
Article:
Hierons, R.M. (2017) Testing from partial finite state machines without harmonised traces. IEEE Transactions on Software Engineering, 43 (11). pp. 1033-1043. ISSN: 0098-5589

https://doi.org/10.1109/TSE.2017.2652457

Testing from Partial Finite State Machines without Harmonised Traces

Robert M. Hierons, *Senior Member, IEEE*

Abstract—This paper concerns the problem of testing from a partial, possibly non-deterministic, finite state machine (FSM) $\mathcal{S}$. Two notions of correctness (quasi-reduction and quasi-equivalence) have previously been defined for partial FSMs but these, and the corresponding test generation techniques, only apply to FSMs that have harmonised traces. We show how quasi-reduction and quasi-equivalence can be generalised to all partial FSMs. We also consider the problem of generating an m-complete test suite from a partial FSM $\mathcal{S}$: a test suite that is guaranteed to determine correctness as long as the system under test has no more than m states. We prove that we can complete $\mathcal{S}$ to form a completely-specified non-deterministic FSM $\mathcal{S}'$ such that any m-complete test suite generated from $\mathcal{S}'$ can be converted into an m-complete test suite for $\mathcal{S}$. We also show that there is a correspondence between test suites that are reduced for $\mathcal{S}$ and $\mathcal{S}'$ and also that are minimal for $\mathcal{S}$ and $\mathcal{S}'$.

Index Terms—Software engineering/software/program verification, software engineering/testing and debugging, systems and software, checking experiment, partial finite state machine.

1 Introduction

Testing is probably the most important approach to verification and validation but is typically manual, error prone and expensive. There has thus been much interest in automating software testing, one of the most promising approaches (called *model-based testing (MBT)*) being to base automation on a model. Many MBT methods take as input a state-based model expressed as a (possibly non-deterministic) finite state machine (FSM). While the tester might not produce an FSM, a tool might map a model to an FSM. There has thus been significant interest in testing from an FSM [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. Industrial experience has shown that the use of MBT can significantly reduce the cost of testing [11].

In the context of testing from an FSM, there has been much interest in automatically generating a test suite with a given fault detection power. This class of problem has been formalised in terms of a fault model[1]: a set $\mathcal{F}$ of FSMs such that the tester believes that the *system under test (SUT)* behaves like an unknown element of $\mathcal{F}$ (see, for example, [13]). Given fault model $\mathcal{F}$, a test suite $\mathcal{T}$ is a *checking experiment* if all faulty elements of $\mathcal{F}$ fail $\mathcal{T}$. The standard fault model is the set of FSMs with the same input and output alphabets as the specification FSM $\mathcal{S}$ and at most m states (for some given m). A test suite $\mathcal{T}$ is said to be *m-complete* if it is a checking experiment for this fault model and many methods have been devised to generate m-complete test suites [4], [8], [9], [14], [15]. One of the advantages of using m-complete test suites is that they have guaranteed fault detection ability: we know that if the SUT passes an m-complete test suite then either the SUT is correct or it has more than m states. In practice the value of m chosen will depend on the judgment of the tester and this parameter provides a trade-off between test suite size and effectiveness.

An FSM M is completely-specified if for every state s and input x the behaviour of M when it receives x in state s is specified; otherwise it is partial. Almost all work on generating m-complete test suites has assumed that the specification $\mathcal{S}$ is completely-specified. However, it has been observed that in practice many FSMs are partial [10]. A partial FSM might be used, for example, for a component that receives input from other components since this places restrictions on the inputs received [10]. Sometimes, when an FSM specification $\mathcal{S}$ has no transition with input x from state s, this means either that x should not be applied in state s or that the environment will never supply input x when $\mathcal{S}$ is in state s [10]. In such cases, in testing we should not apply x when $\mathcal{S}$ is in state s. These observations have led to the development of a few techniques for generating m-complete test suites from a partial FSM [10], [16], [17], [18].

In order to reason about test effectiveness, it is normal to assume that the SUT behaves like an unknown completely-specified FSM $\mathcal{I}$. Having made this assumption, it is possible to define correctness in terms of a relationship between two models: the specification FSM $\mathcal{S}$ and the unknown FSM $\mathcal{I}$ that models the SUT. This relationship is typically said to be an *implementation relation*. In this paper we consider the two implementation relations, quasi-equivalence and quasi-reduction, normally used when testing from a partial FSM. These were previously developed for testing from a partial

• *R. M. Hierons is with the Department of Computer Science, Brunel University London, UK.*
E-mail: rob.hierons@brunel.ac.uk

1. This is similar to the notion of a test hypothesis [12].

deterministic FSM (DFSM) and from an observable[2] non-deterministic FSM with harmonised traces[3] [9], [10], [17]. Under quasi-equivalence and quasi-reduction, if an input x is not specified in some state s that can be reached by input sequence $\bar{x}$ then the all outputs are allowed in response to x being received after $\bar{x}$. We show that the current definitions of quasi-equivalence and quasi-reduction have properties that mean that they are not suitable for testing from a partial non-deterministic FSM that does not have harmonised traces. While this is not surprising, since these implementation relations are not designed to deal with a wider class of partial FSMs, these results show that previous devised techniques for testing from a partial non-deterministic FSM can only safely be applied to FSMs that are observable and have harmonised traces; even if such test generation techniques are sound for the general case, there is the need to prove this. We generalise these implementation relations to form what we call generalised-quasi-equivalence and generalised-quasi-reduction.

An FSM M has harmonised traces if for every input sequence $\bar{x}$, input x and pair of states s, s' that can be reached from the initial state of M by applying $\bar{x}$, we have that either x is specified in states s and s' or it is not specified in either of these states. Let us suppose that we have to apply input sequence $\bar{x}$ if we wish to take M to state s and M does not have harmonised traces. Further, let us suppose that $\bar{x}$ can also take M to state s' and that input x is specified in s but not s'. The problem caused in testing is as follows: we cannot use the input sequence $\bar{x}x$ to test what happens when x is applied in state s because $\bar{x}$ can also take M to state s' and we should not apply x when M is in s'. This helps motivate the interest in FSMs with harmonised traces since it avoids scenarios such as the one above and allows us to safely use test sequences (input sequences) chosen to achieve certain objectives such as applying x in state s.

We make two main changes in order to generalise quasi-reduction and quasi-equivalence. For the first change, let us suppose that M produces output sequence $\bar{y}$ if $\bar{x}$ takes M to state s and produces output sequence $\bar{y}'$ if $\bar{x}$ takes M to state s'. If we have that $\bar{y} \neq \bar{y}'$ then the above scenario is not problematic if we allow testing to be adaptive, with the next input being chosen based on the input/output sequence so far observed. Specifically, a test case can choose to apply input x if $\bar{y}$ is produced in response to $\bar{x}$ but not if $\bar{y}'$ is produced. This observation immediately allows us to weaken the notion of harmonised traces to require the following: for every input/output sequence $\bar{x}/\bar{y}$, input x and pair of states s, s' that can be reached from the initial state of M by $\bar{x}/\bar{y}$, we have that either x is specified in states s and s' or it is not specified in either of these states. However, requiring this condition would not generalise quasi-equivalence and quasi-reduction to all partial FSMs. The second change, which does achieve this, is to allow scenarios in which input/output sequence $\bar{x}/\bar{y}$ takes M to states s and s' and input x is specified in state s but not s' and, in such cases, to say that *any* behaviour is allowed if x is received after $\bar{x}/\bar{y}$. The motivation for this is that if x is received after $\bar{x}/\bar{y}$ then it may well have been received in state s' and in such situations one would say that all behaviours are allowed. In addition, any behaviour is acceptable in such a situation since x will not be received after $\bar{x}/\bar{y}$; it does not matter how the SUT behaves after $\bar{x}/\bar{y}$. We will see that the proposed test generation method will also ensure that no test applies x after $\bar{x}/\bar{y}$. These two changes, when combined, allow us to generalise quasi-reduction and quasi-equivalence to allow any partial FSM as specification. Naturally, if x is not specified in a state reached by $\bar{x}/\bar{y}$ then we need to ensure that no test case can apply x after $\bar{x}/\bar{y}$.

Having generalised the notions of quasi-reduction and quasi-equivalence, we focus on testing for generalised-quasi-reduction and explore the problem of generating an m-complete test suite from a deterministic or non-deterministic partial FSM $\mathcal{S}$. Rather than devising a new test suite generation method, we generalise the method of Petrenko et al. [9] for observable FSMs with harmonised traces, which is to define a transformation that produces a completely-specified FSM $\mathcal{M}(\mathcal{S})$ such that any m-complete test suite $\mathcal{T}$ for $\mathcal{M}(\mathcal{S})$ can be transformed to produce an m-complete test suite $\mathcal{T}_1$ for $\mathcal{S}$. The FSM $\mathcal{M}(\mathcal{S})$ returned is a non-deterministic FSM even if $\mathcal{S}$ is deterministic. This result shows that methods for generating an m-complete test suite from a completely-specified FSM can be adapted to generate an m-complete test suite from a partial FSM. In addition, the techniques for testing from completely-specified FSMs are typically less involved. Note that even if one or more test cases in $\mathcal{T}$ can lead to an input x being applied after a trace $\bar{x}/\bar{y}$ such that x is not specified in some state of M reached by $\bar{x}/\bar{y}$, importantly no test case in $\mathcal{T}_1$ can have this property.

We explore an additional issue not previously considered, which is whether efficient/reduced test suites for $\mathcal{M}(\mathcal{S})$ lead to efficient/reduced test suits for $\mathcal{S}$. We say that an m-complete test suite $\mathcal{T}$ is reduced if, for all $t \in \mathcal{T}$, the test suite formed from $\mathcal{T}$ by replacing t by a proper prefix of t is not m-complete. This says that if we make $\mathcal{T}$ smaller, by removing a test case or replacing a test case t with a proper prefix of t, then the resultant test suite cannot be m-complete. We prove that if an m-complete test suite $\mathcal{T}$ is reduced for $\mathcal{M}(\mathcal{S})$ then the test suite $\mathcal{T}_1$ generated from $\mathcal{T}$ is also reduced for $\mathcal{S}$. We also prove that a smallest m-complete test suite $\mathcal{T}$ for $\mathcal{M}(\mathcal{S})$ is transformed into a smallest m-complete test suite for $\mathcal{S}$. We therefore show that not only can we use techniques for generating an m-complete test suite from a completely-specified FSM but also that the notion of minimality is preserved and the transformation does not introduce redundancy.

2. FSM $\mathcal{S}$ is observable if for every state s and input/output pair x/y, there is at most one transition from s with label x/y.

3. A non-deterministic FSM $\mathcal{S}$ has harmonised traces if, for each input sequence $\bar{x}$ that is the input portion of the label of a path of $\mathcal{S}$, the same sets of inputs are specified in all states reached by $\bar{x}$.

The results in this paper have a number of practical consequences. First, they provide implementation relations that are not restricted to observable specifications that have harmonised traces and so make it possible to formally reason about test effectiveness when testing from any partial FSM. They thus open up the possibility to develop test generation methods that produce m-complete test suites when testing from a partial FSM that does not have harmonised traces. Second, the results show that any method that returns an m-complete test suite for a completely-specified non-deterministic FSM can be used when testing from a partial FSM. This result is of interest since there are many techniques for testing from a completely-specified FSM[4] (see, for example, [8], [9], [14], [19], [20], [21], [22], [23], [24], [25], [26], [27]) and such techniques tend to be less involved than those devised for partial FSMs. Third, we show that this approach, of using methods for testing from completely-specified FSMs, does not introduce redundancy. In addition, by providing implementation relations that apply to all partial FSMs, this work should provide a formal framework within which it is possible to reason about other problems related to partial FSMs. For example, if we wish to devise methods for transforming a partial FSM specification M into an observable partial FSM M' then we might wish to know that the same set of SUTs are correct implementations of M and M'.

The paper is structured as follows. We start by giving preliminary definitions in Section 2. In Section 3 we define implementation relations and in Section 4 we define test cases for partial FSMs. Section 5 then shows how the problem of generating an m-complete test suite from a partial FSM can be mapped to the corresponding problem for a completely-specified FSM. Section 6 then explores efficiency issues. Finally, in Section 7 we summarise the paper and discuss possible future work.

2 PRELIMINARIES

Throughout this paper we use X to denote the set of inputs that the SUT might receive and Y to denote the set of outputs. Thus, if $x \in X$ and $y \in Y$ then x/y is an input/output pair. If $x_1, \ldots, x_k \in X$ and $y_1, \ldots, y_k \in Y$ then $\bar{\rho} = x_1/y_1 \ldots x_k/y_k$ is an *input/output sequence*, also called a *trace*, $x_1 \ldots x_k$ is the *input portion* of $\bar{\rho}$, and $y_1 \ldots y_k$ is the *output portion* of $\bar{\rho}$. Given a set A we will use A^* to denote the set of finite sequences of elements of A, with this including the empty sequence denoted ϵ. If $\bar{x} = x_1 \ldots x_k \in X^*$ and $\bar{y} = y_1 \ldots y_k \in Y^*$ then $\bar{x}$ is an input sequence, $\bar{y}$ is an output sequence, and we also use $\bar{x}/\bar{y}$ to represent the trace $x_1/y_1 \ldots x_k/y_k$.

Given a set A of pairs we let $\pi_1(A)$ and $\pi_2(A)$ denote the sets of projections of elements of A and so $\pi_1(A) = \{a | \exists b.(a,b) \in A\}$ and $\pi_2(A) = \{b | \exists a.(a,b) \in A\}$. The following defines non-deterministic finite state machines.

4. Many of these are based on what is called state-counting

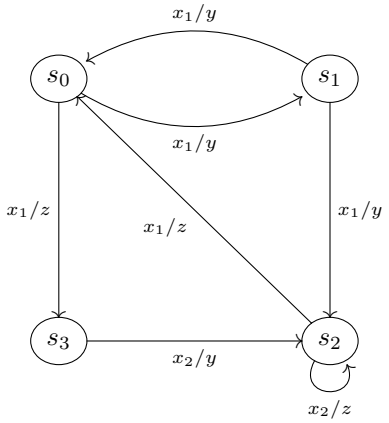

Fig. 1. Partial FSM $\mathcal{S}_1$

Definition 1: A partial non-deterministic finite state machine (FSM) is defined by a tuple (S, s_0, X, Y, h) in which:

1) S is the finite set of states.
2) $s_0 \in S$ is the initial state.
3) X is the finite input alphabet.
4) Y is the finite output alphabet.
5) h is the transition relation of type $S \times X \leftrightarrow S \times Y$.

We say that that x is *defined* in state s if there exist s' and y such that h relates (s,x) and (s',y); this is the case if $((s,x),(s',y)) \in h$. We let $dom\ h$ denote the set of pairs (s,x) on which h is defined. If h is defined on (s,x) then we let $h(s,x)$ denote the set of pairs (s',y) such that $((s,x),(s',y)) \in h$: $h(s,x) = \{(s',y) | ((s,x),(s',y)) \in h\}$. If $(s',y) \in h(s,x)$ then tuple $(s,s',x/y)$ is said to be a *transition* of M. M is *observable* if for all $(s,x) \in S \times X$ and $y \in Y$ there is at most one state s' such that $(s',y) \in h(s,x)$. If $|h(s,x)| \leq 1$ for all $s \in S$ and $x \in X$ then M is a deterministic FSM (DFSM). We will use 'DFSM' for finite state machines that are deterministic and 'FSM' whenever a finite state machine might be deterministic or non-deterministic. If $|h(s,x)| \geq 1$ for all $s \in S$ and $x \in X$ then M is *completely-specified*. Note that M is still a partial non-deterministic finite state machine (it satisfies the conditions of Definition 1) and so completely-specified FSMs are a special case of partial FSMs. Figure 1 gives an example of a partial FSM $\mathcal{S}_1$ with input alphabet $\{x_1, x_2\}$ and output alphabet $\{y, z\}$.

A sequence $(s, s_1, x_1/y_1)(s_1, s_2, x_2/y_2) \ldots (s_{a-1}, s_a, x_a/y_a)$ of consecutive transitions of M is a *path* that has *starting state* s, *label* $x_1/y_1 \ldots x_a/y_a$, and *ending state* s_a. Given state s of M we let $L_M(s)$ denote the set of labels of paths of M that have starting state s and we let $L(M) = L_M(s_0)$ denote the *set of traces of* M. Given trace $\bar{\rho} \in L_M(s)$ we let s-after-$\bar{\rho}$ denote the set of states s' such that M has a path that has starting state s, label $\bar{\rho}$ and ending state s'. Thus, s-after-$\bar{\rho}$ denotes the set of states of M that can be reached from s through trace $\bar{\rho}$. Given completely-specified FSMs $\mathcal{I}$ and $\mathcal{S}$ with

the same input alphabets, we say that $\mathcal{I}$ is a *reduction* of $\mathcal{S}$ if $L(\mathcal{I}) \subseteq L(\mathcal{S})$. Most methods for testing from a completely-specified FSM test for reduction, although there has been some work that looks at testing for equivalence [24].

We can extend the function h to take input sequences as follows: $h(s, \epsilon) = \{(s, \epsilon)\}$ and $h(s, x\bar{x}) = \{(s'', y\bar{y}) | \exists s' \in S.(s', y) \in h(s, x) \wedge (s'', \bar{y}) \in h(s', \bar{x})\}$. For example, in $\mathcal{S}_1$ we have that $h(s_0, x_1) = \{(s_1, y), (s_3, z)\}$ and so, since there is no transition with input x_1 leaving state s_3, $h(s_0, x_1x_1) = \{(s_0, yy), (s_2, yy)\}$. To obtain the possible output sequences in response to input sequence $\bar{x}$ from s we simply take the projection $\pi_2(h(s, \bar{x}))$. For example, in $\mathcal{S}_1$ we have that $\pi_2(h(s_0, x_1)) = \pi_2(\{(s_1, y), (s_3, z)\}) = \{y, z\}$ and $\pi_2(h(s_0, x_1x_1)) = \pi_2(\{(s_0, yy), (s_2, yy)\}) = \{yy\}$.

Throughout this paper we assume that the specification is a partial FSM $\mathcal{S}$ and that the SUT behaves like some unknown completely-specified FSM $\mathcal{I}$ with the same input and output alphabets as $\mathcal{S}$.

3 Implementation relations

This section considers the notions of correctness (implementation relations) used when testing from a partial FSM $\mathcal{S}$. In this we use $\Omega_S(s)$ to denote the set of input sequences that are input portions of traces in $L_S(s)$ and let $\Omega(\mathcal{S})$ denote $\Omega_S(s_0)$ [17]. In the following, $\mathbb{P}(X^*)$ denotes the powerset of X^*.

Definition 2: Given FSM $\mathcal{S}$, $\Omega_\mathcal{S}$ is the smallest function from the state set S of $\mathcal{S}$ to $\mathbb{P}(X^*)$ such that the following hold for all $s \in S$.

1) $\epsilon \in \Omega_\mathcal{S}(s)$
2) If $(s', y) \in h(s, x)$ and $\bar{x} \in \Omega_\mathcal{S}(s')$ then $x\bar{x} \in \Omega_\mathcal{S}(s)$.

If s_0 is the initial state of $\mathcal{S}$ then $\Omega(\mathcal{S})$ denotes $\Omega_\mathcal{S}(s_0)$.

Consider again the FSM $\mathcal{S}_1$. Here we have that $x_1x_1 \in \Omega(\mathcal{S}_1)$ since there is a path $(s_0, s_1, x_1/y)(s_1, s_2, x_1/y)$ from s_0 whose label has input portion x_1x_1. Since there is no transition from s_0 with input x_2, no input sequence starting with x_2 is in $\Omega(\mathcal{S}_1)$.

An FSM M is said to have *harmonised traces* if whenever two or more states of M can be reached from the initial state by a common input sequence, these states have the same sets of specified inputs. More formally, FSM M has harmonised traces if for all $\bar{x} \in \Omega_M(s_0)$ and all $s, s' \in \pi_1(h(s_0, \bar{x}))$, we have that (for all $x \in X$) $h(s, x)$ is defined if and only if $h(s', x)$ is defined.

Note that $\mathcal{S}_1$ does not have harmonised traces since input x_1 can take $\mathcal{S}_1$ to states s_1 and s_3 and x_2 is defined in s_3 but not s_1. Clearly, $\mathcal{S}_1$ also is not observable since there are transitions from s_1 with label x_1/y to states s_0 and s_2. Note that x_2 is defined in s_2 but not s_0 and so we cannot use the standard technique for transforming an FSM into an observable FSM; this transformation would form a state defined by the set $\{s_0, s_2\}$ of states and x_2 would be defined in this new state (since x_2 is defined in state s_2) despite x_2 not being defined in state s_0. Test techniques might then lead to test cases in which x_2 can be applied when $\mathcal{S}_1$ could be in state s_0.

We now give relations that have been defined for states of an observable partial FSM with harmonised traces [9]. States s_1 and s_2 of M are equivalent if the same sets of input sequence are defined from them and the corresponding output sequences are identical. We therefore have that s_1 and s_2 are *equivalent* if $\Omega_M(s_1) = \Omega_M(s_2)$ and for all $\bar{x} \in \Omega_M(s_1)$ we have that $\pi_2(h(s_1, \bar{x})) = \pi_2(h(s_2, \bar{x}))$. FSMs $\mathcal{S}$ and $\mathcal{I}$ are *equivalent* if their initial states are equivalent[5]. The following is clear.

Proposition 1: States s_1 and s_2 of FSM M are equivalent if and only if $L_M(s_1) = L_M(s_2)$.

Equivalence thus corresponds to the classical notion of equivalence for finite automata. Quasi-equivalence requires the SUT to be defined where the specification is defined and for such input sequences it requires that $\mathcal{I}$ and $\mathcal{S}$ have the same sets of traces [9], [17].

Definition 3: Given observable FSM M with harmonised traces, state s_1 is *quasi-equivalent* to state s_2 if $\Omega_M(s_2) \subseteq \Omega_M(s_1)$ and for all $\bar{x} \in \Omega_M(s_2)$ we have that $\pi_2(h(s_1, \bar{x})) = \pi_2(h(s_2, \bar{x}))$. Further, given observable FSMs $\mathcal{I}$ and $\mathcal{S}$ with harmonised traces, $\mathcal{I}$ is quasi-equivalent to $\mathcal{S}$ if the initial state of $\mathcal{I}$ is quasi-equivalent to the initial state of $\mathcal{S}$.

Finally, we have the notion of a quasi-reduction [9], [17], which relaxes quasi-equivalence by allowing the SUT to be less non-deterministic than the specification.

Definition 4: Given observable FSM M with harmonised traces, state s_1 is a *quasi-reduction* of state s_2 if $\Omega_M(s_2) \subseteq \Omega_M(s_1)$ and for all $\bar{x} \in \Omega_M(s_2)$ we have that $\pi_2(h(s_1, \bar{x})) \subseteq \pi_2(h(s_2, \bar{x}))$. Given observable FSMs $\mathcal{I}$ and $\mathcal{S}$ with harmonised traces and the same input and output alphabets, we say that $\mathcal{I}$ is a quasi-reduction of $\mathcal{S}$ if and only if the initial state of $\mathcal{I}$ is a quasi-reduction of the initial state of $\mathcal{S}$.

Since the SUT is completely-specified and we are interested in cases where the specification is partial, we will not want to test for equivalence. However, quasi-equivalence and quasi-reduction are potential implementation relations when testing from a partial FSM. Note that for DFSMs the notions of quasi-equivalence and quasi-reduction coincide and it is normal to use the term quasi-equivalence (see, for example, [10]).

Let us suppose that we take a partial specification $\mathcal{S}$ and create a completely-specified FSM $\mathcal{I}$ by adding transitions to S. Further, let us suppose that every transition added to $\mathcal{S}$, in forming $\mathcal{I}$, has a starting state s and input x such that x is not defined in state s of $\mathcal{S}$. The added transitions all correspond to state/input pairs (s, x) such that x should not be applied in state s and so we would expect $\mathcal{I}$ to be a correct implementation of $\mathcal{S}$ under either quasi-reduction or quasi-equivalence.

We now explore what happens if we remove the restriction to observable FSMs that have harmonised

5. We can use definitions regarding comparing two states of an FSM when comparing two FSMs by taking the disjoint union of these FSMs.

traces. Consider again the partial FSM $\mathcal{S}_1$ shown in Figure 1. Since $x_1/y\, x_1/y \in L(\mathcal{S}_1)$ we have that $x_1x_1 \in \Omega(\mathcal{S}_1)$. Now let us suppose that we try to complete $\mathcal{S}_1$ to form an FSM $\mathcal{I}_1$ that is a quasi-reduction of $\mathcal{S}_1$. By the definition of quasi-reduction, we must have that every trace produced by $\mathcal{I}_1$ in response to an element of $\Omega(\mathcal{S}_1)$ is also a trace of $\mathcal{S}_1$. Thus, every response of $\mathcal{I}_1$ to the input sequence $x_1x_1 \in \Omega(\mathcal{S}_1)$ must also be a response of $\mathcal{S}_1$ to x_1x_1. This cannot be the case since $\mathcal{I}_1$ has been formed by completing $\mathcal{S}_1$ and so must have at least one path whose label has input portion x_1x_1 and an output portion that starts with z. However, we formed $\mathcal{I}_1$ from $\mathcal{S}_1$ by adding transitions such that every transition added to $\mathcal{S}_1$ has a starting state s and input x such that x is not defined in state s of $\mathcal{S}_1$. We would therefore expect $\mathcal{I}_1$ to be a correct implementation of $\mathcal{S}_1$ if we were using a suitable implementation relation. A similar observation may be made with respect to quasi-equivalence but the situation is even worse, in the sense that there may be no completely-specified FSM that is quasi-equivalent to the specification.

Proposition 2: There exists a partial FSM $\mathcal{S}$ such that no completely-specified FSM is quasi-equivalent to $\mathcal{S}$.

Proof: We will prove that no completely-specified FSM is quasi-equivalent to $\mathcal{S}_1$. Proof by contradiction: let us suppose that completely-specified FSM $\mathcal{I}_1$ is quasi-equivalent to $\mathcal{S}_1$. First consider the input sequence $x_1 \in \Omega(\mathcal{S}_1)$. Since $\mathcal{I}_1$ is quasi-equivalent to $\mathcal{S}_1$ we have that $\mathcal{I}_1$ has a path with label x_1/z. However, since $\mathcal{I}_1$ is completely-specified this implies that $\mathcal{I}_1$ has a path with label $x_1/z\, x_1/z'$ for some output z'. This cannot be the label of a path of $\mathcal{S}_1$ and so, since $x_1x_1 \in \Omega(\mathcal{S}_1)$, this contradicts $\mathcal{I}_1$ being quasi-equivalent to $\mathcal{S}_1$ as required. □

There is nothing pathological about $\mathcal{S}_1$ and so the above observations and results suggest that the current definitions of quasi-equivalence and quasi-reduction for FSMs cannot be applied if we do not restrict attention to FSMs that are observable and have harmonised traces. This motivates our interest in devising new implementation relations and we will now explore how this issue can be eliminated by generalising these definitions.

The results regarding $\mathcal{S}_1$ are a consequence of having an input sequence x_1 that can lead to two different states (s_1 and s_3) such that there is an input x_1 that is specified in one of these states but not the other. The problem was that $\Omega_{\mathcal{S}}$ does not differentiate between the different traces that lead to these two states. Thus, the traditional definitions of quasi-equivalence and quasi-reduction are too strong when $\mathcal{S}$ does not have harmonised traces since to test for these notions of correctness one might have to apply an input x after a trace $\bar{\rho}$ such that the input portion of $\bar{\rho}$ is a defined input sequence but x is not specified in a state of $\mathcal{S}$ reached by $\bar{\rho}$. For example, when testing from $\mathcal{S}_1$ we should not apply input x_1 after trace x_1/z since x_1 is not specified in the state s_3 reached by x_1/z. However, x_1x_1 is a defined input sequence for $\mathcal{S}_1$ and so the traditional definitions of quasi-equivalence and quasi-reduction allow x_1x_1 to be applied, even if the response to the first x_1 is z. This goes against the philosophy behind quasi-equivalence and quasi-reduction and explains why these implementation relations were defined for FSMs with harmonised traces.

Let us suppose that if, after observing trace $\bar{\rho}$, the specification $\mathcal{S}$ might be in a state s where input x is not specified. Then x should not be supplied after $\bar{\rho}$ even if there is also a state s' that can be reached using $\bar{\rho}$ such that x is specified in s'. The reason for this is that x should not be supplied in the first scenario and we can't distinguish between these cases. This leads to the following definition of what we will call generalised-quasi-reduction in order to distinguish it from the earlier definition.

Definition 5: Given states s_1 and s_2 of FSM M, s_1 is a *generalised-quasi-reduction (GQR)* of s_2 if for all $\bar{\rho} \in L_M(s_1) \cap L_M(s_2)$ and $x \in X$, if $x \in \Omega_M(s)$ for all $s \in s_2$-after-$\bar{\rho}$ then:

1) for all $s' \in s_1$-after-$\bar{\rho}$ we have that $x \in \Omega_M(s')$; and
2) $\{y' \in Y | \bar{\rho}x/y' \in L_M(s_1)\} \subseteq \{y' \in Y | \bar{\rho}x/y' \in L_M(s_2)\}$.

Given FSMs $\mathcal{I}$ and $\mathcal{S}$ with the same input and output alphabets, we say that $\mathcal{I}$ is a GQR of $\mathcal{S}$ if and only if the initial state of $\mathcal{I}$ is a GQR of the initial state of $\mathcal{S}$.

In the above, $L_M(s_1) \cap L_M(s_2)$ is guaranteed to be non-empty since ϵ is in both $L_M(s_1)$ and $L_M(s_2)$. The first condition above requires that if the response to x after $\bar{\rho}$ is specified (for every state reached via $\bar{\rho}$) and $\bar{\rho}$ is a trace of $\mathcal{I}$ then the response of $\mathcal{I}$ to x after $\bar{\rho}$ is also specified. The second part requires that for such a trace $\bar{\rho}$ we have that all responses of $\mathcal{I}$ to x after $\bar{\rho}$ are also responses of $\mathcal{S}$ to x after $\bar{\rho}$. The above definition does not require us to consider the situation in which trace $\bar{\rho}$ reaches states s and s' from state s_2 of M and x is specified in state s but not state s'. The key is that the definition of GQR places no requirements of the behaviour of M if x is received after $\bar{\rho}$ is observed when testing from s_1: all possible outputs are allowed in response to x and from then on all outputs are allowed in response to any additional inputs received. This is consistent with how quasi-reduction and quasi-equivalence deal with the input of x after an input sequence $\bar{x}$ if x is not specified after $\bar{x}$. Naturally, in such situations x should not be applied after $\bar{\rho}$ in testing. Later we will see how test generation can avoid applying an input x in such situations.

Consider the FSM $\mathcal{S}_1$ and the above definition of GQR. We have that only one state (s_1) of $\mathcal{S}_1$ is reached by trace x_1/y and so an SUT that is a GQR of $\mathcal{S}_1$ must produce output y if it receives input x_1 after trace x_1/y. However, $\mathcal{S}_1$ has two possible states, s_0 and s_2, after $x_1/y\, x_1/y$ and x_2 is defined in only one of these. Thus, an SUT that is a GQR of $\mathcal{S}_1$ can produce any output if it receives x_2 after $x_1/y\, x_1/y$ has been observed.

Now consider the FSM $\mathcal{I}_1$ in Figure 2 in which, for example, we use a label such as $x_1/y, x_2/z$ to denote there being two corresponding transitions: one with label x_1/y and the other with label x_2/z. Since $\mathcal{S}_1$ has a path

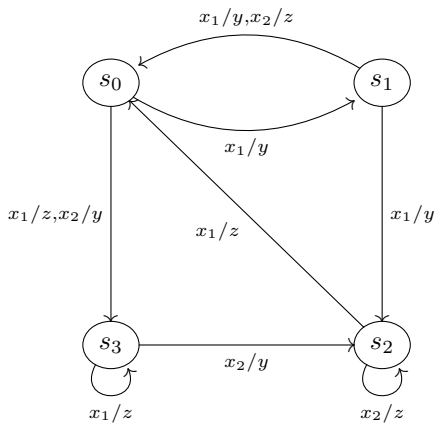

Fig. 2. Completely-specified FSM $\mathcal{I}_1$

whose label has input portion x_1x_1, in order for $\mathcal{I}_1$ to be a quasi-reduction of $\mathcal{S}_1$ we require that all traces in $L(\mathcal{I}_1)$, that have input portion x_1x_1, are also traces of $L(\mathcal{S}_1)$. Thus, since $\mathcal{I}_1$ has a path with label $x_1/z\,x_1/z$ and $\mathcal{S}_1$ does not, we know that $\mathcal{I}_1$ is not a quasi-reduction of $\mathcal{S}_1$. However, if we consider this trace $x_1/z x_1/z$ then we find that x_1/z takes $\mathcal{S}_1$ to state s_3 and x_1 is not specified in this state. Thus, under GQR the SUT is allowed to produce any output in response to x_1 after trace x_1/z. In fact, it is not too hard to show that $\mathcal{I}_1$ is a GQR of $\mathcal{S}_1$. This is because the only transitions added to $\mathcal{S}_1$ to create $\mathcal{I}_1$ involve state/input pairs (s, x) such that x is not specified in state s of $\mathcal{S}_1$; under GQR all behaviours are allowed for such pairs.

More generally, if we complete an FSM specification $\mathcal{S}$ to form a completely-specified FSM $\mathcal{I}$ in this way then $\mathcal{I}$ must be a GQR of $\mathcal{S}$.

Definition 6: FSM $\mathcal{I}$ is a *completion* of partial FSM $\mathcal{S} = (S, s_0, X, Y, h)$ if $\mathcal{I}$ is completely-specified and is equivalent to an FSM $\mathcal{S}' = (S', s_0, X, Y, h')$, with $S \subseteq S'$, that can be formed from $\mathcal{S}$ by adding states and transitions in a manner such that if x is defined in state s of $\mathcal{S}$ then $h'(s, x) = h(s, x)$.

Proposition 3: Given partial FSM $\mathcal{S}$, if $\mathcal{I}$ is a completion of $\mathcal{S}$ then $\mathcal{I}$ is a GQR of $\mathcal{S}$.

Proof: We use proof by contradiction and so assume that $\mathcal{I}$ is a completion of $\mathcal{S}$ and $\mathcal{I}$ is not a GQR of $\mathcal{S}$. By the definition of GQR, since $\mathcal{I}$ is completely-specified there must exist $\bar{\rho} \in L(\mathcal{S}) \cap L(\mathcal{I})$, input x that is specified in all states of $\mathcal{S}$ reached by paths with label $\bar{\rho}$, and output y such that $\bar{\rho}\, x/y \in L(\mathcal{I})$ and $\bar{\rho}\, x/y \notin L(\mathcal{S})$. Since $\mathcal{I}$ is a completion of $\mathcal{S}$, $\mathcal{I}$ is equivalent to an FSM $\mathcal{S}' = (S', s_0, X, Y, h')$, with $S \subseteq S'$, that can be formed from $\mathcal{S}$ by adding states and transitions in a manner such that if x is defined in state s of $\mathcal{S}$ then $h'(s, x) = h(s, x)$. But since x is defined in all states of $\mathcal{S}$ reached by paths with label $\bar{\rho}$, this implies that $\bar{\rho}\, x/y \in L(\mathcal{I})$ if and only if $\bar{\rho}\, x/y \in L(\mathcal{S})$. We therefore obtain a contradiction as required. □

We now define generalised-quasi-equivalence.

Definition 7: Given states s_1 and s_2 of FSM M, s_1 is *generalised-quasi-equivalent (GQE)* to state s_2 if for all $\bar{\rho} \in L_M(s_1) \cap L_M(s_2)$ and $x \in X$, if $x \in \Omega_M(s)$ for all $s \in s_2$-after-$\bar{\rho}$ then:

1) for all $s' \in s_1$-after-$\bar{\rho}$ we have that $x \in \Omega_M(s')$; and
2) $\{y' \in Y | \bar{\rho}x/y' \in L_M(s_1)\} = \{y' \in Y | \bar{\rho}x/y' \in L_M(s_2)\}$.

Given FSMs $\mathcal{I}$ and $\mathcal{S}$ with the same input and output alphabets, we say that $\mathcal{I}$ is generalised-quasi-equivalent to $\mathcal{S}$ if and only if the initial state of $\mathcal{I}$ is generalised-quasi-equivalent to the initial state of $\mathcal{S}$.

Similar to GQR, if we complete a partial FSM then we obtain a correct implementation.

Proposition 4: Given partial FSM $\mathcal{S}$, if $\mathcal{I}$ is a completion of $\mathcal{S}$ then $\mathcal{I}$ is GQE to $\mathcal{S}$.

These notions of correctness only consider the application of an input x after trace $\bar{\rho}$ if x is specified in all states of the specification that are reached by $\bar{\rho}$. If this is not the case (there is at least one state of the specification, reached by $\bar{\rho}$, in which x is not specified) then there are no constraints on the behaviour of the SUT. This restriction, to when we consider the response to x after $\bar{\rho}$, is an inevitable consequence of the requirement not to apply an input x in states where x is not defined. If this is too weak then the tester might consider producing a more complete specification. An alternative is to add probes to the SUT so that the test system can distinguish between different paths that have the same trace.

When testing from a completely-specified FSM the notions of correctness can be defined in terms of the traces of the specification. Thus, if specifications $\mathcal{S}$ and $\mathcal{S}'$ are completely-specified and have the same set of traces then an SUT is a reduction of $\mathcal{S}$ if and only if it is a reduction of $\mathcal{S}'$. However, for partial FSMs the underlying assumption is that if an input x is not specified in state s then x will not be applied in state s and this allows us to define FSMs $\mathcal{S}$ and $\mathcal{S}'$, with the same set of traces, such that there is some FSM that is a GQR of $\mathcal{S}$ but not $\mathcal{S}'$. To see this consider the FSMs in Figure 3. It is straightforward to see that these FSMs have the same traces. However, after x_1/y the first FSM $\mathcal{S}$ can be in a state where x_1 is not specified or in a state in which x_2 is not specified. Thus, under GQR the SUT can produce any output in response to either x_1 or x_2 after x_1/y. In contrast, after x_1/y the second FSM $\mathcal{S}'$ must be in a state q_1 in which both x_1 and x_2 are specified. Thus, if an SUT is a GQR of $\mathcal{S}'$ then after x_1/y it must produce output y in response to x_1 and z in response to x_2.

GQE and GQR reduce to quasi-equivalence and quasi-reduction if one considers observable FSMs with harmonised traces.

Proposition 5: If FSM M is observable and has harmonised traces and s_1 and s_2 are states of M then:

1) s_1 is a generalised-quasi-reduction of s_2 if and only if s_1 is a quasi-reduction of s_2.
2) s_1 is generalised-quasi-equivalent to s_2 if and only if s_1 is quasi-equivalent to s_2.

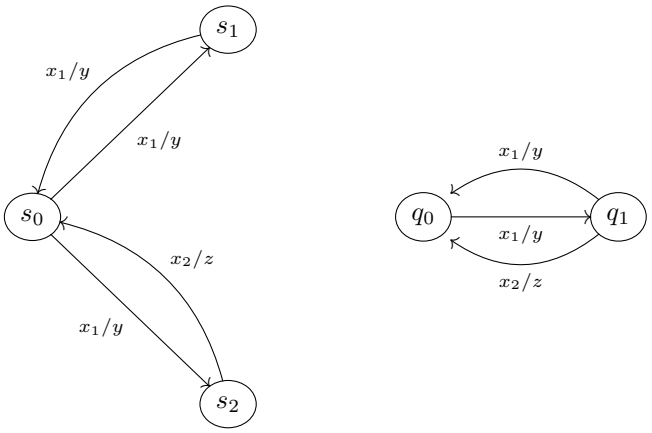

Fig. 3. FSMs $\mathcal{S}$ and $\mathcal{S}'$ with the same traces

Proof: We will prove the first part; the proof of the second is similar.

First assume that s_1 is a GQR of s_2 and we are required to prove that s_1 is a quasi-reduction of s_2. It is sufficient to prove that if $\bar{x}$ is a defined input sequence for s_2 then $\bar{x}$ is also a defined input sequence for s_1 and $\pi_2(h(s_1, \bar{x})) \subseteq \pi_2(h(s_2, \bar{x}))$. We will use proof by induction on the length of $\bar{x}$. The result immediately holds for the base case ϵ. We will assume that it holds for all sequences of length k or less and that $\bar{x}$ is a defined input sequence for s_2 that has length $k+1$. Thus, $\bar{x} = \bar{x}'x$ for an input sequence $\bar{x}'$ of length k and an input x.

By the induction hypothesis, $\bar{x}'$ is a defined input sequence for s_1 and $\pi_2(h(s_1, \bar{x}')) \subseteq \pi_2(h(s_2, \bar{x}'))$. Thus, there is a trace $\bar{\rho} \in L_M(s_1) \cap L_M(s_2)$ with input portion $\bar{x}'$. Since $\bar{x}'x$ is a defined input sequence for s_2 and M has harmonised traces, x is specified in all states reached from s_2 by $\bar{x}'$. Thus, since s_1 is a GQR of s_2 we know that x is specified in all states that can be reached from s_1 by $\bar{x}'$. Thus, $\bar{x}$ is a specified input sequence from s_1.

By the definition of quasi-reduction, it is now sufficient to prove that $\pi_2(h(s_1, \bar{x})) \subseteq \pi_2(h(s_2, \bar{x}))$. By the inductive hypothesis we know that $\pi_2(h(s_1, \bar{x}')) \subseteq \pi_2(h(s_2, \bar{x}'))$. If we consider some $\bar{\rho} \in \pi_2(h(s_1, \bar{x}')) \cap \pi_2(h(s_2, \bar{x}'))$ then, by Definition 5, we have $\{y' \in Y | \bar{\rho}x/y' \in L_M(s_1)\} \subseteq \{y' \in Y | \bar{\rho}x/y' \in L_M(s_2)\}$ The result therefore follows.

Now assume that s_1 is a quasi-reduction of s_2 and we are required to prove that s_1 is a GQR of s_2. We therefore assume that $\bar{\rho} \in L_M(s_1) \cap L_M(s_2)$, $x \in X$, and for all $s \in s_2$-after-$\bar{\rho}$ we have that $x \in \Omega_M(s)$ and we are required to prove that:

1) for all $s' \in s_1$-after-$\bar{\rho}$ we have that $x \in \Omega_M(s')$; and
2) $\{y' \in Y | \bar{\rho}x/y' \in L_M(s_1)\} \subseteq \{y' \in Y | \bar{\rho}x/y' \in L_M(s_2)\}$.

Let $\bar{x}$ denote the input portion of $\bar{\rho}$. Since s_1 is a quasi-reduction of s_2, $\Omega_M(s_2) \subseteq \Omega_M(s_1)$ and so $\bar{x}x$ is a defined input sequence for s_1. Thus, since M has harmonised traces, x is defined in all $s' \in s_1$-after-$\bar{\rho}$ and so the first property holds. The second property is immediate from the definition of s_1 being a quasi-reduction of s_2. □

GQR is appropriate when non-determinism corresponds to there being several acceptable behaviours. In this paper we focus on the problem of testing for GQR and later we prove (Proposition 6) that this is a suitable implementation relation. Note that unlike [9], we do not require the SUT to be deterministic. However, if the SUT is non-deterministic then it is necessary to run each test case used multiple times and the completeness of a test suite typically relies on the use of a fairness assumption.

4 Test cases and test suites

This section defines test cases and test suites for a partial FSM $\mathcal{S}$. A test case will be similar to what has been called an adaptive test case [17]. Adaptivity allows the tester to choose an input on the basis of the observed trace. This will be crucial where the response to input x, after input sequence $\bar{x}$, is specified after a trace $\bar{\rho}$ but not after another trace $\bar{\rho}'$; adaptivity allows the tester to apply x after $\bar{\rho}$ but not after $\bar{\rho}'$. A test case and the SUT will interact through synchronising on common actions until a failure occurs or the test case terminates. We follow the previously proposed approach of representing a test case as an acyclic partial FSM [17]. A test case will be deterministic in that at each state of the test case there is at most one input that is specified.

Definition 8: Given partial FSM $\mathcal{S}$, a *test case* for $\mathcal{S}$ is an observable partial acyclic FSM t, with transition relation h' and initial state t_0, that satisfies the following.

1) For each state s of t there is at most one input x such that $h'(s, x)$ is defined;
2) If $\bar{\rho} \in L(t)$ and x is defined in the unique state in t_0-after-$\bar{\rho}$ then $h(s, x)$ is defined in every state $s \in s_0$-after-$\bar{\rho}$;
3) If $\bar{\rho}x/y \in L(t)$ then $\{y' | \bar{\rho}x/y' \in L(\mathcal{S})\} = \{y' | \bar{\rho}x/y' \in L(t)\}$.

The application of a test case will operate as follows. If a test case t with transition relation h' is in state s then:

1) If no transitions leave s then testing terminates.
2) Otherwise, we find the input x that labels transitions from s, we apply x to the SUT, and observe the output y. t moves to the state s' such that $(s', y) \in h'(s, x)$; if there is no such state s' then testing terminates and a failure has been observed.

The first condition in Definition 8 requires that the next input is uniquely defined. The second condition requires that input x can only be applied if $\mathcal{S}$ *must* be in a state s such that $h(s, x)$ is defined and so we know that we can apply x. The last condition requires that the corresponding outputs of the test case and specification are the same. Thus, if an output not specified in t is observed then it corresponds to a failure.

We let $T(\mathcal{S})$ denote the set of all possible test cases for $\mathcal{S}$. $T(\mathcal{S})$ is exactly the test cases we might wish to use: if a test case is not in $T(\mathcal{S})$ then either it includes input after a trace not in $L(\mathcal{S})$ (so this does not help testing), or it could apply input in a state where this is not allowed, or

it has output that is not consistent with the specification. We will call a set of test cases a *test suite*.

We now define what it means for the SUT to pass/fail a test case from $T(\mathcal{S})$. Essentially, a model $\mathcal{I}$ of the SUT fails test case t if some execution of $\mathcal{I}$ with t leads to a trace that is not a trace of $\mathcal{S}$. This is the case if t has a path with label $\bar{\rho}$ to a state s where it applies input x such that the SUT can produce an output y in response to x after $\bar{\rho}$ such that $\bar{\rho}x/y \notin L(\mathcal{S})$ (or, equivalently, that $\bar{\rho}x/y \notin L(t)$).

Definition 9: Given completely-specified FSM $\mathcal{I}$ that models an SUT and (possibly partial) FSM $\mathcal{S}$ that acts as the specification, $\mathcal{I}$ *fails* a test case $t \in T(\mathcal{S})$ if and only if there exists $\bar{\rho}x/y \in L(\mathcal{I})$ and $y' \in Y$ such that $\bar{\rho}x/y' \in L(t)$ and $\bar{\rho}x/y \notin L(t)$. Otherwise, $\mathcal{I}$ *passes* t.

The following result shows that the notion of passing a test case has the expected relationship to GQR. In the following, $s_0^{\mathcal{I}}$ is the initial state of $\mathcal{I}$.

Proposition 6: Given completely-specified FSM $\mathcal{I}$ that models an SUT and partial FSM $\mathcal{S}$ that acts as the specification, $\mathcal{I}$ is a GQR of $\mathcal{S}$ if and only if for all $t \in T(\mathcal{S})$ we have that $\mathcal{I}$ passes t.

Proof: First let us suppose that $\mathcal{I}$ passes all test cases in $T(\mathcal{S})$ and we are required to prove that $\mathcal{I}$ is a GQR of $\mathcal{S}$. Let us suppose that $\bar{\rho} \in L(\mathcal{I}) \cap L(\mathcal{S})$, $x \in X$, and for all $s \in s_0$-after-$\bar{\rho}$ we have that $x \in \Omega_{\mathcal{S}}(s)$. Below we define a test case $t = t_{\mathcal{S}}(\bar{\rho}x/y)$. Essentially, $t_{\mathcal{S}}(\bar{\rho}_1)$ has trace $\bar{\rho}_1$ and 'completes' this by allowing other outputs where these are consistent with the specification. The important point is that $\bar{\rho}_1$ is a trace of $t_{\mathcal{S}}(\bar{\rho}_1)$ and $t_{\mathcal{S}}(\bar{\rho}_1)$ is a test case in $T(\mathcal{S})$.

Test case $t_{\mathcal{S}}(\bar{\rho}_1)$ has initial state t_0, a path with label $\bar{\rho}_1$, and for every prefix $\bar{\rho}_2x/y$ of $\bar{\rho}_1$ we have that: for all $y' \in Y \setminus \{y\}$ there is a transition with label x/y' from the unique state in t_0-after-$\bar{\rho}_2$ to a state s' if and only if $\bar{\rho}_2x/y'$ is a trace of $\mathcal{S}$[6]. Here, s' is a 'terminating' state of $t_{\mathcal{S}}(\bar{\rho}_1)$: no transitions have s' as a starting state.

Since $\bar{\rho} \in L(\mathcal{I}) \cap L(\mathcal{S})$ and $\mathcal{I}$ is completely-specified, for all $s \in s_0^{\mathcal{I}}$-after-$\bar{\rho}$ we have that $x \in \Omega_{\mathcal{I}}(s)$. By the definition of GQR it is sufficient to prove that $\{y' \in Y | \bar{\rho}x/y' \in L(\mathcal{I})\} \subseteq \{y' \in Y | \bar{\rho}x/y' \in L(\mathcal{S})\}$. However, this follows from $\mathcal{I}$ passing $t_{\mathcal{S}}(\bar{\rho}x/y)$ as required.

Now let us suppose that $\mathcal{I}$ is a GQR of $\mathcal{S}$ and we are required to prove that $\mathcal{I}$ passes all test cases in $T(\mathcal{S})$. Proof by contradiction: let us suppose that $\mathcal{I}$ fails test case $t \in T(\mathcal{S})$ and so there is some trace, that can be observed when testing $\mathcal{I}$ with t, that is not a trace of $\mathcal{S}$. Let us suppose that $\bar{\rho}$ is some minimal such trace and so $\bar{\rho} = \bar{\rho}'x/y$ for some $x \in X$, $y \in Y$ and $\bar{\rho}' \in L(\mathcal{I}) \cap L(\mathcal{S})$. By the definition of $T(\mathcal{S})$, for all $s \in s_0$-after-$\bar{\rho}$ we have that $x \in \Omega_{\mathcal{S}}(s)$. But this contradicts $\mathcal{I}$ being a GQR of $\mathcal{S}$. □

The notion of passing a test case is entirely natural (all observations that can be made when testing the SUT are allowed by the specification) and the notion of a test case is quite general (it allows test cases to be adaptive). This result thus demonstrates the suitability of the definition of GQR given in this paper. In Section 7 we briefly discuss how the overall approach might be extended to GQE.

6. Recall that the traces of a test case are traces of the specification; if a different output is produced then we have failure.

5 Generating an m-complete test suite

This section explores test suite generation. We first say what it means for a test suite to be m-complete.

Definition 10: Given partial FSM $\mathcal{S}$ that acts as the specification and integer m, test suite $\mathcal{T} \subseteq T(\mathcal{S})$ is m-complete for $\mathcal{S}$ if for every completely-specified FSM $\mathcal{I}$ with the same input and output alphabets as $\mathcal{S}$ and at most m states we have that if $\mathcal{I}$ is not a GQR of $\mathcal{S}$ then $\mathcal{I}$ fails some test case from $\mathcal{T}$.

The following, previously defined [9] construction, completes a partial FSM $\mathcal{S}$ to form a completely-specified FSM $\mathcal{M}(\mathcal{S})$.

Definition 11: Given FSM $\mathcal{S} = (S, s_0, X, Y, h)$, $\mathcal{M}(\mathcal{S})$ is the FSM $(S \cup \{e\}, s_0, X, Y, h')$ in which $e \notin S$ and h' is defined by the following:

1) If $s \neq e$ and $(s, x) \in dom\ h$ then $h'(s, x) = h(s, x)$.
2) If $s = e$ or $(s, x) \notin dom\ h$ then $h'(s, x) = \{(e, y) | y \in Y\}$.

The FSM $\mathcal{S}_1$ (also shown in Figure 1) and its completion are shown in Figure 4. As before, for example, the label $x_2/y, x_2/z$ on an arc means that there are two transitions, with labels x_2/y and x_2/z, with the same start and end states.

The FSM $\mathcal{M}(\mathcal{S})$ (Definition 11) behaves like $\mathcal{S}$ until it receives an input x in a state s such that h is not defined on (s, x) and it then moves to the new state e and all outputs are possible in response to an input. We now show how testing from partial FSM $\mathcal{S}$ for GQR relates to testing for reduction from the (completely-specified) FSM $\mathcal{M}(\mathcal{S})$; this generalises the result of Petrenko et al. [9] that was for observable FSMs with harmonised traces.

Proposition 7: Given partial FSM $\mathcal{S}$ that forms the specification and completely-specified FSM $\mathcal{I}$ that models an SUT, $\mathcal{I}$ is a GQR of $\mathcal{S}$ if and only if $\mathcal{I}$ is a reduction of $\mathcal{M}(\mathcal{S})$.

Proof: First let us suppose that $\mathcal{I}$ is a GQR of $\mathcal{S}$ and we are required to prove that $\mathcal{I}$ is a reduction of $\mathcal{M}(\mathcal{S})$. We will prove that every trace of $\mathcal{I}$ is also a trace of $\mathcal{M}(\mathcal{S})$. Proof by contradiction: let us suppose that there are traces of $\mathcal{I}$ that are not in $L(\mathcal{M}(\mathcal{S}))$ and let $\bar{\rho}$ be a minimal such trace and so $\bar{\rho} = \bar{\rho}'x/y$ for some trace $\bar{\rho}'$, $x \in X$, and $y \in Y$. Trace $\bar{\rho}$ cannot take $\mathcal{M}(\mathcal{S})$ to state e, since in this case all outputs would be possible when $\mathcal{M}(\mathcal{S})$ is given x after $\bar{\rho}'$ (contradicting the minimality of $\bar{\rho}$). Thus, $\bar{\rho}'$ is a trace of $\mathcal{S}$ and in $\mathcal{S}$ the response to x is specified in all states reached by $\bar{\rho}'$. But, by definition, since $\bar{\rho} = \bar{\rho}'x/y$ is a trace of $\mathcal{I}$ and $\mathcal{I}$ is a GQR of $\mathcal{S}$, we have that $\bar{\rho}$ is a trace of $\mathcal{S}$. It is now sufficient to observe that $L(\mathcal{S}) \subseteq L(\mathcal{M}(\mathcal{S}))$.

Now let us suppose that $\mathcal{I}$ is a reduction of $\mathcal{M}(\mathcal{S})$ and we are required to prove that $\mathcal{I}$ is a GQR of $\mathcal{S}$. Proof

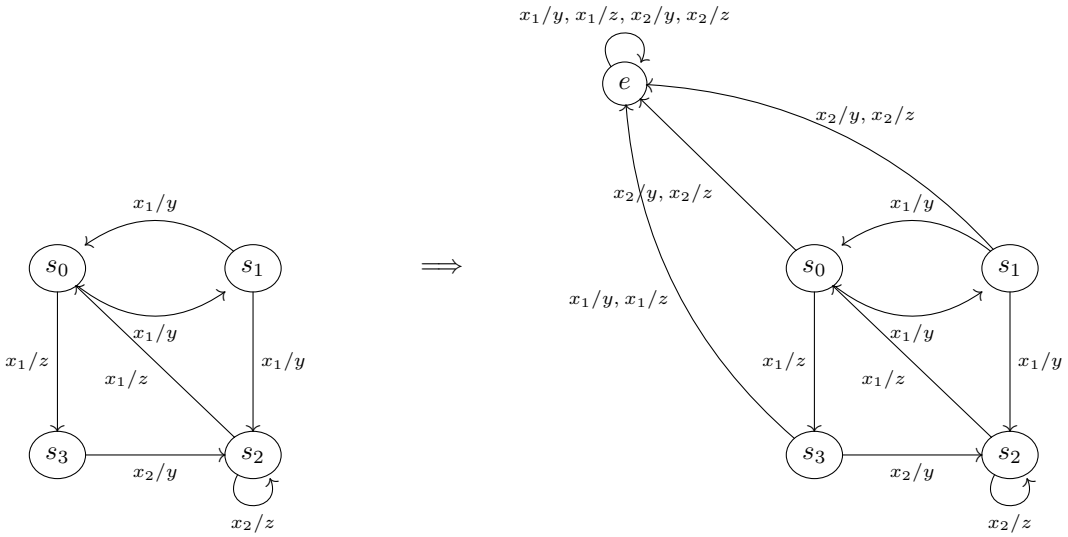

Fig. 4. Partial FSM $\mathcal{S}_1$ and its completion $\mathcal{M}(\mathcal{S}_1)$

by contradiction: let us suppose that $\mathcal{I}$ is not a GQR of $\mathcal{S}$. Since $\mathcal{I}$ is completely-specified we have some $\bar{\rho}' \in L(\mathcal{I}) \cap L(\mathcal{S})$, $x \in X$, and $y \in Y$ such that the response of $\mathcal{S}$ to x is specified in all states reached by $\bar{\rho}'$ and $\bar{\rho}'x/y \in L(\mathcal{I}) \setminus L(\mathcal{S})$. But, by definition, since the response of $\mathcal{S}$ to x after $\bar{\rho}'$ is specified we must have that $\mathcal{S}$ and $\mathcal{M}(\mathcal{S})$ have the same set of possible outputs in response to x after $\bar{\rho}'$. This contradicts $\mathcal{I}$ being a reduction of $\mathcal{M}(\mathcal{S})$ as required. □

We therefore know that to test whether the SUT is a GQR of $\mathcal{S}$ it is sufficient to determine whether an FSM $\mathcal{I}$ that models the SUT is a reduction of $\mathcal{M}(\mathcal{S})$. The following two results will allow us to reason about the effectiveness of test suites when testing from partial FSM $\mathcal{S}$ and completely-specified FSM $\mathcal{M}(\mathcal{S})$.

Proposition 8: Given partial FSM $\mathcal{S}$ that forms the specification and completely-specified FSM $\mathcal{I}$ that models an SUT, if $\mathcal{I}$ fails test case $t \in T(\mathcal{S})$ when testing from $\mathcal{S}$ then $\mathcal{I}$ fails test case t when testing from $\mathcal{M}(\mathcal{S})$.

Proof: Since $\mathcal{I}$ fails t when testing from $\mathcal{S}$ there is some trace $\bar{\rho} = \bar{\rho}'x/y$ in $L(\mathcal{I})$ such that $\bar{\rho}'x/y' \in L(t)$ for some $y' \in Y$ and $\bar{\rho} \notin L(\mathcal{S})$. Let $\bar{\rho}$ be a minimal such trace. By the definition of $T(\mathcal{S})$ the response to x is specified in all states of $\mathcal{S}$ reached by $\bar{\rho}$ and so by the definition of $\mathcal{M}(\mathcal{S})$ we have that the set of possible responses to x after $\bar{\rho}'$ are the same in $\mathcal{S}$ and $\mathcal{M}(\mathcal{S})$. But this implies that $\mathcal{I}$ fails t when testing from $\mathcal{M}(\mathcal{S})$. □

In the proof of Proposition 9 below we will use the notion of a prefix of a test case, which can be constructed by deleting subtrees. We use the observation that if $t \notin T(\mathcal{S})$ then the set of prefixes of t that are in $T(\mathcal{S})$ ($pref(t) \cap T(\mathcal{S})$) contains a unique maximal element.

Proposition 9: Given specification (partial) FSM $\mathcal{S}$ and completely-specified FSM $\mathcal{I}$ that models an SUT, if $\mathcal{I}$ fails test case $t \in T(\mathcal{M}(\mathcal{S}))$ when testing from $\mathcal{M}(\mathcal{S})$ then $\mathcal{I}$ fails the maximal prefix t' of t that is in $T(\mathcal{S})$ when testing from $\mathcal{S}$.

Proof: Since $\mathcal{I}$ fails test case t when testing from $\mathcal{M}(\mathcal{S})$ we have that $\mathcal{I}$ can produce a trace $\bar{\rho}$, in response to t, that is not a trace of $\mathcal{M}(\mathcal{S})$. Let $\bar{\rho}$ be some minimal such trace. Then $\bar{\rho} = \bar{\rho}'x/y$ for some $\bar{\rho}' \in L(\mathcal{S})$ and input x and, by the definition of $\mathcal{M}(\mathcal{S})$ and $\bar{\rho}$, we have that the response to x is specified in all states of $\mathcal{S}$ reached by $\bar{\rho}'$. Thus, as in the proof of Proposition 8, the possible responses to x after $\bar{\rho}'$ are the same in $\mathcal{S}$ and $\mathcal{M}(\mathcal{S})$. Further, the maximal prefix t' of t that is in $T(\mathcal{S})$ must have a state reached by $\bar{\rho}'$ from which input x is applied. This implies that $\mathcal{I}$ fails t' when testing from $\mathcal{S}$ and so the result follows. □

We can therefore convert a test suite $\mathcal{T}$ for reduction to $\mathcal{M}(\mathcal{S})$ into a test suite (called $R(\mathcal{T})$) for checking whether the SUT is a GQR of $\mathcal{S}$ (we use maximal prefixes that are in $T(\mathcal{S})$). Thus, we can use methods for testing from a completely-specified FSM when testing from a partial FSM.

Theorem 1: Given partial FSM $\mathcal{S}$, if $\mathcal{T}$ is m-complete when testing from $\mathcal{M}(\mathcal{S})$ then the test suite $R(\mathcal{T})$ is m-complete when testing from $\mathcal{S}$.

Proof: This follows from Proposition 9. □

In the next section we explore the efficiency of m-complete test suites generated from $\mathcal{M}(\mathcal{S})$.

6 Test Suite Efficiency

We now address the issue of efficiency, where a test suite $\mathcal{T}$ is efficient if $\mathcal{T}$ achieves the test objective (in this case, being m-complete) and $\mathcal{T}$ is minimal for some notion of minimality. We consider what happens to reduced m-complete test suites for $\mathcal{M}(\mathcal{S})$ when adapted for use with $\mathcal{S}$. We would like the process of converting a test suite for $\mathcal{M}(\mathcal{S})$, for use with $\mathcal{S}$, to not introduce any redundancy: if we generate a non-redundant test suite $\mathcal{T}$ for $\mathcal{M}(\mathcal{S})$ then the test suite $R(\mathcal{T})$ is guaranteed not to

be redundant for $\mathcal{M}(\mathcal{S})$. In this section we prove that the above property holds and then consider other notions of minimality.

In this section we first define what we mean by a test suite being reduced, with this essentially requiring that the test suite is m-complete but loses this property if we make it smaller (by replacing any test case by its proper prefixes).

Definition 12: Given partial FSM $\mathcal{S}$, m-complete test suite $\mathcal{T} \subseteq T(\mathcal{S})$ is *reduced* for $\mathcal{S}$ if for every test case $t \in \mathcal{T}$ we have that $(\mathcal{T} \cup pref(t)) \setminus \{t\}$ is not m-complete.

First we prove a result regarding reduced test suites.

Proposition 10: Given partial FSM $\mathcal{S}$, if m-complete test suite $\mathcal{T}$ is reduced for $\mathcal{M}(\mathcal{S})$ then $\mathcal{T} \subseteq T(\mathcal{S})$.

Proof: Proof by contradiction: assume that m-complete test suite $\mathcal{T}$ is reduced for $\mathcal{M}(\mathcal{S})$ and $\mathcal{T} \not\subseteq T(\mathcal{S})$. Let $t \in \mathcal{T}$ be such that $t \notin T(\mathcal{S})$ and let t' be the maximal prefix of t that is in $T(\mathcal{S})$. Since $\mathcal{T}$ is m-complete and reduced for $\mathcal{M}(\mathcal{S})$ there is some FSM $\mathcal{I}$ with at most m states such that $\mathcal{I}$ fails t but passes t'. But since t' is the maximal prefix of t that is in $T(\mathcal{S})$, all outputs are allowed by $\mathcal{M}(\mathcal{S})$ in response to the inputs in t that follow t'. Thus, since $\mathcal{I}$ passes t' it must pass t and this provides a contradiction as required. □

We obtain the following result, which shows that if we take a test suite $\mathcal{T}$ that is reduced for $\mathcal{M}(\mathcal{S})$ then the test suite $R(\mathcal{T})$ is reduced for $\mathcal{S}$.

Theorem 2: Given partial FSM $\mathcal{S}$, if $\mathcal{T}$ is m-complete and reduced for $\mathcal{M}(\mathcal{S})$ then the test suite $R(\mathcal{T})$ is m-complete and reduced for $\mathcal{S}$.

Proof: By Theorem 1, $R(\mathcal{T})$ is m-complete for $\mathcal{S}$. Since $\mathcal{T}$ is reduced for $\mathcal{M}(\mathcal{S})$, by Proposition 10 we know that $\mathcal{T} \subseteq T(\mathcal{S})$ and so we have that $R(\mathcal{T}) = \mathcal{T}$. Consider some test case $t \in \mathcal{T}$. Since $\mathcal{T}$ is reduced for $\mathcal{M}(\mathcal{S})$ there is some FSM $\mathcal{I}$ with at most m states that fails t when testing from $\mathcal{M}(\mathcal{S})$ but that passes all test cases in $(\mathcal{T} \cup pref(t)) \setminus \{t\}$ when testing from $\mathcal{M}(\mathcal{S})$. Since $t \in T(\mathcal{S})$, by Proposition 9, $\mathcal{I}$ fails t when testing from $\mathcal{S}$. Further, by Proposition 8, $\mathcal{I}$ passes all test cases in $(\mathcal{T} \cup pref(t)) \setminus \{t\}$ when testing from $\mathcal{S}$. The result thus follows. □

This shows that reduced test suites for $\mathcal{M}(\mathcal{S})$ are mapped to reduced test suites for $\mathcal{S}$; the mapping does not introduce redundancy. We might be interested in optimisations when generating an m-complete test suite from $\mathcal{M}(\mathcal{S})$ and so we consider what happens to a minimal test suite for $\mathcal{M}(\mathcal{S})$ for other notions of minimality. However, there are several alternative notions of minimal. For example, we want to minimise the number of test cases if the reset between test cases is expensive [28], [29], [30]. In other cases we are interested in the test suite size: the sums of the lengths of the test cases. We thus introduce the notion of a cost function.

Definition 13: Given test suite $\mathcal{T}$, we have the following three cost functions:

1) $f_1(\mathcal{T}) = \sum_{t \in \mathcal{T}} |t|$
2) $f_2(\mathcal{T}) = \sum_{t \in \mathcal{T}} \ell(t)$
3) $f_3(\mathcal{T}) = |\mathcal{T}|$

Here $|t|$ is the number of states in t, $\ell(t)$ is the length of the longest trace in $L(t)$, and $|\mathcal{T}|$ is the number of test cases in $\mathcal{T}$ (the number of resets used with $\mathcal{T}$).

Definition 14: Given partial FSM $\mathcal{S}$ and integer m, test suite $\mathcal{T}$ is *minimal* for $\mathcal{S}$ and cost function f_i if $\mathcal{T}$ is m-complete and for every test suite $\mathcal{T}'$ that is m-complete for $\mathcal{S}$ we have that $f_i(\mathcal{T}) \leq f_i(\mathcal{T}')$.

We now show that minimality is preserved.

Theorem 3: Given partial FSM $\mathcal{S}$ and cost function f_i, $1 \leq i \leq 3$, if $\mathcal{T}$ is minimal for $\mathcal{M}(\mathcal{S})$ and f_i then $R(\mathcal{T})$ is minimal for $\mathcal{S}$ and f_i.

Proof: First consider f_1. By definition, a minimal test suite for f_1 is reduced. Thus, since $\mathcal{T}$ is reduced, by Proposition 10 we have that $R(\mathcal{T}) = \mathcal{T}$ and by Theorem 2 we have that $R(\mathcal{T})$ is reduced for $\mathcal{S}$. Proof by contradiction: let us suppose that $\mathcal{T}$ is not minimal for $\mathcal{S}$. Then there is an m-complete test suite $\mathcal{T}'$ for $\mathcal{S}$ such that $f_1(\mathcal{T}') < f_1(\mathcal{T})$. However, since $\mathcal{T}'$ is m-complete for $\mathcal{S}$, by Proposition 8 we have that $\mathcal{T}'$ is also m-complete for $\mathcal{M}(\mathcal{S})$ and this contradicts the minimality of $\mathcal{T}$ for $\mathcal{M}(\mathcal{S})$ as required. The proof for f_2 follows in a similar way.

Now consider f_3. Since $\mathcal{T}$ is minimal for $\mathcal{M}(\mathcal{S})$ there is a reduced m-complete test suite $\mathcal{T}_2$ for $\mathcal{M}(\mathcal{S})$ with $|\mathcal{T}| = |\mathcal{T}_2|$ and this is also minimal. Since $f_3(\mathcal{T}) = f_3(\mathcal{T}_2)$ it is sufficient to prove that $\mathcal{T}_2$ is minimal for $\mathcal{S}$. By Proposition 10 we have that $R(\mathcal{T}_2) = \mathcal{T}_2$ and by Theorem 2 we have that $\mathcal{T}_2$ is reduced for $\mathcal{S}$. Proof by contradiction: let us suppose that $\mathcal{T}_2$ is not minimal for $\mathcal{S}$. Then there is an m-complete test suite $\mathcal{T}'$ for $\mathcal{S}$ such that $f_3(\mathcal{T}') < f_3(\mathcal{T}_2)$. However, since $\mathcal{T}'$ is m-complete for $\mathcal{S}$, by Proposition 8, $\mathcal{T}'$ is also m-complete for $\mathcal{M}(\mathcal{S})$ and this contradicts the minimality of $\mathcal{T}$ for $\mathcal{M}(\mathcal{S})$ as required. □

The above results show that reduced/minimal test suites for $\mathcal{M}(\mathcal{S})$ get mapped to reduced/minimal test suites for $\mathcal{S}$ (three notions of minimality). Thus, the mapping from test suites for $\mathcal{M}(\mathcal{S})$ to test suites for $\mathcal{S}$ does not introduce redundancy and there is value in finding a minimal test suite for $\mathcal{M}(\mathcal{S})$.

7 Conclusions

This paper considered the problem of generating an m-complete test suite from a partial FSM $\mathcal{S}$. We explored implementation relations for the case where a completely-specified FSM $\mathcal{I}$ models the SUT and a partial FSM $\mathcal{S}$ is the specification. We considered quasi-equivalence and quasi-reduction but found that the previous definitions for observable partial FSMs with harmonised traces could not be used in the more general setting. We therefore generalised these to generalised-quasi-equivalence (GQE) and generalised-quasi-reduction (GQR).

Having defined GQE and GQR, we concentrated on testing for GQR. We found that we can complete $\mathcal{S}$ to form a completely-specified FSM $\mathcal{M}(\mathcal{S})$ such that we can map an m-complete test suite for $\mathcal{M}(\mathcal{S})$ to an m-complete test suite for $\mathcal{S}$. Thus, we can use methods for testing from a completely-specified FSM to test from any partial FSM.

We then considered efficiency. We defined the notion of an m-complete test suite $\mathcal{T}$ being reduced: for all $t \in \mathcal{T}$, if we remove t from $\mathcal{T}$ or replace t by a proper prefix of t then the resultant test suite is not m-complete. We proved that reduced test suites for $\mathcal{M}(\mathcal{S})$ are mapped to reduced test suites for $\mathcal{S}$. Similarly, a minimal test suite for $\mathcal{M}(\mathcal{S})$ is mapped to a minimal test suite for $\mathcal{S}$.

One of the main contributions of this paper was to show that the problem of testing (for GQR) from a partial FSM can be expressed in terms of the better understood problem of testing from a completely-specified FSM. One problem for future work is to determine whether the approach can be extended to testing for GQE. The approach used in this paper was based on proving that $\mathcal{I}$ is a GQR of $\mathcal{S}$ if and only if $\mathcal{I}$ is a reduction of $\mathcal{M}(\mathcal{S})$. One might look to prove that $\mathcal{I}$ is GQE to $\mathcal{S}$ if and only if $\mathcal{I}$ is equivalent to $\mathcal{M}(\mathcal{S})$. However, such a result will not generally hold since an FSM $\mathcal{I}$ that is quasi-equivalent to $\mathcal{S}$ does not have to implement all of the transitions added to $\mathcal{S}$ to form $\mathcal{M}(\mathcal{S})$. For example, if x is not specified in a state of $\mathcal{S}$ reached by trace $\bar{\rho}$ then $\mathcal{M}(\mathcal{S})$ produces all possible outputs in response to x after $\bar{\rho}$ but an FSM $\mathcal{I}$ that is GQE to $\mathcal{S}$ might only produce some of these outputs in response to x after $\bar{\rho}$. Fortunately, in such circumstance the test generation process removes any test cases that check the output in response to x after $\bar{\rho}$. As a result, it may still be possible to utilise a test suite $\mathcal{T}$ that tests whether $\mathcal{I}$ is equivalent to $\mathcal{M}(\mathcal{S})$; we test the SUT with $R(\mathcal{T})$. Testing for GQE is a topic for future work.

The ideas in this paper have the potential to have an impact elsewhere. For example, if we wish to devise methods for transforming a partial FSM specification M into an observable partial FSM M' then we might wish to know that the same set of SUTs are correct implementations of M and M'; we require general implementation relations to reason about this. There is also the interesting question as to whether the results, regarding the completion $\mathcal{M}(\mathcal{S})$, have any implications regarding the problem of distinguishing states and whether, for example, we can map the problem of distinguishing a set of states of a partial FSM M to a corresponding problem of distinguishing a set of states of completely-specified FSM $\mathcal{M}(\mathcal{S})$. This might allow us to reuse techniques, complexity results and bounds.

REFERENCES

[1] A. L. Bonifácio and A. V. Moura, "Test suite completeness and partial models," in *12th International Conference on Software Engineering and Formal Methods (SEFM 2014)*, ser. LNCS, vol. 8702. Springer, 2014, pp. 96–110.

[2] ——, "Partial models and weak equivalence," in *11th International Colloquium on Theoretical Aspects of Computing (ICTAC 2014)*, ser. LNCS, vol. 8687. Springer, 2014, pp. 80–96.

[3] ——, "On the completeness of test suites," in *Symposium on Applied Computing (SAC 2014)*. ACM, 2014, pp. 1287–1292.

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

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

[6] R. M. Hierons and U. C. Türker, "Incomplete distinguishing sequences for finite state machines," *The Computer Journal*, vol. 58, no. 11, pp. 3089–3113, 2015.

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

[8] A. Petrenko, N. Yevtushenko, A. Lebedev, and A. Das, "Nondeterministic state machines in protocol conformance testing," in *Proceedings of Protocol Test Systems, VI (C-19)*. Pau, France: Elsevier Science (North-Holland), 28-30 September 1994, pp. 363–378.

[9] A. Petrenko, N. Yevtushenko, and G. v. Bochmann, "Testing deterministic implementations from nondeterministic FSM specifications," in *IFIP TC6 9th International Workshop on Testing of Communicating Systems*. Darmstadt, Germany: Chapman and Hall, 9–11 September 1996, pp. 125–141.

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

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

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

[13] G. v. Bochmann, A. Das, R. Dssouli, M. Dubuc, A. Ghedamsi, and G. Luo, "Fault models in testing," in *Protocol Test Systems IV*. Elsevier Science (North-Holland), 1992, pp. 17–30.

[14] R. M. Hierons, "Testing from a non-deterministic finite state machine using adaptive state counting," *IEEE Transactions on Computers*, vol. 53, no. 10, pp. 1330–1342, 2004.

[15] M. P. Vasilevskii, "Failure diagnosis of automata," *Cybernetics*, vol. 4, pp. 653–665, 1973.

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

[17] A. Petrenko and N. Yevtushenko, "Conformance tests as checking experiments for partial nondeterministic FSM," in *5th International Workshop on Formal Approaches to Software Testing (FATES 2005)*, ser. Lecture Notes in Computer Science, vol. 3997. Springer, 2006, pp. 118–133.

[18] A. Simão and A. Petrenko, "Generating checking sequences for partial reduced finite state machines," in *20th IFIP TC 6/WG 6.1 International Conference Testing of Software and Communicating Systems, 8th International Workshop on Formal Approaches to Testing of Software (TestCom/FATES 2008)*, ser. Lecture Notes in Computer Science, vol. 5047. Springer, 2008, pp. 153–168.

[19] H. AboElFotoh, O. Abou-Rabia, and H. Ural, "A test generation algorithm for protocols modeled as non-deterministic FSMs," *The Software Engineering Journal*, vol. 8, no. 4, pp. 184–188, 1993.

[20] S. Y. Boroday, "Distinguishing tests for non-deterministic finite state machines," in *IFIP TC6 11th International Workshop on Testing of Communicating Systems*. Tomsk, Russia: Kluwer Academic Press, August 31 – September 2 1998, pp. 101–107.

[21] S. Fujiwara and G. v. Bochmann, "Testing non-deterministic state machines with fault coverage," in *Proceedings of Protocol Test Systems, IV*, 1991, pp. 267–280.

[22] R. M. Hierons, "Adaptive testing of a deterministic implementation against a nondetermistic finite state machine," *The Computer Journal*, vol. 41, no. 5, pp. 349–355, 1998.

[23] I. Hwang, T. Kim, S. Hong, and J. Lee, "Test selection for a nondeterministic FSM," *Computer Communications*, vol. 24, no. 12, pp. 1213–1223, 2001.

[24] G. L. Luo, G. v. Bochmann, and A. Petrenko, "Test selection based on communicating nondeterministic finite-state machines using a generalized Wp-method," *IEEE Transactions on Software Engineering*, vol. 20, no. 2, pp. 149–161, 1994.

[25] A. Petrenko, A. Simão, and N. Yevtushenko, "Generating checking sequences for nondeterministic finite state machines," in *Fifth IEEE International Conference on Software Testing, Verification and Validation (ICST 2012)*, 2012, pp. 310–319.

[26] P. Tripathy and K. Naik, "Generation of adaptive test cases from non-deterministic finite state models," in *Proceedings of the 5th International Workshop on Protocol Test Systems*, Montreal, September 1992, pp. 309–320.
[27] F. Zhang and T.-Y. Cheung, "Optimal transfer trees and distinguishing trees for testing observable nondeterministic finite-state machines," *IEEE Transactions on Software Engineering*, vol. 29, no. 1, pp. 1–14, 2003.
[28] R. M. Hierons, "Minimizing the number of resets when testing from a finite state machine," *Information Processing Letters*, vol. 90, no. 6, pp. 287–292, 2004.
[29] R. M. Hierons and H. Ural, "Generating a checking sequence with a minimum number of reset transitions," *Automated Software Engineering*, vol. 17, no. 3, pp. 217–250, 2010.
[30] A. Simão and A. Petrenko, "Fault coverage-driven incremental test generation," *The Computer Journal*, vol. 53, no. 9, pp. 1508–1522, 2010.