, $\pi_1(\sigma_2) = o_1$ and $\pi_2(\sigma_2) = x_2o_2x_2o_2$ and, as a result, $\sigma_1 \sim \sigma_2$. We will use $|.|$ to denote the cardinality of a set or the length of a sequence and so, for example, in the above we have that $|\mathcal{P}| = 2$ and $|\pi_1(\sigma_1)| = 4$.

We now define what it means for an input sequence to be controllable [47]. Essentially, an input sequence is controllable if each local tester knows when to send its inputs. Since testing is distributed, a local tester only observes the corresponding local trace and so must be able to decide when to send its inputs based on this local trace. Now let us suppose that we have input sequence $\bar{x} = x_1\dots x_m$ and consider the condition under which the tester at port p knows when to send $x_{j+1} \in X_p$. We are interested in distinguishing states so the local tester may not know the state from which $\bar{x}$ is applied. As a result, if we set $\bar{x}' = x_1\dots x_j$ then the local observation, before x_{j+1}, should be $\pi_p(\lambda(s, \bar{x}'))$ for some $s \in S$. Thus, we require that the observation of $\pi_p(\lambda(s, \bar{x}'))$ could not have been made if some different prefix $\bar{x}'' = x_1\dots x_i$ of $\bar{x}$ had been applied from a state s'. Further, we require this condition to hold for all of the states from which $\bar{x}$ might be applied and for all $1 \leq j < m$. The following formalises the above intuition.

DEFINITION 2.6. *Input sequence* $\bar{x} = x_1\dots x_m$ *is* controllable *for state set* S' *of FSM* M *if for all* $s, s' \in S'$ *and distinct prefixes* $\bar{x}', \bar{x}''$ *of* $\bar{x}$ *we have that if* $\bar{x}' = x_1\dots x_j$ *for* $1 \leq j < m$ *and* $x_{j+1} \in X_p$ *for port* p *then* $\pi_p(\lambda(s, \bar{x}')) \neq \pi_p(\lambda(s', \bar{x}''))$. *Further,* $\bar{x}$ *is* controllable for M *if it is controllable for state set* S.

When an input sequence $\bar{x}$ is applied to an FSM, the FSM produces outputs. If we are testing from a single-port FSM and $\bar{x}$ leads to different outputs when applied from different states of M, then $\bar{x}$ can be used to identify the states of M. State identification is important, not only for distinguishing the states of an M, but also for deriving test sequences from M. There are a number of approaches to distinguishing states of a single–port FSM and it has been shown that *distinguishing sequences* (DSs) have the advantage of leading to shorter test sequences [37].

There are two types of DSs, adaptive distinguishing sequences (ADSs) and preset distinguishing sequences (PDSs). An ADS can be seen as being a decision tree where the choice of next input depends on the output that has been observed. Recently Hierons and Türker investigate the problem of deriving controllable ADSs from a multi-port FSM. They proved that controllable ADS existence is PSPACE-Hard [39] but left decidability open.

For single-port FSMs a PDS is an input sequence that leads to a different output sequence for each state: the tester applies this input sequence and observes the resultant output sequence. One of the benefits of using a PDS is that testing need not be adaptive, with this allowing the use of a simpler test infrastructure. While many test tools support adaptive testing, the additional computation required in adaptive testing can be problematic when there are timing constraints (see, for example, [48]). To obtain such benefits in distributed testing we require that the *local* testers do not have to be adaptive.

Example 1. Consider the input sequence $\bar{x} = x_1x_1x_2$, in which x_1 is input at port 1 and x_2 is input at port 2. Further, let us suppose that $\bar{x}$ is applied from a state set $S' = \{s_1, s_2, s_3\}$ such that from s_1 we should obtain the trace $\sigma_1 = x_1/\langle o_1, -\rangle\, x_1/\langle o_1, o_2\rangle\, x_2/\langle o_1, -\rangle$ (Figure 3a), from s_2 we should obtain the trace $\sigma_2 = x_1/\langle o_1, -\rangle\, x_1/\langle o_1, o_2\rangle\, x_2/\langle o_1', -\rangle$ (Figure 3b) and from s_3 we should obtain the trace $\sigma_3 = x_1/\langle o_1, o_2'\rangle x_1/\langle o_1, o_2'\rangle\, x_2/\langle o_1, -\rangle$ (Figure 3c). Here we have that in σ_1 and σ_2 the tester at port 2 applies input x_2 after observing o_2 and in σ_3 the tester at port 2 applies input x_2 after observing $o_2'o_2'$. Thus, although $x_1x_1x_2$ is a fixed input sequence that causes no controllability problems for S' and distinguishes the states from S', its application requires the tester at port 2 to be adaptive and thus it cannot be applied using local testers that are not adaptive.

In order for a controllable global input sequence $\bar{x}$ to not require the local testers to be adaptive, given port p we will require that the observations made at p before an input x at p are identical for all states in S': the tester at p thus simply waits for this local trace to be observed before applying x.

Definition 2.7. *A global input sequence $\bar{x}$ that is controllable for set S' of states of FSM M is a* controllable PDS *for S' if and only if the following hold:*

1. *Given states $s, s' \in S'$ with $s \neq s'$, if σ and σ' are the global traces that result from applying $\bar{x}$ in states s and s' respectively then $\sigma \not\sim \sigma'$.*
2. *Given states $s, s' \in S'$ with $s \neq s'$, if σ and σ' are the global traces that result from applying $\bar{x}$ in states s and s' respectively then for all ports p we have that the longest prefixes of $\pi_p(\sigma)$ and $\pi_p(\sigma')$ that end in an input are identical.*

When this holds, each local tester follows a fixed pattern until its last input has been supplied and then it simply observes any further output. As a result, the local testers do not have to be adaptive. We now consider the problem of generating such controllable PDSs.

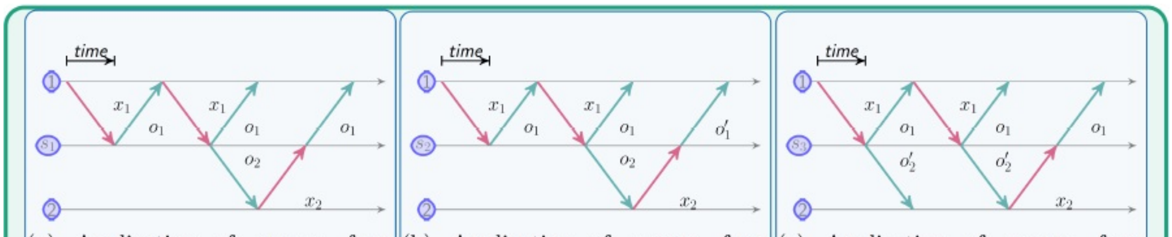
3. GENERATING A CONTROLLABLE PDS

In this section we explore the problem of deciding whether an FSM has a controllable PDS and investigate problems associated with controllable PDSs. Initially we prove that the general problem is undecidable, by relating it to the undecidable Post Correspondence Problem. In practice, there are likely to be upper bounds on the length of a PDS that is useful. We thus also explore bounded PDS existence; the upper bound naturally makes the problem decidable.

It is known that the problem of determining whether there is an input sequence that distinguishes two states is undecidable even if we are testing from a deterministic FSM [18]. However, this work did not restrict attention to controllable input sequences and, indeed, it is possible to determine in polynomial time whether there is a controllable input sequence that distinguishes two states of a deterministic FSM [17]. We consider controllable PDSs and show that the existence of a controllable PDS is undecidable. We show this by a reduction from the undecidable *Post Correspondence Problem.*

Definition 3.1. *Post's Correspondence Problem (PCP) is to decide, for sequences $\alpha_1, \alpha_2, \ldots, \alpha_b$ and $\beta_1, \beta_2, \ldots, \beta_b$, whether there is a sequence $a_1a_2\ldots a_j$ of indices in $[1..b]$ such that $\alpha_{a_1}\alpha_{a_2}\ldots\alpha_{a_j} = \beta_{a_1}\beta_{a_2}\ldots\beta_{a_j}$.*

The following takes an instance $\alpha_1, \alpha_2, \ldots, \alpha_b$, $\beta_1, \beta_2, \ldots, \beta_b$ of the PCP and constructs an FSM M where there is a controllable PDS for M if and only if there is a solution to this instance of the PCP. In the construction, m is the maximum of the lengths of the sequences that define this instance of the PCP. The construction includes four pairs $(s_1, s_1'), (s_2, s_2'), (s_3, s_3'), (s_4, s_4')$ of special states and for each $1 \leq i \leq b$ there is a corresponding input x_i. In



(a) Application of $x_1x_1x_2$ from state s_1.
(b) Application of $x_1x_1x_2$ from state s_2.
(c) Application of $x_1x_1x_2$ from state s_3.

FIGURE 3: A preset test sequence that requires adaptive testers.

s_1, s_1', s_2, s_2' the input of x_i a total of m times leads to output sequence α_i; from s_3, s_3', s_4, s_4' we instead get output sequence β_i. These output sequences are sent to port 2. M is constructed so that a PDS must start with an input *init* at port 0 that takes it to one of s_1, s_2, s_3, s_4, must then apply some x_i a total of m times, then apply some x_j m times etc. A PDS must finally apply input x' at port 2. The key point in the proof is that, by the definition of a PDS, the final input of x' can only occur if the sequences observed at port 2 are identical and this is the case if and only if the PDS has applied an input sequence $init(x_{a_1})^m(x_{a_2})^m \dots (x_{a_j})^m$, where x^m denotes x repeated m times, that leads to the same output sequences at port 2 from all of the states. This is the case if and only if $\alpha_{a_1}\alpha_{a_2}\dots\alpha_{a_j} = \beta_{a_1}\beta_{a_2}\dots\beta_{a_j}$; the sequence $a_1a_2\dots a_j$ of indices defines a solution to this instance of the PCP.

Theorem 3.1. *Given FSM M with state set S, checking the existence of a controllable* PDS *for S is undecidable and this holds even if we restrict attention to FSMs with three ports.*

Proof. We will prove this by showing that any algorithm that solves this problem can be used to solve Post's Correspondence Problem. Let us suppose that we have an instance of PCP defined by sequences $\alpha_1, \alpha_2, \dots, \alpha_b$ and $\beta_1, \beta_2, \dots, \beta_b$. We will now define an FSM $\mathcal{M}$ such that there is a controllable PDS for the states of $\mathcal{M}$ if and only if there is a solution to this instance of PCP.

Let m denote the length of the longest sequence in $\alpha_1, \alpha_2, \dots, \alpha_b, \beta_1, \beta_2, \dots, \beta_b$ (ie. $m = max\{|\alpha_1|, |\alpha_2|, \dots, |\alpha_b|, |\beta_1|, |\beta_2|, \dots, |\beta_b|\}$). Within the state set S of $\mathcal{M}$ we will include $S' = \{s_1, s_2, s_3, s_4\}$ and special states $\{s_\star, s_e\}$ and so $S' \cup \{s_\star, s_e\} \subseteq S$.

The FSM has three ports $\mathcal{P} = \{0, 1, 2\}$. The input and output alphabets of port 0 are $X_0 = \{init\}$, $Y_0 = \{\emptyset\} \cup S$ respectively. For every state s_i in $S \setminus S'$ we introduce a transition labelled with $init/ < s_i, start, \varepsilon >$ that ends in s_1. Thus, with input *init* the tester at port 0 can identify the state if the FSM was in a state from $S \setminus S'$. If M receives input *init* when in a state from S' then there is no change in state and output $< \varepsilon, start, \varepsilon >$ is produced.

For all $1 \leq j \leq b$ there is an input x_j at port 1. With input x_j state $s_\star$ produces empty output ($< \varepsilon, \varepsilon, \varepsilon >$) and goes to state s_e. With input x_j state s_e produces empty output and loops. Thus, an input sequence starting with x_j cannot distinguish s_e and $s_\star$ and so a PDS for $\mathcal{M}$ cannot start with an x_j. We will structure transitions labelled by inputs from port 1 such that a sequence of m consecutive inputs of x_j at port 1 leads to output sequence α_j at port 2 from states s_1, s_1' and s_2, s_2' and output sequence β_j at port 2 from states s_3, s_3' and s_4, s_4'. This can be achieved since $m = max\{|\alpha_1|, |\alpha_2|, \dots, |\alpha_b|, |\beta_1|, |\beta_2|, \dots, |\beta_b|\}$. The first such sequence takes states s_1, s_2, s_3, s_4 to s_1', s_2', s_3', s_4' respectively and after that these input sequences lead to cycles. If we only apply such sequences then we do not distinguish s_1 from s_2 and we also do not distinguish s_3 from s_4.

If an input sequence from $\{x_1, x_2, \dots, x_b\}^*$ is not in $(x_1^m + x_2^m + \dots + x_b^m)^*$ (is not a sequence of m instances of some x_{a_1} followed by m instances of some x_{a_2} etc.) then the s_i are all taken to state s_e. It is straightforward to add transitions that achieve this; such transitions are added to the states other than the s_i' in a cycle (from s_i') whose transitions all have label x_j (some x_j). The transitions added all have the same output and this ensures that, from the observation above, if (after *init*) we apply an input sequence in $\{x_1, x_2, \dots, x_b\}^*$ that is not in $(x_1^m + x_2^m + \dots + x_b^m)^*$ then we cannot distinguish s_1 from s_2 and we also cannot distinguish s_3 from s_4.

We have input x' at port 2. From state s_j', $1 \leq j \leq 4$, the input of x' at port 2 leads to output s_j at port 0 and M moves to the state s_e. From all other states input x' takes $\mathcal{M}$ to s_e with no outputs at any port.

Note that a PDS for $\mathcal{M}$ must begin with input *init* at port 0 (since this is the only way of distinguishing s_e and $s_\star$) and this leads to all the states except states in set S' being distinguished at port 0. As a result of the difference in output at port 0, *init* cannot be applied again in a controllable PDS.

From the above we can conclude that a (minimal) controllable PDS must end in input x' and must precede this with an input sequence in the form *init* $x_{a_1}^m x_{a_2}^m \dots x_{a_j}^m$ and the output sequence at port 2 must be the following:

- From states s_1 and s_2 the sequence $start\ \alpha_{a_1}\alpha_{a_2}\dots\alpha_{a_j}$
- From states s_3 and s_4 the sequence

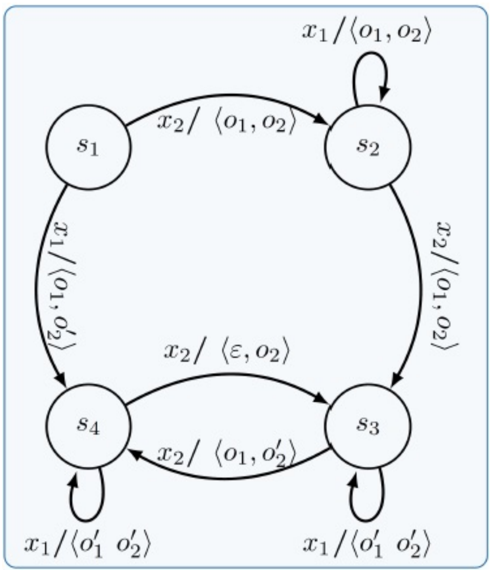

$x_1/\langle o_1, o_2\rangle$
$x_2/\ \langle o_1, o_2\rangle$
s_1
s_2
$x_1/\langle o_1, o_2'\rangle$
$x_2/\langle o_1, o_2\rangle$
$x_2/\ \langle \varepsilon, o_2\rangle$
s_4
s_3
$x_2/\ \langle o_1, o_2'\rangle$
$x_1/\langle o_1'\ o_2'\rangle$
$x_1/\langle o_1'\ o_2'\rangle$

FIGURE 4: Example FSM M

start $\beta_{a_1}\beta_{a_2}\dots\beta_{a_j}$.

This initial input sequence of the form *init* $x^m_{a_1}x^m_{a_2}\dots x^m_{a_j}$ distinguishes all states except those in S'. By the definition of a controllable PDS, we require there to be a common sequence of outputs at port 2 before x' is applied and so we require that *start* $\alpha_{a_1}\alpha_{a_2}\dots\alpha_{a_j}$ = *start* $\beta_{a_1}\beta_{a_2}\dots\beta_{a_j}$. There is thus an input sequence that has the required properties, and so defines a controllable PDS, if and only if there is a solution to the given instance of PCP. The result therefore follows from the PCP being undecidable. □

We can adapt the proof to show that the existence of a controllable PDS for a subset of states $S' \subseteq S$ is undecidable for FSMs with *two* ports. To prove this we simply construct an FSM $\mathcal{M}'$ from $\mathcal{M}$ by dropping port 0 and taking $S' = \{s_1, s_2, s_3, s_4\}$.

Corollary 3.1. *Given FSM M and a state set $S' \subseteq S$, checking the existence of a controllable* PDS *for state set S' is undecidable and this holds even if we restrict attention to FSMs with two ports.*

Recall that checking the existence of a PDS is a decidable problem when a single-port FSM is under consideration [40]. Hence one may wonder why such a complexity jump exists. We observe that this negative result is due to what we call "unsplitting". The unsplitting issue is that, as a result of observability problems, it is possible to have the situation in which an input sequence $\bar{x}$ distinguishes two states of M but some extension $\bar{x}x$ of $\bar{x}$ does not.

Example 2. Consider the FSM in Figure 4 and input sequence x_1x_2. The local traces from different states at port 1 are as follows: x_1o_1 from s_1, $x_1o_1o_1$ from s_2, $x_1o_1'o_1$ from s_3, and x_1o_1' from s_4. Note that each trace is different and so the tester at port 1 can distinguish the states. However, let us suppose that for testing purposes we need to concatenate the input sequence x_1x_2 with input x_2 to form $x_1x_2x_2$. The local traces at port 1 are now: $x_1o_1o_1$ from s_1, $x_1o_1o_1o_1$ from s_2, $\underline{x_1o_1'o_1}$ from s_3, and $\underline{x_1o_1'o_1}$ from s_4. The local traces at port 1 from states s_3 and s_4 are now indistinguishable.

As a result, we cannot apply techniques that maintain a partition of the set of states of M (identifying which have not yet been distinguished) and refine the partition whenever a new input is chosen.

It has been shown that for a given FSM M and a port $p \in \mathcal{P}$ it is possible to construct a controllable input sequence, that distinguishes two states of M, in polynomial time and this sequence has length at most $k(n-2)+1$ [17]. Such a sequence is called a separating sequence. Therefore, one can use separating sequences of length $k(n-2)+1$ to distinguish states. Since we require at most $n-1$ separating sequences to pairwise distinguish all of the states of M, the sum of the lengths of the separating sequences is at most $\ell = (n-1)(k(n-2)+1)$. In practice one might use such a value ℓ as an upper bound on the length of any controllable preset distinguishing sequence $\bar{x}$ used; if there is no controllable PDS with length less than ℓ then one instead uses a set of separating sequences, of this total length, that distinguish the states of M. This leads to the following problem

Definition 3.2. *The* Bounded PDS problem *is to decide whether there is a controllable* PDS $\bar{x}$ *for a given FSM M such that the length of the* PDS *is not more than $\ell \in \mathbb{Z}_{>0}$.*

We will show that when we limit the length of a PDS, we can decide whether the underlying FSM possesses a controllable PDS. It is known that for single-port FSMs the shortest PDSs can be exponentially long [40]. Clearly, this result also holds for FSMs. On the other hand, as stated above, the length of a separating sequence is bounded by a polynomial function. Therefore, we will consider the general bounded PDS problem and the case where the bound is a polynomial function of the size of the FSM under consideration. We use the following result regarding PDSs from [39], which is immediate from the corresponding result for single-port FSMs [40].

Lemma 3.1. *Given a single-port FSM M in which no transition produces empty output, checking the existence of a preset distinguishing sequence is* PSPACE-Complete.

Proposition 3.1. *Given an FSM M, checking the existence of a controllable* PDS *with length at most ℓ 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 PDSs for M. First observe that all input sequences are controllable. In addition, since no output is produced at port 1, the restriction that no input can follow a difference in output is always satisfied. Thus, we can consider all input sequences. Finally, an input sequence distinguishes two states of M if and only if it distinguishes two states of M_1. Thus, an input sequence is a controllable PDS for M if and only if it is a PDS for M_1. We set $\ell = 2^n - 1$, which is a known upper bound on the length of the shortest PDS for a single-port FSM [49]. Thus, M_1 has a PDS if and only if M has a controllable PDS of length at most ℓ and so the result follows from Lemma 3.1. □

Now we can show that an algorithm that uses exponential space can decide the bounded PDS problem.

Proposition 3.2. *Given an FSM M, checking the existence of a controllable* PDS *of length ℓ is in* EXPSPACE.

Proof. A non-deterministic Turing machine can guess an input sequence $\bar{x}$ of length ℓ and it can compute and store each $\lambda(s, \bar{x})$ in space that is polynomial in ℓ. In order to check that $\bar{x}$ is controllable the Turing machine can simply compare prefixes of the sequences of the form $\lambda(s, \bar{x})$: there are controllability problems if there are prefixes σ and σ' of such traces with different lengths that have the same projection at a port p such that after σ and σ' the behaviour of the tester at p differs (the tester at p cannot distinguish between these cases). This can be checked in time that is polynomial in terms of ℓ. Finally, the Turing machine can check in time that is polynomial in terms of ℓ whether $\bar{x}$ is a PDS. Thus, a non-deterministic Turing machine can check whether a guess $\bar{x}$ is a controllable PDS in space that is polynomial in terms of ℓ and so exponential in terms of the representation of ℓ (that takes $O(\log_2 \ell)$ space). The problem is therefore in non-deterministic EXPSPACE. Finally, using Savitch's Theorem [50] we know that a deterministic Turing machine can also solve the problem in exponential space. We therefore have that the problem is in EXPSPACE. □

Propositions 3.1 and 3.2 give the following.

Theorem 3.2. *Given an FSM M and a bound ℓ, deciding whether there is a controllable* PDS *of length ℓ for M is in* EXPSPACE *and is* PSPACE-Hard. *This holds even if we restrict attention to FSMs with two ports.*

As noted before, instead of using a PDS we could use a set containing controllable separating sequences and we have a polynomial upper bound on the sum of the lengths of the sequences in such a set. Thus, in practice we are likely to have a polynomial upper bound on the length of PDSs in which we are interested.

Proposition 3.3. *Given an FSM M with n states and function f, that is bounded above by a polynomial, deciding whether there is a controllable* PDS *of length at most $\ell = f(n)$ is in* NP.

Proof. A non-deterministic Turing Machine can guess an input sequence $\bar{x} = x_1 \dots x_j$ of length at most ℓ and can then generate the corresponding set $Tr = \{\bar{x}/\lambda(s, \bar{x}) | s \in S\}$ of traces in polynomial time. It is then sufficient for the Turing Machine to check that: a) $\bar{x}$ is controllable for M; and b) $\bar{x}$ is a PDS for M. In order to decide whether $\bar{x}$ is controllable for M it is sufficient to compare the prefixes of traces in Tr; if two prefixes have the same projection at port p then the action of the local tester at p must be the same after each. Since the size of Tr is bounded above by a polynomial in n, this can be checked in polynomial time. In order to check whether $\bar{x}$ is a PDS we first check whether there are states $s \neq s'$ such that traces $\bar{x}/\lambda(s, \bar{x})$ and $\bar{x}/\lambda(s', \bar{x})$ have the same set of projections on the ports and this can be achieved in polynomial time. Finally, we need to check that if the ith input x_i in $\bar{x}$ is at port p then for all $s \neq s'$ we have that $\pi_p(x_1 \dots x_{i-1}/\lambda(s, x_1 \dots x_{i-1})) = \pi_p(x_1 \dots x_{i-1}/\lambda(s', x_1 \dots x_{i-1}))$ and again this can be checked in polynomial time. Thus, a non-deterministic Turing Machine can solve the problem in polynomial time and so the problem is in NP. □

We now prove that the problem is NP-Hard, using a reduction from the Bounded PCP.

Definition 3.3. *The Bounded Post's Correspondence Problem (BPCP) is to decide, for sequences $\alpha_1, \alpha_2, \dots, \alpha_b$ and $\beta_1, \beta_2, \dots, \beta_b$ and $K \leq b$, whether there is a sequence $a_1 a_2 \dots a_j$ of indices in $[1..b]$ such that $j \leq K$ and $\alpha_{a_1}\alpha_{a_2} \dots \alpha_{a_j} = \beta_{a_1}\beta_{a_2} \dots \beta_{a_j}$.*

It is known that the Bounded Post's Correspondence Problem is NP-complete [51].

We can use the construction, given in the proof of Theorem 3.1, to define an FSM $\mathcal{M}$ in terms of $\alpha_1, \alpha_2, \dots, \alpha_b$ and $\beta_1, \beta_2, \dots, \beta_b$. Recall that in this construction m is the length of the longest sequence in $\alpha_1, \alpha_2, \dots, \alpha_b, \beta_1, \beta_2, \dots, \beta_b$. From the proof of Theorem 3.1, we know that an input sequence $\bar{x}$ is a controllable PDS for $\mathcal{M}$ if and only if $\bar{x}$ is of the form *init* $x^m_{a_1} x^m_{a_2} \dots x^m_{a_j}$ followed by x' for some $a_1 a_2 \dots a_j$ such that $\alpha_{a_1}\alpha_{a_2} \dots \alpha_{a_j} = \beta_{a_1}\beta_{a_2} \dots \beta_{a_j}$. Thus, $\mathcal{M}$ has a controllable PDS of length at most $mK + 2$ if and only if there is a solution to the corresponding instance of the BPCP. In principle, to use this we require the upper bound to be a polynomial in terms of the number of states of $\mathcal{M}$. However, this can be achieve by making the following small changes.

1. Remove states s_1', s_2', s_3', s_4' and rename states s_1, s_2, s_3, s_4 to $s_1^0, s_2^0, s_3^0, s_4^0$ respectively.
2. Include extra states of the form $s_1^i, s_2^i, s_3^i, s_4^i$, $0 < i \leq K$, such that the input of an x_j a total of m times takes $s_1^i, s_2^i, s_3^i, s_4^i$ to $s_1^{i+1}, s_2^{i+1}, s_3^{i+1}, s_4^{i+1}$ respectively ($0 \leq i < K$). Naturally, output sequence α_j is produced from states s_1^i and s_2^i and output sequence β_j is produced from states s_3^i and s_4^i.
3. The input of x' in s_j^i leads to output s_j ($1 \leq j \leq 4$, $0 < i \leq K$).
4. The input of an x_j takes states $s_1^K, s_2^K, s_3^K, s_4^K$ to s_1^K with fixed output.
5. If required, states are added so that the number of states exceeds $mK + 2$. These states are all distinguished from every other state by the initial input of *init*.

The first three changes essentially 'unfold' the cycle, in states s_1', s_2', s_3', s_4' of $\mathcal{M}$. The fourth change ensures that an input sequence of the form $x_{a_1}^m x_{a_2}^m \dots x_{a_j}^m$ can only be a controllable PDS if $j \leq K$. This ensures that the resultant FSM has a controllable PDS if and only if it has a PDS that corresponds to a solution to the given instance of the BPCP. The last step ensures that we can use the number of states of the FSM as an upper bound. We therefore have the following result.

Proposition 3.4. *Given an FSM M with n states and function f, that is bounded above by a polynomial, deciding whether there is a controllable* PDS *of length at most $\ell = f(n)$ is* NP-Hard.

We therefore have the following result.

Theorem 3.3. *Given an FSM M with n states and function f, that is bounded above by a polynomial, deciding whether there is a controllable* PDS *of length at most $\ell = f(n)$ is* NP-Complete.

This is a relatively positive result: it shows that if we restrict attention to the usual case (we have a polynomial upper bound on the length of a *useful* PDS) then the existence problem is NP-Complete. As far as we are aware there is no corresponding result for single-port FSMs but clearly the general bounded case for single-port FSMs is PSPACE-Complete.

In the following section we consider the problem of PDS generation for a class of FSMs that we call C-FSMs.

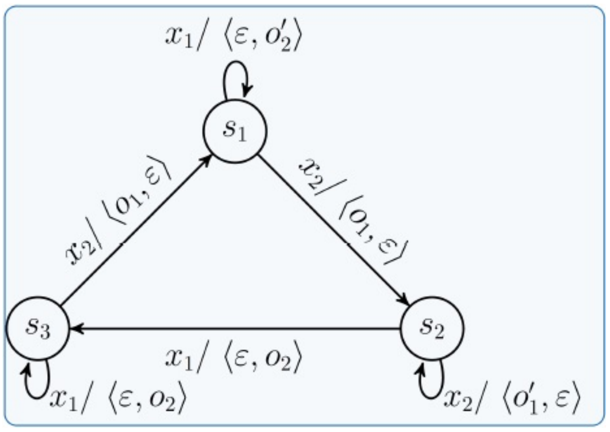

$x_1/\ \langle \varepsilon, o_2' \rangle$
s_1
$x_2/\ \langle o_1, \varepsilon \rangle$
$x_2/\ \langle o_1, \varepsilon \rangle$
s_3
s_2
$x_1/\ \langle \varepsilon, o_2 \rangle$
$x_1/\ \langle \varepsilon, o_2 \rangle$
$x_2/\ \langle o_1', \varepsilon \rangle$

FIGURE 5: A simple C-FSM.

4. PDS GENERATION: A SPECIAL CASE

In this section we prove that the PDS existence problem is decidable for a special class of FSMs. In this class of FSMs, if the input of x leads to no output at port p for some state of an FSM M then the input of x leads to no output at p for all states of M. This can be seen as imposing a fixed pattern on the communication that occurs between the testers at the ports through interacting with M. Below (Proposition 4.2) we prove that if an FSM is in this class of FSMs then an input sequence is controllable for a state s if and only if it is controllable for all non-empty sets of states; this simplifies the reasoning regarding controllability. We also prove (Proposition 4.3) that this class of FSMs does not suffer from observability problems. We then show that the FSM M can be mapped to a canonical FSM $\chi_{min}^{\mathcal{P}}(M)$, with PDSs being generated from $\chi_{min}^{\mathcal{P}}(M)$. This allows us to derive an upper bound on the length of a shortest PDS (Lemma 4.1), with decidability being a consequence of this.

An example of such an FSM is given in Figure 5.

Definition 4.1. *An FSM $M = (\mathcal{P}, S, s_0, X, Y, \delta, \lambda)$ is said to be* consistent *if for all $s, s' \in S$, $x \in X$, and $p \in \mathcal{P}$, $\pi_p(\lambda(s, x)) = \varepsilon$ implies that $\pi_p(\lambda(s', x)) = \varepsilon$.*

We will call an FSM that is consistent a *C-FSM*. We now discuss the practical relevance of C-FSMs.

First, for an FSM $M = (\mathcal{P}, S, s_0, X, Y, \delta, \lambda)$ and a subset of input alphabet $X' \subseteq X$, let $M|_{X'}$ be the restriction of M to the transitions with inputs in X'. Formally, $M|_{X'}$ is the FSM $M|_{X'} = (\mathcal{P}, S, s_0, X', Y, \delta', \lambda')$ where $\delta'(s, x) = \delta(s, x)$ and $\lambda'(s, x) = \lambda(s, x)$ for $x \in X'$. Even if an FSM M is not a C-FSM, there may exist a subset of inputs X' such that $M|_{X'}$ is a C-FSM. In such cases, the results given for C-FSMs apply to $M|_{X'}$ and hence to M.

Second, we might obtain a C-FSM if we fix the configuration of a system. Consider, for example, *Mobile Multiplayer Game Protocols* [52, 53]. These might include the following.

1. MakeMove: a client applies an input (which encapsulates their move) to the server and none of the clients receive an output.

2. Message: a client sends an input (which encapsulates a message) and the server issues outputs (messages) to the friends (team members) only.

3. Stat: a client sends an input (which encapsulates a message) and the server supplies the output which encapsulates stat. info (inventory, life, etc) only to the friends (team members).

4. AddUSer: a client sends an input (which encapsulates a request) to the server and only the server receives this message and returns output (success/failure) to the client.

5. JoinGame: a client sends an input (which encapsulates a message) and the server sends an output to all players when the new player is successfully added to the game.

In the above, transitions corresponding to a single input lead to the same pattern of communication irrespective of state as long as we fix the users and the set of friends for each user.

Third, if we consider now *Heterogeneous Distributed Computing* (HDC) [54], we also find that some procedures can be represented by C-FSMs. Given a task and a set of physically distributed computers (slaves) each of which have separate storages (buckets), a HDC server aims to distribute tasks to the buckets of slaves, so that slaves do computation [55]. One well known example that uses this protocol is *Generating Large Prime Numbers* (see, for example, [56]).

HDC recognises the fact that the slaves in the distributed system may differ in their capabilities through, for example, some having faster clock cycles, GPUs, larger memory capacity, bigger disk farms, printers and other peripherals. Thus the HDC server may form *groups of slaves* such that two slaves are in the same group if they have identical properties. It is possible to select a group of slaves for a particular task type using *specification strings*. For example in the Amazon Elastic Compute Cloud (EC2) specification string `m1.large` refers to systems with *7.5GB of RAM, 2 virtual cores, 850 GB of Hard Disk, on a 64-bit platforms* [57]. In [58] extensions of specification strings are discussed.

The server may also form *task-group matching* such that task types are mapped to groups if they can process instances of these task types [54, 59]. Afterwards, the HDC server distributes work to the slaves [54, 60].

A simple abstract scenario for HDC hence might be as follows: assume that we have groups of slaves $G_1 = \{pc_1, pc_3, \ldots\}, G_2 = \{pc_k, pc_l, \ldots\}, \ldots G_k = \{pc_m, pc_n, \ldots\}$ such that two slaves reside in a group if they share identical properties (note that G_i can be thought of as a specification string for the slaves in set G_i). We have a set of task types $\{T_1, T_2, T_3, \ldots T_p\}$ and we let GT_i be the set of tasks that can be assigned to slaves in G_i ($1 \leq i \leq m$). Clearly, such a system is complex and the procedures used by such a system cannot be represented by a C-FSM. However, as discussed in [58, 61, 62] in such a system usually slaves monitor their buckets and when a task request arrives, slaves start executing the task. In order to achieve this the server might use the following procedures.

1. PutConcurrentWork_`Spec.`_`String`: the server sends an input (identical tasks) to a group of slaves designated by the specification string.

2. PutWork_`Spec.`_`String`: the server sends an input (a vector of tasks) to a group of slaves designated by the specification string.

3. NoWork_`Spec.`_`String`: the server sends an input (message) to a group of slaves, designated by the specification string, declaring that there is no suitable work for the group.

Therefore procedures like NoWork, PutWork and PutConcurrentWork will execute through restricted communication patterns. However, let us suppose that task types and groups of slaves have been chosen such that $\{GT_1, \ldots, GT_m\}$ is a partition of the set of task types: given a task type T_i there is only one G_j to which T_i can be allocated. Further, let us suppose that when a task or vector of tasks is allocated to a group it is allocated to all slaves in that group. Now, if the server asks slaves to execute a task from T_3, for example, then it will contact the same set of slaves, irrespective of the actual task, and so the server issues the procedure call `PutConcurrentWork`_G_1 with its data (assuming $T_3 \in GT_1$). If we extend the inputs so that they include the specification string that defines the task type then the above operations define a C-FSM.

This example included two approaches that helped us model a system as a C-FSM. First, we restricted the scenario so that, for example, the $\{GT_1, \ldots, GT_m\}$ is a partition of the set of task types. Second, we extended the set of inputs by introducing several 'variants' of an input with each variant being defined by a specification string that represents a set of values for some input parameter. The situation here is that while an input (operation name) x does not have a fixed communication pattern, it does have a fixed communication pattern if we choose a particular specification string. In principle, we could have one input for each possible parameter value for x but we obtain a smaller model if, as above, we group together parameter values that lead to the same communication pattern.

C-FSMs have the following important properties, the first two relating to controllability.

Proposition 4.1. *Given distinct states s and s' of C-FSM M, if input sequence $\bar{x}$ is controllable from state s then $\bar{x}$ is controllable from s'.*

Proof. Let us suppose that $\bar{x} = x_1x_2 \dots x_r$, trace x_1/y_1 $x_2/y_2 \dots x_r/y_r$ labels a walk from state s and trace x_1/y'_1 $x_2/y'_2 \dots x_r/y'_r$ labels a walk from state s'. By definition, $\bar{x}$ is controllable from s if and only if for all $1 < i \leq r$ we have that if $port(x_i) = p$ then $\pi_p(x_{i-1}/y_{i-1}) \neq \varepsilon$. But since M is a C-FSM, this is the case if and only if for all $1 < i \leq r$ we have that if $port(x_i) = p$ then $\pi_p(x_{i-1}/y'_{i-1}) \neq \varepsilon$. By definition, this is the case if and only if $\bar{x}$ is controllable from s' and so the result follows. □

We can extend this result to a set of states.

Proposition 4.2. *Given C-FSM M with state set S, state $s \in S$ and set $S' \subseteq S$ of states of M, if input sequence $\bar{x}$ is controllable from s then $\bar{x}$ is controllable from S'.*

Proof. Let us suppose that $\bar{x} = x_1x_2 \dots x_r$ and trace $x_1/y_1\, x_2/y_2 \dots x_r/y_r$ labels a walk from state s. By the definition of controllability, we require to prove that for all $s_i, s_j \in S'$ and proper prefixes $\bar{x}_i$ and $\bar{x}_j$ of $\bar{x}$ with $|\bar{x}_i| \leq |\bar{x}_j|$ and $|\bar{x}_i| < r$, if $port(x_{i+1}) = p$ then $(\pi_p(\lambda(s_i, \bar{x}_i)) = \pi_p(\lambda(s_j, \bar{x}_j))) \implies (\bar{x}_i = \bar{x}_j)$. Since M is a C-FSM, $\pi_p(\lambda(s_i, \bar{x}_i)) = \pi_p(\lambda(s_j, \bar{x}_j))$ implies that there are no inputs or outputs at p in the subsequence $x_{i+1}/y_{i+1}\ x_{i+2}/y_{i+2} \dots x_j/y_j$ of $x_1/y_1\ x_2/y_2 \dots x_r/y_r$. Since $\bar{x}$ is controllable from s and the input x_{j+1} is at port p, we have that $x_{i+1}/y_{i+1}\ x_{i+2}/y_{i+2} \dots x_j/y_j = \varepsilon$ and so $\bar{x}_i = \bar{x}_j$ as required. □

The following relates to observability problems.

Proposition 4.3. *Given C-FSM M, states s and s' of M, and input sequence $\bar{x}$, if $\lambda(s, \bar{x}) \neq \lambda(s', \bar{x})$ then there is some port p such that $\pi_p(\lambda(s, \bar{x})) \neq \pi_p(\lambda(s', \bar{x}))$.*

Proof. Let $\bar{x}'$ be the shortest prefix of $\bar{x}$ such that $\lambda(s, \bar{x}') \neq \lambda(s', \bar{x}')$. Then $\bar{x}' = \bar{x}''x$ for some $x \in X$ and by the minimality of $\bar{x}'$ we have that $\lambda(s, \bar{x}'') = \lambda(s', \bar{x}'')$. Since $\lambda(s, \bar{x}') \neq \lambda(s', \bar{x}')$, $\lambda(\delta(s, \bar{x}''), x) \neq \lambda(\delta(s', \bar{x}''), x)$. Thus, there must be some port p such that $\pi_p(\lambda(s, \bar{x}')) \neq \pi_p(\lambda(s', \bar{x}'))$. But if $\bar{x} = \bar{x}'\bar{x}'''$ then since M is a C-FSM we know that $\pi_p(\lambda(\delta(s, \bar{x}'), \bar{x}'''))$ and $\pi_p(\lambda(\delta(s, \bar{x}'), \bar{x}'''))$ contain the same number of outputs and so $\pi_p(\lambda(s, \bar{x})) \neq \pi_p(\lambda(s', \bar{x}))$ as required. □

In other words, for a given C-FSM when an input sequence $\bar{x}$ 'globally distinguishes' states s, s' of M then $\bar{x}$ also 'locally distinguishes' the states s, s' of M.

We can therefore conclude that when testing from C-FSMs there are no observability problems and an input sequence is controllable for a state if and only if it is controllable for all non-empty sets of states. It will transpire that these properties simplify the PDS existence problem.

In the following we use a *canonical* FSM [33]. A canonical FSM ($\chi^{\mathcal{P}}_{min}(M)$) of a C-FSM M with respect to port p is an FSM that captures only the controllable walks of M. In other words, by considering walks of $\chi^{\mathcal{P}}_{min}(M)$, we avoid controllability problems. We will show that by the virtue of Proposition 4.3, using the canonical FSM $\chi^{\mathcal{P}}_{min}(M)$ of M will allow us to avoid controllability and observability problems.

The construction of the FSM $\chi^{\mathcal{P}}_{min}(M)$ uses ideas from [63]. Given FSM M, the canonical FSM $\chi^{\mathcal{P}}_{min}(M)$ is constructed in two steps. First we construct a partial[4] FSM $\chi_{min}(M)$, second we construct $\chi^{\mathcal{P}}_{min}(M)$.

The partial FSM $\chi_{min}(M) = (\mathcal{P}, S_{min}, s'_0, X, Y, \delta_{min}, \lambda_{min})$, is constructed as described in [33]. The first step is to reveal the adjacent transitions that can be followed without raising a controllability problem. In order to do this, we first construct the *Arrive* and *Depart* lists which are defined as follows:

Let T be the set of transitions of M.

- $Depart^p(s_i) = \{(s_i, s_j, x/y) \in T | x \in X_p\}$ is the set of transitions from s_i whose input is at p.
- $Arrive^P(s_i) = \{(s_j, s_i, x/y) \in T | ports(x/y) = P\}$ is the set of transitions ending at s_i that involve the set P of ports.

For each $s_i \in S$ and $P \subseteq \mathcal{P}$ there can be vertex s^P_i representing the situation in which the state is s_i and the next input must be at a port in P (since otherwise there will be a controllability problem).

The set S_{min} of states of $\chi_{min}(M)$ is defined by the following.

1. For all $1 \leq i \leq n$ and $P \subseteq \mathcal{P}$, $s^P_i \in S_{min}$ if $Arrive^P(s_i) \neq \emptyset$.

2. State $s^{\mathcal{P}}_0$ is in S_{min} and $s^{\mathcal{P}}_0$ is the initial state of $\chi_{min}(M)$.

The state $s^{\mathcal{P}}_0$ represents the situation before testing starts: the SUT is in the initial state and, since no inputs have been applied, the first input can be applied at any port without causing a controllability problem. The set T_{min} of transitions of $\chi_{min}(M)$ (and so the functions δ_{min} and λ_{min}) is defined by, for each transition $t = (s_i, s_j, x/y) \in T$ and $s^P_i \in S_{min}$ with $port(x) \in P$, including in T_{min} the transition $(s^P_i, s^{P_t}_j, x/y)$ where $P_t = ports(x/y)$. As an example, consider the FSM M given in Figure 5. The *Arrive* and *Depart* lists for FSM M are given in Table 1 and the canonical FSM $\chi_{min}(M)$ is given in Figure 6b.

The following results are known [33].

Proposition 4.4. *For each controllable walk $\bar{\rho}$ in M that starts at s_0, there is a unique controllable walk $\bar{\rho}'$ in $\chi_{min}(M)$ that starts at $s^{\mathcal{P}}_0$ such that $\bar{\rho}$ and $\bar{\rho}'$ have the same label.*

[4]An FSM is partial if there is at least one state s and input x such that the response to x when in state s is not specified.

$Depart^1(s_1)$	$(s_1, s_1, x_1/ < \varepsilon, o_2' >)$	$Arrive^{1,2}(s_1)$	$(s_1, s_1, x_1/ < \varepsilon, o_2' >)$
$Depart^2(s_1)$	$(s_1, s_2, x_2/ < o_1, \varepsilon >)$	$Arrive^{1,2}(s_1)$	$(s_3, s_1, x_2/ < o_1, \varepsilon >)$
$Depart^1(s_2)$	$(s_2, s_3, x_1/ < \varepsilon, o_2 >)$	$Arrive^{1,2}(s_2)$	$(s_1, s_2, x_2/ < o_1, \varepsilon >)$
$Depart^2(s_2)$	$(s_2, s_2, x_2/ < o_1', \varepsilon >)$	$Arrive^{1,2}(s_2)$	$(s_2, s_2, x_2/ < o_1', \varepsilon >)$
$Depart^1(s_3)$	$(s_3, s_3, x_1/ < \varepsilon, o_2 >)$	$Arrive^{1,2}(s_3)$	$(s_2, s_3, x_1/ < \varepsilon, o_2 >)$
$Depart^2(s_3)$	$(s_3, s_1, x_2/ < o_1, \varepsilon >)$	$Arrive^{1,2}(s_3)$	$(s_3, s_3, x_1/ < \varepsilon, o_2 >)$

TABLE 1: *Arrive* and *Depart* lists for FSM M given in Figure 5.

Proposition 4.5. *For each walk $\bar{\rho}'$ in $\chi_{min}(M)$ that starts at $s_0^{\mathcal{P}}$, there is a unique controllable walk $\bar{\rho}$ in M that starts at s_0 such that $\bar{\rho}$ and $\bar{\rho}'$ have the same label.*

The final step is to create a completely specified FSM $\chi_{min}^{\mathcal{P}}(M) = (\mathcal{P}, S_{min}^p, X, Y, \delta_{min}^{\mathcal{P}}, \lambda_{min}^{\mathcal{P}})$ in which for each state s and port p there is a state $s^{\{p\}}$; this will allow us to explore controllable PDSs that start with input at p. This is achieved by applying the following to $\chi_{min}(M)$.

1. For every state s of M such that $s^{\{p\}}$ is not in S_{min}.
   (a) Add the state $s^{\{p\}}$.
   (b) For every input x at p, if $s_i = \delta(s, x)$ and $y = \lambda(s, x)$ then add the transition $(s^{\{p\}}, s_i^P, x/y)$ such that $P = ports(x/y)$.
2. For every input x and state $s \in S_{min}$ such that there is no transition from s with input x, add a self-loop transition from s to s with input x and output ε at all ports.

The completely specified canonical machine $\chi_{min}^{\mathcal{P}}(M)$ of M is given in Figure 6b. We can now show how controllable PDS construction for M relates to PDS construction for $\chi_{min}^{\mathcal{P}}(M)$. Note that we require a particular type of PDS for $\chi_{min}^{\mathcal{P}}(M)$: since a PDS will be applied by distributed testers we require that the trace before we apply an input at p has fixed projection at p (see Definition 2.7). In the following, a PDS for $\chi_{min}^{\mathcal{P}}(M)$ is said to be *non-redundant* if it does not lead to the execution of any of the self-loops added to make $\chi_{min}^{\mathcal{P}}(M)$ completely-specified. In the following, given an input sequence $\bar{x}$ of length ℓ and $i \leq \ell$ we let $\bar{x}_i$ denote the prefix of $\bar{x}$ that has length i.

Proposition 4.6. *An input sequence $\bar{x}$ is a controllable* PDS *that starts with input at p for set $S' = \{s_1, s_2 \ldots, s_r\} \subseteq S$ of states of M if and only if $\bar{x}$ is a non-redundant* PDS *for set $S'' = \{s_1^{\{p\}}, s_2^{\{p\}}, \ldots, s_r^{\{p\}}\}$ of states of $\chi_{min}^{\mathcal{P}}(M)$ such that for all $1 < i \leq |\bar{x}|$, if $x_i \in X_q$ then for all $s_i^{\{p\}}, s_j^{\{p\}} \in S''$ we have that $\pi_q(\lambda_{min}^{\mathcal{P}}(s_i^{\{p\}}, \bar{x}_{i-1})) = \pi_q(\lambda_{min}^{\mathcal{P}}(s_j^{\{p\}}, \bar{x}_{i-1}))$.*

Proof. First let us suppose that $\bar{x}$ is a controllable PDS for $S' = \{s_1, s_2, \ldots, s_r\}$ that starts with input at p. Since $\bar{x}$ is controllable, the label of the walk in M from s_i that has input portion $\bar{x}$ is the same as the label of the walk in $\chi_{min}^{\mathcal{P}}(M)$ from $s_i^{\{p\}}$ that has input portion $\bar{x}$. Thus, since $\bar{x}$ distinguishes the states in S' it also distinguishes the states in S''. Further, since $\bar{x}$ is a controllable PDS for S', by definition for all $1 < i \leq |\bar{x}|$, if $x_i \in X_q$ then for all $s_i^{\{p\}}, s_j^{\{p\}} \in S''$ we have that $\pi_q(\lambda_{min}^{\mathcal{P}}(s_i^{\{p\}}, \bar{x}_{i-1})) = \pi_q(\lambda_{min}^{\mathcal{P}}(s_j^{\{p\}}, \bar{x}_{i-1}))$. Finally, since $\bar{x}$ is controllable in M, it is non-redundant in $\chi_{min}^{\mathcal{P}}(M)$ and so the result holds.

Now let us suppose that $\bar{x}$ is a non-redundant PDS for set $S'' = \{s_1^{\{p\}}, s_2^{\{p\}}, \ldots, s_r^{\{p\}}\}$ of states of $\chi_{min}^{\mathcal{P}}(M)$ such that for all $1 < i \leq |\bar{x}|$, if $x_i \in X_q$ then for all $s_i^{\{p\}}, s_j^{\{p\}} \in S''$ we have that $\pi_q(\lambda_{min}^{\mathcal{P}}(s_i^{\{p\}}, \bar{x}_{i-1})) = \pi_q(\lambda_{min}^{\mathcal{P}}(s_j^{\{p\}}, \bar{x}_{i-1}))$. Similar to before, the label of the walk from $s_i^{\{p\}}$ in $\chi_{min}^{\mathcal{P}}(M)$ that has input portion $\bar{x}$ is the same as the label of the walk from s_i in M that has input portion $\bar{x}$. Thus, $\bar{x}$ distinguishes the states in S' and so, by Proposition 4.3, $\bar{x}$ distinguishes the states from S' in distributed testing. □

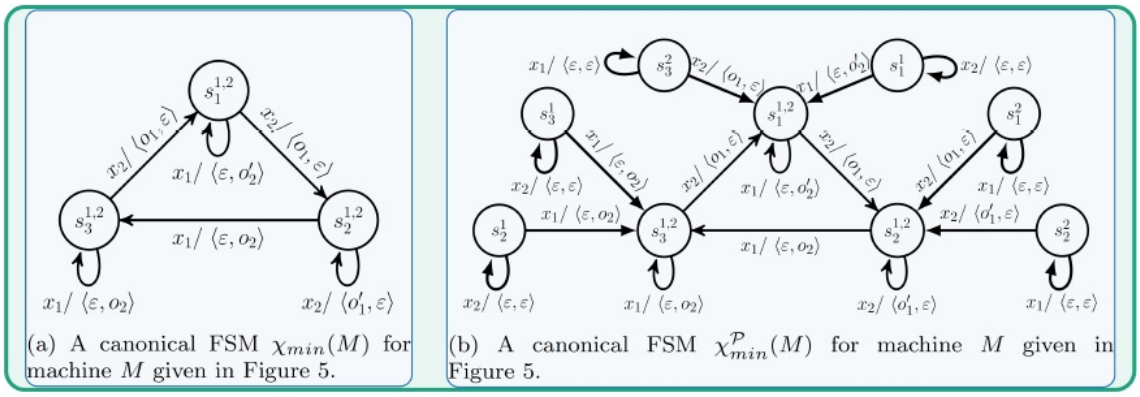
The definition of $\chi_{min}^{\mathcal{P}}(M)$ uses sets of ports as labels on states and this might appear to lead to a combinatorial explosion. However, the number of states of $\chi_{min}^{\mathcal{P}}(M)$ is bounded above by the number of transitions of M plus one (for the initial state) plus an additional $|S||\mathcal{P}|$ states of the form $s^{\{p\}}$. Let n be the number of states, k be the number of ports and m be the number of inputs of C-FSM M. Also let n_{min} be the number of states of machine $\chi_{min}^{\mathcal{P}}(M)$, then $n_{min} \leq nk+nm+1$. Thus, $\chi_{min}^{\mathcal{P}}(M)$ can be constructed in polynomial time.

We now present some definitions and observations related to PDSs and then we give an upper bound on the length of minimal controllable PDSs for C-FSMs[5].

Given an input sequence $\bar{x}$, we will let $\mathcal{B}^{\bar{x}}$ denote the set of sets of 'current states' that can occur if we know that $\bar{x}$ has been applied from a state in S''. Thus, if one or more states in S'' leads to output $\bar{y}$ when $\bar{x}$ is applied then one of the sets in $\mathcal{B}^{\bar{x}}$ is the set of states that can be reached from states in $S'' = \{s_1^{\{p\}}, \ldots, s_n^{\{p\}}\}$ by walks with label $\bar{x}/\bar{y}$ (the set $\{\delta_{min}^{\mathcal{P}}(s^{\{p\}}, \bar{x}) | \lambda_{min}^{\mathcal{P}}(s^{\{p\}}, \bar{x}) = \bar{y}\}$). More formally, we have that $\mathcal{B}^{\bar{x}} = \{\{\delta_{min}^{\mathcal{P}}(s^{\{p\}}, \bar{x}) | \lambda_{min}^{\mathcal{P}}(s^{\{p\}}, \bar{x}) = \bar{y}\} | \exists s^{\{p\}} \in S''. \bar{y} = \lambda_{min}^{\mathcal{P}}(s^{\{p\}}, \bar{x})\}$.

While applying an input sequence $\bar{x}_i$, different states of S'' may produce different traces. This will lead to the *splitting* of the set S'' into smaller sets of states. On the

[5]This upper bound is used in the proof that the decision problem is in PSPACE; it seems likely that smaller upper bounds can be found but that is not a concern here.


$s_1^{1,2}$
$s_2^{1,2}$
$s_3^{1,2}$
$x_2/\langle o_1,\varepsilon\rangle$
$x_1/\langle\varepsilon,o_2'\rangle$
$x_1/\langle\varepsilon,o_2\rangle$
$x_2/\langle o_1',\varepsilon\rangle$
s_1^1
s_1^2
s_2^1
s_2^2
s_3^1
s_3^2
$x_1/\langle\varepsilon,\varepsilon\rangle$
$x_2/\langle\varepsilon,\varepsilon\rangle$
$x_1/\langle\varepsilon,o_2^1\rangle$

(a) A canonical FSM $\chi_{min}(M)$ for machine M given in Figure 5.

(b) A canonical FSM $\chi^{\mathcal{P}}_{min}(M)$ for machine M given in Figure 5.

FIGURE 6: Canonical forms of FSM given in Figure 5

other hand if a state $s^{\{p\}}$ produces a different output trace from all other states in set $S'' \setminus \{s^{\{p\}}\}$ then $s^{\{p\}}$ is *distinguished* from states $S'' \setminus \{s^{\{p\}}\}$.

We use $\delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_i)$ to denote the set $\mathcal{B}^{\bar{x}_i}$. Clearly, the application of a PDS $\bar{x}$ from set S'' will lead to a set $\mathcal{B}^{\bar{x}}$ of cardinality n; each set is a singleton set. We now give an upper bound on PDS length for C-FSMs.

Lemma 4.1. *Given a C-FSM M with n states, k ports, and m inputs, M has a controllable* PDS *if and only if it has one of length at most $n(n_{min})^n$ where $n_{min} = nk + nm + 1$.*

Proof. Consider the FSM $\chi^{\mathcal{P}}_{min}(M)$ of M with set S_{min} of states. Let $S'' = \{s_1^{\{p\}}, s_2^{\{p\}}, \ldots, s_n^{\{p\}}\}$ and let $\bar{x}$ be a shortest PDS for state set S'' of $\chi^{\mathcal{P}}_{min}(M)$.

Now let us assume that $S'' = \mathcal{B}$, $\bar{x} = \bar{x}_a\bar{x}'x_b\bar{x}_b$ where $\bar{x}' = x_0x_1 \ldots x_{|\bar{x}'|}$ is a fragment of PDS $\bar{x}$ such that $|\delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a)| < |\delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a x_0)| = |\delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a\bar{x}')| < |\delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a\bar{x}'x_b)|$.

We will prove that for any distinct proper prefixes $\bar{x}''$, $\bar{x}'''$ of $\bar{x}'$ we have that $\delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a\bar{x}'') \neq \delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a\bar{x}''')$. We will use proof by contradiction and assume that there exist proper prefixes $\bar{x}''$, $\bar{x}'''$ of $\bar{x}'$ such that $\delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a\bar{x}'') = \delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a\bar{x}''')$ and $|\bar{x}''| < |\bar{x}'''|$.

Now let us suppose that $\bar{x}''' = \bar{x}''\bar{x}_0$ and $\bar{x}' = \bar{x}'''\bar{x}_1 = \bar{x}''\bar{x}_0\bar{x}_1$. Since $\bar{x}$ is a PDS for S'' we must have that $\bar{x}_1x_b\bar{x}_b$ distinguishes any two states that are in the same set in $\delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a\bar{x}''\bar{x}_0)$. From Propositions 4.2 and 4.3 we know that no walk of $\chi^{\mathcal{P}}_{min}(M)$ causes controllability and observability problems in M. Hence, since $\delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a\bar{x}'') = \delta^{\mathcal{P}}_{min}(\mathcal{B}, \bar{x}_a\bar{x}''\bar{x}_0)$, we can replace $\bar{x}'$ (which equals $\bar{x}''\bar{x}_0\bar{x}_1$) by $\bar{x}''\bar{x}_1$ in $\bar{x}$. But this leads to a shorter controllable PDS, which contradicts the minimality of $\bar{x}$.

We can now note that the number of possible values for a set $\mathcal{B}^{\bar{x}_a}$ is bounded above by the number of possible mappings from the n states in S'' to the states reached from S'' by $\bar{x}_a$. Since there are n_{min} possible values that a state in S'' can be mapped to, this gives an upper bound of $(n_{min})^n$. Further, $\mathcal{B}^{\bar{x}}$ initially contains only one set and finally contains n sets and so there are at most $n-1$ points at which the size of $\mathcal{B}^{\bar{x}}$ increases. We can therefore conclude that a minimal PDS can be seen as being a sequence of at most $n-1$ subsequences each of length at most $(n_{min})^n$. Moreover, Proposition 4.6 implies that $\bar{x}$ is a PDS for state set S'' of $\chi^{\mathcal{P}}_{min}(M)$ if and only if $\bar{x}$ is a PDS for FSM M and so the result follows. □

Thus we conclude that PDS existence check for C-FSM is decidable (it is sufficient to check all input sequences of length at most $n(n_{min})^n$).

Theorem 4.1. *Given a C-FSM M, checking the existence of a controllable* PDS *is decidable.*

We will show that the problem is PSPACE-Complete. First we define a nondeterministic Turing Machine $\mathcal{T}$ that can decide the existence of a PDS for a state set S' of C-FSM M. $\mathcal{T}$ will apply inputs one at a time and maintain a current set $\mathcal{C}$ of pairs of states such that (s, s') is in $\mathcal{C}$ if and only if $s \in S'$ and the sequence of inputs received takes M from s to s'. $\mathcal{T}$ also maintains an equivalence relation r defined by two states $s, s'' \in S'$ being related under r if and only if the currently guessed input sequence $\bar{x}$ does not distinguish s and s' $(\lambda(s, \bar{x}) = \lambda(s'', \bar{x}))$. Finally, $\mathcal{T}$ maintains a set of ports P_c from which an input can be supplied (the ports where no differences in outputs have been observed).

Clearly, these pieces of information can be updated when a new input is received: after an input is guessed, $\mathcal{T}$ updates the current set information, the equivalence relation r and finally the set of ports (a port is removed from P_c if the latest input leads to different outputs at this port). Since the problem is to decide existence, $\mathcal{T}$ does not need to store the sequence of previous inputs received. Further, the input sequence received defines a PDS for S' if and only if no two different states from S' are related under r. We now prove that the PDS problem is in PSPACE for C-FSMs.

Proposition 4.7. *Given a C-FSM M, checking the existence of a controllable* PDS *is in* PSPACE.

Proof. We will show that a non-deterministic Turing

Machine $\mathcal{T}$ can decide whether there is a PDS using polynomial space. $\mathcal{T}$ will guess inputs one at a time. It will maintain the set $\mathcal{C}$ of pairs of states, equivalence relation r, the set of allowable ports P_c as described above and this uses polynomial space.

In each iteration, the Turing Machine first guesses an input x from a port in P_c. After guessing a new input x and updating $\mathcal{C}$, r, and P_c the machine checks whether the input sequence received defines a PDS for S: this is the case if and only if r relates no two different states of S. Thus, if M has a PDS for S then $\mathcal{T}$ will find such a PDS using polynomial space.

Consider the case where M does not have a PDS for S: we require that the non-deterministic Turing Machine terminates. In order to ensure this we use the result that if C-FSM M has n states then M has a PDS for S if and only if it has such a PDS with length at most $n(n_{min})^n$. $\mathcal{T}$ therefore includes a counter that counts how many inputs have been received: the machine terminates with failure if the counter exceeds the upper bound. Therefore we need additional $O(\log_2 (n(n_{min})^n)) = O(n \log_2(n_{min}))$ space for the counter and so the space required is still polynomial.

We have defined a non-deterministic Turing Machine that requires only polynomial space in order to solve the problem and so the problem is in nondeterministic PSPACE. We can now use Savitch's Theorem [50], which tells us that a problem is in PSPACE if and only if it is in non-deterministic PSPACE, and the result follows. □

Thus we conclude that deciding whether a given C-FSM has a PDS is PSPACE-Complete.

THEOREM 4.2. *Given a C-FSM M, checking the existence of a controllable* PDS *is* PSPACE-Complete.

Proof. First observe that the reduction presented in Proposition 3.1 generates a C-FSM. Thus we know that deciding whether an FSM has a controllable PDS is PSPACE-Hard. Consequently Propositions 3.1 and 4.7 together imply that the problem is PSPACE-Complete. □

5. CONCLUSIONS

Many algorithms for automatically generating test sequences from a single-port FSM M use input sequences that distinguish the states of M and several such automated test generation algorithms use PDSs. This paper has extended the concepts of PDSs to distributed testing. We explored the problem of deciding whether a multi-port FSM has a PDS, proving that this problem is generally undecidable. We also gave a class of multi-port FSMs (C-FSMs) for which the existence of a PDS is decidable but PSPACE-Complete. C-FSMs encapsulate a class of designs for which the PDS problem is decidable; if a developer can design a distributed system as a C-FSMs then this would simplify testing. Multi-port FSMs were initially used in the context of protocol conformance testing and so are relevant there. Section 4 also described other classes of systems for which multi-port FSMs can be used but there are likely to be classes of systems for which they are not appropriate.

These results are largely negative and might suggest that, although there are potential benefits to using PDSs, this is unlikely to be practical. However, we might have an upper bound ℓ on the length of PDSs of interest and for this case the problem is PSPACE-Hard and in EXPSPACE. We observed that we can use a set of separating sequences of polynomial size to distinguish the states of a multi-port FSM and so in practice a bound ℓ is likely to be a polynomial in the size of M. For such bounds the decision problem is NP-Complete. As far as we are aware there is no corresponding result for single-port FSMs but it is not hard to show that bounded PDS existence is PSPACE-Complete for single-port FSMs.

The results in this paper are for completely-specified multi-port FSMs. One can adapt the definition of a controllable PDS to partial multi-port FSMs by requiring that an input cannot be applied in a state where it is not specified. A number of the results (the undecidability result and the lower bounds) then immediately transfer to partial multi-port FSMs - since completely-specified multi-port FSMs form a special case. The upper-bound proofs can also be adapted for partial multi-port FSMs; when a non-deterministic Turing machine is guessing a controllable PDS it has the additional step of checking that an input is not applied in a state where it is not specified. Thus, the decidability and complexity results are the same for partial FSM.

There are several lines of future work. First, it would be interesting to investigate realistic conditions under which the problems studied can be solved in polynomial time. There may also be scope to develop heuristics for devising controllable PDSs and it is encouraging that although the PDS existence problem is PSPACE-Hard for single-port FSMs it has been found that SAT solvers can be effective in solving this problem [64]. Finally, there is the potential to develop new automated test generation algorithms for distributed testing.

REFERENCES

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

[2] Hennie, F. C. (1964) Fault-detecting experiments for sequential circuits. *Proceedings of Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, Princeton, New Jersey, November, pp. 95–110.

[3] Lee, D. and Yannakakis, M. (1996) Principles and

methods of testing finite-state machines - a survey. *Proceedings of the IEEE*, **84**, 1089–1123.

[4] Moore, E. P. (1956) Gedanken-experiments. In Shannon, C. and McCarthy, J. (eds.), *Automata Studies*. Princeton University Press.

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

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

[7] Tretmans, J. (2008) Model based testing with labelled transition systems. In Hierons, R. M., Bowen, J. P., and Harman, M. (eds.), *Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers*, Lecture Notes in Computer Science, **4949**, pp. 1–38. Springer.

[8] Duale, A. Y. and Uyar, M. U. (2004) A method enabling feasible conformance test sequence generation for EFSM models. *IEEE Transactions on Computers*, **53**, 614–627.

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

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

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

[12] 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 and Distributed Systems (FORTE 2004)*, Lecture Notes in Computer Science, **3235**, pp. 229–242. Springer-Verlag.

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

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

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

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

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

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

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

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

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

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

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

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

[25] Hierons, R., Merayo, M., and Nunez, M. (2008) Controllable test cases for the distributed test architecture. *Automated Technology for Verification and Analysis*, Lecture Notes in Computer Science, **5311**, pp. 201–215. Springer Berlin / Heidelberg.

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

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

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

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

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

[31] 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*, pp. 80–85.

[32] Aho, A., Dahbura, A., Lee, D., and Uyar, M. (1991) An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours. *IEEE Transactions on Communications*, **39**, 1604 –1615.

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

[34] Ural, H. and Zhu, K. (1993) Optimal length test sequence generation using distinguishing sequences. *IEEE/ACM Transactions on Networking*, **1**, 358–371.

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

[36] Petrenko, A., Yevtushenko, N., and v. Bochmann, G. (1996) Testing deterministic implementations from nondeterministic FSM specifications. *Testing of Communicating Systems*, Darmstadt, Germany, September 9–11, pp. 125–141. Chapman and Hall.

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

[38] Ural, H. (1992) Formal methods for test sequence generation. *Computers Communications*, **15**, 311–325.

[39] Hierons, R. M. and Türker, U. C. (2016) Distinguishing sequences for distributed testing: Adaptive distinguishing sequences. *The Computer Journal* , *In press*

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

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

[42] Hierons, R. M., Jourdan, G.-V., Ural, H., and Yenigun, H. (2009) Checking sequence construction using adaptive and preset distinguishing sequences, *IEEE International Conference on Software Engineering and Formal Methods*, Hanoi. November 23-27, pp. 157–166. IEEE, USA.

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

[44] Ural, H., Wu, X., and Zhang, F. (1997) On minimizing the lengths of checking sequences. *IEEE Transactions on Computers*, **46**, 93–99.

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

[46] Hopcroft, J. E. (1971) An n log n algorithm for minimizing the states in a finite automaton. International Symposium on the Theory of Machines and Computations, Technion, Haifa, Israel, August 16–19, pp. 189–196, Academic Press, INC., LONDON.

[47] Hierons, R. M. (2015) Generating complete controllable test suites for distributed testing. *IEEE Transactions on Software Engineering*, **41**, 279-293.

[48] Bannour, B., Escobedo, J. P., Gaston, C., and Gall, P. L. (2012) Off-line test case generation for timed symbolic model-based conformance testing. *24th IFIP WG 6.1 International Conference on Testing Software and Systems*, Aalborg, Denmark, November 19-21, Lecture Notes in Computer Science, pp. 119–135. Springer.

[49] Sokolovskii, M. (1971) Diagnostic experiments with automata. *Kibernetica*, **7**, 44–49.

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

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

[52] Manweiler, J., Agarwal, S., Zhang, M., Roy Choudhury, R., and Bahl, P. (2011) Switchboard: A matchmaking system for multiplayer mobile games. *International Conference on Mobile Systems, Applications, and Services*, Bethesda, MD, USA, June 28 - July 01, pp. 71–84, ACM, USA.

[53] Andersson, A. (2012) Multiplayer Game Server for Turn-Based Mobile Games in Erlang. PhD thesis Uppsala universitet/Institutionen för informationsteknologi, Sweden.

[54] Maheswaran, M., Braun, T. D., and Siegel, H. J. (1999) Heterogeneous distributed computing. *Encyclopedia of Electrical and Electronics Engineering*, pp. 679–690. John Wiley.

[55] Xiang, Y. and Levitin, G. (2011) Service task partition and distribution in star topology computer grid subject to data security constraints. *Rel. Eng. & Sys. Safety*, **96**, 1507–1514.

[56] (GIMPS), Great Internet Mersenne Prime Search (2015). http://www.mersenne.org/

[57] Elastic compute cloud (amazon EC2), A. (2015). https://aws.amazon.com/documentation/ec2/

[58] Crago, S. P., Dunn, K., Eads, P., Hochstein, L., Kang, D., Kang, M., Modium, D., Singh, K., Suh, J., and Walters, J. P. (2011) Heterogeneous cloud computing. *2011 IEEE International Conference on Cluster Computing (CLUSTER)*, Austin, Texas, USA, 26-30 September, pp. 378–385, IEEE, USA.

[59] Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., and Kozuch, M. A. (2012) Heterogeneity and dynamicity of clouds at scale: Google trace analysis. *ACM Symposium on Cloud Computing*, San Jose, CA, USA, October 14 - 17, pp. 7:1–7:13, ACM, USA.

[60] Frederic Magoules, Tan, K.-A., Jie Pan (2009) *Introduction to Grid Computing*, 2 edition. CRC Press.

[61] Minnebo, W. and Van Houdt, B. (2012) Pull versus push mechanism in large distributed networks: Closed form results. *International Teletraffic Congress*, Krakow, Poland, Sept, pp. 1–8.

[62] Huang, H. and Wang, L. (2010) P&P: A combined push-pull model for resource monitoring in cloud computing environment. *IEEE International Conference on Cloud Computing*, Miami, Florida, USA, July, 5–10, pp. 260–267, IEEE, USA

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

[64] Güniçen, C., Türker, U. C., Ural, H., and Yenigün, H. (2012) Generating preset distinguishing sequences using SAT. *Computer and Information Sciences II*, pp. 487–493. Springer, London.