# Complete FSM Testing Using Strong Separability

Robert M. Hierons[1][0000−0002−4771−1446] and
Mohammad Reza Mousavi[2][0000−0002−4869−6794]

[1] School of Computer Science
University of Sheffield, UK, r.hierons@sheffield.ac.uk
[2] Department of Informatics
King's College London, UK, mohammad.mousavi@kcl.ac.uk

**Abstract.** Apartness is a concept developed in constructive mathematics, which has resurfaced in the areas of model learning and model-based testing. We identify some fundamental shortcomings of apartness in quantitative models, such as in hybrid and stochastic systems. We propose a closely-related alternative, called strong separability and show that using it to replace apartness addresses the identified shortcomings. We adapt a well-known complete model-based testing method, the Harmonized State Identifiers (HSI) method, to adopt strong separability. We prove that the adapted HSI method is complete. As far as we are aware, this is the first work to show how complete test suites can be generated for quantitative models such as those found in the development of cyber-physical systems.

**Keywords:** Finite state machine testing · cyber-physical systems · apartness · strong separability

## 1 Introduction

### 1.1 Background

Testing is an important yet costly part of system quality assurance. There has been significant interest in systematic, automated test generation techniques. Model-based testing (MBT) is an important class of such techniques in which test generation is based on a formal model (or *specification*) that defines the set of behaviours of a correct *system under test (SUT)*. Typically, a behaviour is a sequence of inputs and outputs and testing is black-box: we cannot observe the state of the SUT. There are many MBT techniques, with corresponding tools and evidence of effectiveness when applied to industrial systems (see examples of applications [10, 15, 17, 22, 24, 28, 34]; and also survey papers for an overview [18, 20, 25, 30]).

To formally reason about MBT effectiveness, one assumes a common semantic domain between the specification and the SUT [16]. Most MBT work has concerned state-based semantic domains: labelled transition systems (LTSs) [28] or finite state machines (FSMs) [21]. However, the developer might use a different formalism, with a test tool mapping a model to an FSM or LTS and an adapter handling differences in abstraction between a model and the SUT.

The behaviour of an FSM is defined by transitions between states, with a transition having a corresponding input and output. FSM-based test generation techniques typically aim to find two types of faults: output faults (a transition produces the wrong output); and state-transfer faults (a transition takes the SUT to the wrong state). State-transfer faults can lead to the SUT having more states than the specification FSM.

There are many FSM-based test generation techniques that aim to find state-transfer faults in addition to output faults [8]; these techniques utilise input sequences that *separate* states[3] of the specification $M$ in order to *identify* (or *check*) states. Most FSM-based test techniques concern specification FSMs that are minimal, deterministic, and completely-specified (cf. Section 2) and we also consider such FSMs (we discuss further extensions in the Conclusions). Of particular interest are test generation techniques that produce $m$-complete test suites: test suites that are guaranteed to determine correctness as long as the SUT behaves like an (unknown) FSM that has at most $m$ states. Classical examples of such $m$-complete methods include the W-method [10, 34], Wp-method [15], and HSI method [22].

### 1.2 Problem Definition

This paper aims to expand the applicability of complete FSM-based test techniques to a range of modern systems for which test generation is particularly challenging. We are interested in generating $m$-complete tests for systems in which outputs are drawn from a continuous domain and observations are noisy. This is characteristic of cyber-physical systems [19] and the conformance problem for such systems has been extensively researched [6, 1, 12]. For example, in testing automotive [29] and healthcare [26] systems, it is necessary to cater for error margins and noise. For example, ($\tau$, $\epsilon$)-conformance [1] incorporates error margins in time and space. However, it has not previously been possible to develop ($m$-)complete testing theories for such notions of conformance. Our solution can further serve as a building block for active FSM learning of cyber-physical systems using, e.g., variants of the celebrated L$^*$ [2] algorithm, making it possible to model-check complex implementations through their learned FSM models. We will elaborate on this in the discussion of future work.

### 1.3 Contributions

To achieve this, we need to reconsider what we mean by correctness (conformance) in scenarios such as those sketched above. It no longer makes sense to require that, for each input sequence $\bar{x}$, the SUT and specification produce the *same* output sequence. Instead, we require that the output sequence produced by the SUT (i.e., observed in testing) is *similar* to that produced by the specification

[3] An input sequence $\bar{x}$ separates two states $s_1$ and $s_2$ of $M$ if the input of $\bar{x}$ leads to different output sequences when applied in $s_1$ and $s_2$.

[6, 1, 12], e.g., using a distance metric. This leads to a different conformance relation and so the proofs of completeness, for current FSM-based test techniques, do not apply. In this paper we also show that the W, Wp, and HSI methods need not be complete in this context.

We took as a starting point recent work by Vaandrager [31] that shows that it is possible to reason about the completeness of several FSM-based test techniques in terms of *apartness*. Apartness is a form of inequality used in constructive mathematics. Vaandrager [31] showed how the notion of apartness can be used in the generation of an $m$-complete test set. We found, however, that the notion of apartness is too strong for the scenario of interest in this paper (see Section 3).

In this paper, we weaken apartness in a way that allows us to reason about FSM-based testing when conformance is based on a metric $\mu$ and threshold $t$. We start by specifying a corresponding notion of conformance, which is reminiscent of the notions of robustness [14, 9], metric bisimulation [7, 6, 11] and conformance testing [1, 12, 4, 5]. We then define strong separability, which is strictly weaker than apartness and stronger than separability. We consider the case where the states of the specification FSM $M$ are pairwise strongly separable. We give a sufficient condition for a test suite to be $m$-complete, defined in terms of strong separability. We adapt the HSI-method to the scenarios we consider and show that this returns $m$-complete test suites. This completeness result extends to the W-method and the Wp-method. Note that these are exactly the FSM-based test generation techniques considered by Vaandrager [31].

To summarise, the paper makes the following main contributions.

- We show that there are scenarios (classes of system, with a corresponding definition of conformance) in which the notion of apartness is too strong.
- We define strong separability and demonstrate that this is applicable in the identified scenarios.
- We give a sufficient condition for a test suite to be $m$-complete and show that test suites produced by the well-known Harmonized State Identifiers (HSI) method (and similar ones such as the Wp and W-methods) satisfy this condition if the states of the specification FSM are pairwise strongly separable and, for state identification, we use sequences that strongly separate states.
- We illustrate all concepts through our running example and use this to show that the W, Wp and HSI methods are not complete if we do not adapt them (i.e., if we use separability rather than strong separability).

### 1.4 Running Example

To illustrate the concepts and their potential applicability, we use the following hybrid systems model. The same example can be extended to cover stochastic aspects, but for the sake of readability we use a minimal running example.

*Example 1.* Consider a thermostat that senses the temperatures and uses it to switch a heater on/off and control the room temperature. Figure 1 gives a diagrammatic representation of the behaviour. The system starts at time zero and

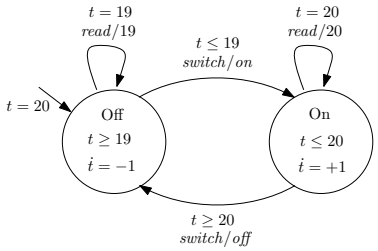


**Fig. 1.** Continuous Thermostat System

the room temperature at 20°C (in the discrete state Off) and the temperature decreases linearly with a first derivative -1. When the temperature reaches 19°C, the display can be updated through a "read" action (which outputs 19) and the heater is switched on through the "switch" action, which outputs "on". The behaviour in the state labelled "On" is similar, with the main difference being that the temperature increases linearly with the first derivative equal to 1.

Consider an implementation in which the initial condition and the guards differ by 0.5°C. We will compare such an implementation (as well as another faulty one) against the specification based on a notion of conformance in the remainder of the paper. The idea is to use an approximate notion of conformance that allows for an error margin and yet, produce test-cases that are complete, i.e., test cases that will fail on any non-conforming implementation with at most $m$ states. Consider, for example, a notion of conformance that has a threshold of 0.5°C for temperature differences (and no difference in the switching outputs) and thus, it accepts the implementation that differs only 0.5°C in temperature with the specification in Figure 1, while rejecting those with temperature differences that exceed the threshold or have incorrect switching behaviour. In this paper we show how complete test suites can be produced for this specification.

### 1.5 Structure

The paper is structured as follows. Section 2 provides preliminary definitions and Section 3 extends FSM concepts and definitions to the scenarios of interest. Within this we define separability and apartness and show that there are classes of system, including the Thermostat example, where separability is not an apartness relation. This leads to us defining strong separability. Section 4 gives a sufficient condition for a test suite to be $m$-complete, with a consequence being that the well-known W, Wp, and HSI methods return $m$-complete test suites for the notion of conformance and class of system considered if we base the choice of state identifiers on strong separability. We also show that separability is insufficient for these test generation techniques. Finally, in Section 5, we draw conclusions and describe possible lines of future work.

## 2 Preliminaries

In this section, we define what we mean by a Finite State Machine and provide classical definitions. In the next section, we adapt this to the context of interest where, for example, outputs may be drawn from a metric space.

**Definition 1.** *A* Finite State Machine (FSM) $M$ *is defined by a tuple* $(S, s_0, X, Y, \delta, \lambda)$ *in which:* $S$ *is the finite set of states;* $s_0 \in S$ *is the initial state;* $X$ *is the finite input alphabet;* $Y$ *is the output alphabet;* $\delta : S \times X \to S$ *is the transition function; and* $\lambda : S \times X \to Y$ *is the output function.*

The transition function and the output function are inductively lifted to sequences of inputs, by defining $\delta(s_i, \varepsilon) = s_i$, $\delta(s_i, x'.\bar{x}) = \delta(\delta(s_i, x'), \bar{x})$ and $\lambda(s_i, \varepsilon) = \varepsilon$, $\lambda(s_i, x'.\bar{x}) = \lambda(s_i, x').\lambda(\delta(s_i, x'), \bar{x})$, where $\varepsilon$ is the empty sequence, $x' \in X$, $\bar{x} \in X^*$. These FSMs are deterministic: for each state and input there is only one possible next state and output. In addition, they are completely-specified: $\delta$ and $\lambda$ are total functions. We restrict attention to deterministic and completely-specified FSMs, leaving other classes of FSM to future work.

In order to reason about test effectiveness, it is normal to assume that the SUT behaves like an unknown FSM [16]. We also assume that we have an upper bound $m$ on the number of states of the SUT.

**Assumption 1** *The specification is an FSM* $M = (S, s_0, X, Y, \delta, \lambda)$ *with* $n$ *states and the behaviour of the SUT can be represented by an FSM* $M_I = (Q, q_0, X, Y, \delta_I, \lambda_I)$ *with at most* $m$ *states.*

Note that we do not assume that the FSM $M_I$ is known. In addition, Definition 1 relaxes the standard definition of an FSM, which requires that the output alphabet $Y$ is finite. This may not seem to be an important point since the set of outputs that can be produced by $M$ is finite. However, allowing $Y$ to be infinite has an impact on the set of potential SUTs being tested, which becomes infinite, and hence, challenges the completeness of tests.

*Example 2.* Consider our thermostat in Example 1 (Figure 1); assume that we would like to focus on the input/output behaviour and abstract from the internal continuous dynamics of the states. This abstraction is key to all well-known MBT techniques based on FSMs and LTSs. It is essential to note that in all such techniques, the states are not observable in testing, and only input-output behaviours are used to distinguish different behaviour.

The FSM arising from the specification in Figure 1 is depicted in Figure 2. The FSM (as well as the three implementations depicted in the same figure), have the set of input symbols $\{r, s\}$, representing reading the temperature and switching the heater, respectively. The output set is $\mathbb{R} \cup \{on, off\}$, i.e., the (uncountable) set of real numbers for temperature and the status values *on* and *off*. In our example, the relaxation to allow for infinite outputs is essential here to represent the infinite set of possible faulty implementations with deviating outputs, while a finite set of inputs would be sufficient to represent any discretisation of the passing of time.

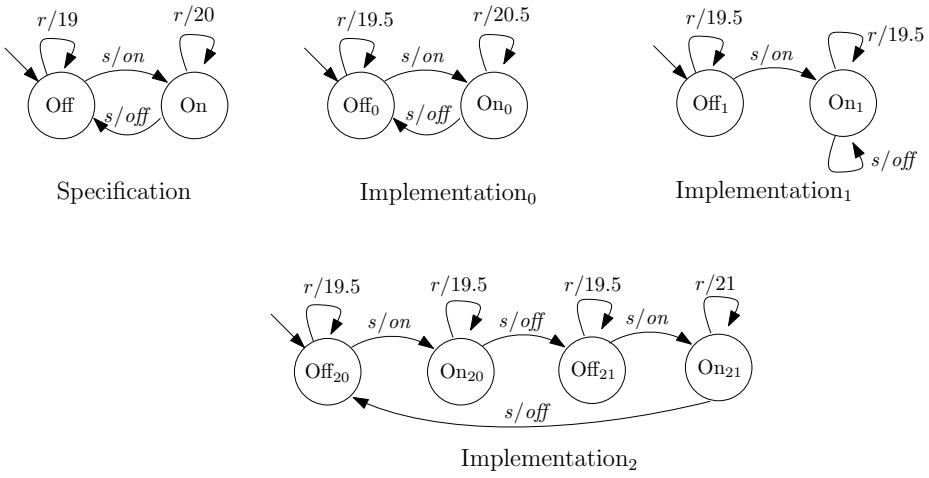


**Fig. 2.** Discretised Thermostat System: A Specification and Three Implementations

We use $\mathcal{F}$ to denote the set of FSMs with input alphabet $X$ and output alphabet $Y$. Given integer $m$, we use $\mathcal{F}^m$ to denote the set of FSMs in $\mathcal{F}$ with at most $m$ states. The following shows that if $Y$ is finite then so is $\mathcal{F}^m$; then it is not difficult to show that there must be a finite $m$-complete test suite.

**Proposition 1.** *Given FSM $M = (S, s_0, X, Y, \delta, \lambda)$ with finite output alphabet $Y$, and integer $m$, $\mathcal{F}^m$ is finite.*

There is a correspondence between FSMs and states: if $M$ is an FSM and $s_i$ is a state of $M$ then we can also see $s_i$ as being an FSM: the FSM formed by making $s_i$ the initial state of $M$. We can thus use $\mathcal{F}$ to denote the set of possible states of the unknown FSM $M_I$ that represents the SUT.

Traditional FSM-based testing typically involves applying an input sequence to the SUT and checking that the output sequence produced is that specified. We formalise this in terms of separating states and FSMs; we use the term d-separate in order to distinguish between the classical notion and that required for the scenarios of interest (defined in Section 3).

**Definition 2.** *Given FSM $M$ and states $s_1, s_2 \in S$, an input sequence $\bar{x}$* d-separates *$s_1$ and $s_2$ if and only if $\lambda(s_1, \bar{x}) \neq \lambda(s_2, \bar{x})$. If $\bar{x}$* d-separates *$s_1$ and $s_2$ then we can write $s_1 \not\equiv_{\bar{x}} s_2$ and say that $s_1$ and $s_2$ are d-separable. If $\bar{x}$ does not d-separate $s_1$ and $s_2$ then we write $s_1 \equiv_{\bar{x}} s_2$. Similarly, given a set $W$ of input sequences, we write $s_1 \not\equiv_W s_2$ if an input sequence in $W$ d-separates $s_1$ and $s_2$; otherwise $s_1 \equiv_W s_2$.*

It is straightforward to see that $\equiv_{\bar{x}}$ is an equivalence relation. In addition, it can be applied when comparing two FSMs $M$ and $M_I$ since one can define an FSM that is the disjoint union of $M$ and $M_I$. The following is the notion of conformance used in traditional FSM-based testing, which we call d-conformance.

**Definition 3.** *Given FSMs* $M = (S, s_0, X, Y, \delta, \lambda)$ *and* $M_I = (Q, q_0, X, Y, \delta_I, \lambda_I)$, $M_I$ d-conforms *to* $M$ *if and only if for all* $\bar{x} \in X^*$ *we have that* $M_I \equiv_{\bar{x}} M$.

## 3 Conformance and Strong Separation

In this section we explain how separability and conformance can be naturally translated to the scenarios of interest. We then define strong separability.

### 3.1 Separability and Conformance

In the scenarios of interest an output might, for example, be the speed of a vehicle as measured by a sensor, which is an estimate of the actual speed. An output in $Y$ could also be a probability distribution; in testing we apply a test case multiple times, observe a set of values drawn from an unknown distribution, and use statistical hypothesis testing techniques. In such situations, we need a way of comparing observations made with the expected output and we formalise this as a similarity relation $\sim$ on outputs. As previously explained, similarity will often be defined in terms of a metric $\mu$ and a threshold $t$; outputs $y$ and $y'$ are similar if $\mu(y, y') \leq t$ (see Example 3 below). Note that $\sim$ normally will not be transitive and so need not be an equivalence relation. The following lifts $\sim$ to output sequences.

**Definition 4.** *Given similarity relation* $\sim$ *on* $Y$, $y, y' \in Y$, *and* $\bar{y}, \bar{y}' \in Y^*$ *we have that:*

- $\varepsilon \sim \varepsilon$;
- $y\bar{y} \sim y'\bar{y}'$ *if and only if* $y \sim y'$ *and* $\bar{y} \sim \bar{y}'$;

We can define conformance in terms of $\sim$ as follows.

**Definition 5.** *Given FSMs* $M = (S, s_0, X, Y, \delta, \lambda)$ *and* $M_I = (Q, q_0, X, Y, \delta_I, \lambda_I)$, $M_I$ conforms *to* $M$ *if and only if for all* $\bar{x} \in X^*$ *we have that* $\lambda(s_0, \bar{x}) \sim \lambda_I(q_0, \bar{x})$.

*Example 3.* Consider the FSMs given in Figure 2. We instantiate the metric $\mu$ for real-valued outputs to be the absolute difference between them; the distance between *on* and *off* (as well as the distance between real numbers on one hand, and status values) is defined to be $\infty$. Define two outputs $o, o' \in \mathbb{R}$ to be $\epsilon$-conforming, denoted by $t \sim_\epsilon t'$, when $|o - o'| \leq \epsilon$. This leads to a natural notion of conformance, inspired by the hybrid conformance literature [1], on FSMs.

According to this notion of conformance, $\mathit{Specification} \sim_{0.5} \mathit{Implementation}_0$. This conformance relation holds, because for all input sequences, the difference between the respective outputs is bounded by 0.5.

Consider the specification *Specification* and implementation $\mathit{Implementation}_1$ in Figure 2. In fact, $\mathit{Implementation}_1$ does not $\epsilon$-conform to *Specification* for any value of $\epsilon$. This is witnessed by the output to input sequence $s.s.s$, where the difference between the last output in the two systems is $\infty$.

Consider again the specification *Specification* and implementation $Implementation_2$ in Figure 2. It does *not* hold that *Specification* $\sim_{0.5}$ $Implementation_2$ either. This is witnessed by the output to input sequence $s.s.s.r$ (the difference between the last output in the two systems is 1).

We now define the notion of separating two states for the scenario of interest.

**Definition 6.** *Given FSM $M$ and states $s_1, s_2 \in S$, an input sequence $\bar{x}$* separates *$s_1$ and $s_2$ if and only if $\lambda(s_1, \bar{x}) \not\sim \lambda(s_2, \bar{x})$. If $\bar{x}$* separates *$s_1$ and $s_2$ then we can write $s_1 \not\approx_{\bar{x}} s_2$. If $\bar{x}$ does not separate $s_1$ and $s_2$ then we write $s_1 \approx_{\bar{x}} s_2$. Similarly, given a set $W$ of input sequences, we write $s_1 \not\approx_W s_2$ if there is a input sequence in $W$ that separates $s_1$ and $s_2$ and otherwise we write $s_1 \approx_W s_2$.*

Where the input sequence of interest is clear, we often use $\not\approx$ and $\approx$, avoiding reference to the input sequences used. Conformance and separability are related.

**Proposition 2.** *Given FSMs $M = (S, s_0, X, Y, \delta, \lambda)$ and $M_I = (Q, q_0, X, Y, \delta_I, \lambda_I)$, $M_I$ conforms to $M$ if and only if no input sequence separates $s_0$ and $q_0$.*

When FSMs $M$ and $M_I$ do not conform to each other, we denote this by $M \not\approx M_I$; following Proposition 2, this can only happen if there exists $\bar{x} \in X^*$ that separates the initial states of $M$ and $M_I$, which we denote by $M \not\approx_{\bar{x}} M_I$.

*Example 4.* Consider again the FSMs given in Figure 2 and the notion of similarity $\sim_{0.5}$; we have that the states *Off* in *Specification* and $Off_0$ in $Implementation_0$ are not separable by any input sequence and the two FSMs conform to each other with respect to $\sim_{0.5}$.

However, *Specification* and $Implementation_2$ do not conform to each other. The input sequence $s.s.s.r$ separates the states *Off* and $Off_{20}$, because the former produces the output sequence $on.off.on.20$, while the later produces $on.off.on.21$ and the two output sequences have a distance of 1. Note that $s.s.r.s$ does not separate these states even though $s.s.r.s$ takes *Specification* and $Implementation_2$ to states that do not correspond to one another; testing is black-box and we *cannot* observe the internal state reached.

### 3.2 Apartness and Strong Separability

Previous work has used the notion of an apartness relation and shown how one can use this to reason about test effectiveness when there is a finite set of outputs and conformance is defined in terms of equality of outputs rather than similarity.

**Definition 7.** *A binary relation $\#$ on a set $Z$ is an* apartness relation *if and only if it satisfies the following properties*

- *It is irreflexive: for all $x \in Z$, $x\#x$ does not hold;*
- *It is symmetric: for all $x, y \in Z$, if $x\#y$ then $y\#x$; and*
- *It is co-transitive: for all $x, y, z \in Z$, if $x\#y$ then either $x\#z$ or $y\#z$.*

Consider the classical notion of separability for FSMs defined in Section 2 (d-separability). It is clear that d-separability is both irreflexive and symmetric. Now let us suppose that input sequence $\bar{x}$ d-separates states $s_1$ and $s_2$ of $M$ and so $\bar{x}$ leads to different output sequences $\bar{y}_1$ and $\bar{y}_2$ when applied in $s_1$ and $s_2$. Further, let us suppose that we input $\bar{x}$ in some other state $s_i$ and this leads to output sequence $\bar{y}$. Since $\bar{x}$ d-separates $s_1$ and $s_2$ we have that $\bar{y}_1 \neq \bar{y}_2$. We therefore must have that either $\bar{y} \neq \bar{y}_1$ or $\bar{y} \neq \bar{y}_2$ (or both) and so $\bar{x}$ must d-separate $s_i$ from at least one of $s_1$ and $s_2$. As a result, d-separability is co-transitive and so is an apartness relation when testing a deterministic SUT against a completely-specified deterministic FSM $M$, with identity rather than similarity. The results of Vaandrager [31], regarding a test suite being $m$-complete, can therefore be applied.

Now consider what we mean by separability when similarity is defined in terms of a metric $\mu$ and a threshold $t$. Then an input sequence separates states $s_1$ and $s_2$ if the resultant observations $o_1$ and $o_2$ are such that $\mu(o_1, o_2) > t$. In such scenarios, separability is not an apartness relation since it need not be co-transitive, as shown by the following. Note that since we do not know the FSM model $M_I$ of the SUT, in the definition of apartness (Definition 7), $Z$ will be the set of all states of all FSMs in $\mathcal{F}$ and not just the states of $M$.

*Example 5.* Consider the FSM thermostat models in Example 2 (Figure 2); there, consider the states *Off* and *On* in the *Specification* FSM. They are separable using both inputs $r$ and $s$. Let us consider what happens if we use $r$ to separate these states and we have some state $s_i$ of the SUT. Then $s_i$ can have $r/19.5$, e.g., the state $Off_0$ in $Implementation_0$, that cannot be separated using only input $r$ from either of the two states. This shows that separability is not co-transitive and so is not an apartness relation.

Since we cannot use apartness, we explore how this can be weakened for the scenarios of interest. We first discuss how d-separability is used in FSM-based testing. In most FSM-based test techniques that aim to find state-transfer faults, input sequences that d-separate states of a specification FSM $M$ are used to check the current state of the SUT. To see how this works, let us suppose that input sequence $\bar{x}$ d-separates states $s_1$ and $s_2$ of $M$ and that the (expected) output sequences produced are $\bar{y}_1$ and $\bar{y}_2$ respectively. Let us suppose that $\bar{x}$ is applied to the SUT when the SUT is in some state $s_i$. There are three possibilities:

1. The SUT produces $\bar{y}_2$ in response to $\bar{x}$: $\bar{x}$ d-separates $s_i$ and $s_1$.
2. The SUT produces $\bar{y}_1$ in response to $\bar{x}$: $\bar{x}$ d-separates $s_i$ and $s_2$.
3. The SUT produces another output sequence $\bar{y}_3$ in response to $\bar{x}$: $\bar{x}$ d-separates $s_i$ and from both $s_1$ and $s_2$.

FSM-based test techniques utilise such information provided by applying input sequences that d-separate states of the specification. However, in Example 5 we saw that the above does not hold in the scenarios considered in this paper: if two states of the specification are separated by an input sequence $\bar{x}$ then it is possible that a state of the SUT is not separated from either of them by $\bar{x}$.

We now strengthen the notion of separability, to what we call strong separability, in a manner that ensures that strong separability satisfies the property (of d-separability in traditional FSM-based testing scenarios) described above.

**Definition 8.** *Given FSM $M$ and states $s_1, s_2 \in S$, input sequence $\bar{x}$ is a* witness *that states $s_1$ and $s_2$, $s_1 \neq s_2$, are* strongly separable*, denoted $s_1 \#^w_{\bar{x}} s_2$, if and only if for all $s_3 \in \mathcal{F}$ we have that either $s_1 \not\approx_{\bar{x}} s_3$ or $s_2 \not\approx_{\bar{x}} s_3$. We also say that $s_1$ and $s_2$ are* strongly separable *and denote this $s_1 \#^w s_2$.*

If similarity is defined by a metric $\mu$ and threshold $t$ then strong separability essentially requires us to double the threshold. The following is the situation for input sequences of length one and follows immediately from metrics satisfying the triangle inequality; it is straightforward to generalise the result to input sequences of arbitrary length.

**Proposition 3.** *If similarity is defined in terms of a metric $\mu$ and threshold $t$ then an input $x$ strongly separates states $s_1$ and $s_2$ of $M$ if and only if $\mu(\lambda(s_1, x), \lambda(s_2, x)) > 2t$.*

*Example 6.* Consider the FSM thermostat models in Example 2 (Figure 2). Recall that states *Off* and *On* in the *Specification* FSM are separable using both inputs $r$ and $s$. However, we have seen that they are not strongly separable using input $r$. Namely, there can be a state that has $r/19.5$, e.g., the state $Off_0$ in $Implementation_0$, that cannot be separated using only input $r$ from either of the two states. In such cases, one can fix this issue, i.e., obtain strong separability, by either choosing different input sequences (in this case $s$) or decreasing the threshold of similarity, e.g., to 0.1, so that all possible states in $\mathcal{F}$ are separable (by $r$) from at least one of these two states. In practice, the use of a lower threshold might require changes to the equipment used to observe the SUT in testing or to additional test runs with the same input sequence (so that more precise estimates are produced).

The following states and proves basic properties of strong separability.

**Proposition 4.** *Given FSM $M$ and input sequence $\bar{x}$, the relation $\#^w_{\bar{x}}$ is symmetric and irreflexive.*

Using the following, we can compare strong separability and separability.

**Proposition 5.** *Given FSM $M$, input sequence $\bar{x}$ and states $s_1$ and $s_2$ of $M$ such that $s_1 \#^w_{\bar{x}} s_2$, we have that $\bar{x}$ separates $s_1$ and $s_2$.*

The following is immediate from the definition of strong separability and corresponds to the property (described above) of d-separability in traditional FSM-based testing.

**Proposition 6.** *Given states $s_1$ and $s_2$ of $M$ with $s_1 \#^w_{\bar{x}} s_2$, if $s_3 \in \mathcal{F}$ then at least one of the following must hold.*

1. *$\bar{x}$ separates $s_3$ from $s_1$;*
2. *$\bar{x}$ separates $s_3$ from $s_2$.*

## 4 Test Generation

In this section, we explore the problem of test generation when, for every pair of states of specification $M$, we have a specific input sequence that provides a witness that these states are strongly separable.

**Assumption 2** *For every pair $s_1, s_2$ of distinct states of $M$, there is an input sequence $\bar{x}$ that is a witness that $s_1$ and $s_2$ are strongly separable. We will use $w(s_1, s_2)$ to represent such a witness and require that $w(s_1, s_2) = w(s_2, s_1)$.*

Recall that $M_I$ conforms to $M$ if and only if for all $\bar{x} \in X^*$ we have that $\lambda(s_0, \bar{x}) \sim \lambda_I(q_0, \bar{x})$ (Definition 5). In testing, we will apply input sequences (*test sequences*) to the SUT and, for each such test sequence $\bar{x}$, check whether the response of the SUT to $\bar{x}$ is related to the specified response to $\bar{x}$ under $\sim$; namely, whether $\lambda_I(q_0, \bar{x}) \sim \lambda(s_0, \bar{x})$. If $\lambda(s_0, \bar{x}) \not\sim \lambda_I(q_0, \bar{x})$ then the SUT *fails* $\bar{x}$ and otherwise it *passes* $\bar{x}$. A test suite will thus be a set of input sequences. We are interested in generating test suites with guaranteed fault detection ability.

**Definition 9.** *Given an FSM $M$, a set $T$ of input sequences is* $m$-complete *if for all $M_I \in \mathcal{F}^m$ we have that $M_I$ conforms to $M$ if and only if $M_I$ does not fail any test sequence in $T$.*

We will retain the notion of a state cover used in classical FSM-based test generation techniques: a minimal prefix-closed set of input sequences that, between them, reach all states of $M$.

**Definition 10.** *Given a finite state machine $M$, a set $V$ of input sequences is a* state cover *for $M$ if $V$ is prefix closed and for all $s_i \in S$ there is exactly one input sequence $v_i \in V$ such that $\delta(s_0, v_i) = s_i$.*

*Example 7.* Consider the *Specification* FSM in Example 2 (Figure 2); a state cover set $V$ for *Specification* is $\{\varepsilon, s\}$.

In the following, for a state $s_i$ of $M$ we use $W(s_i)$ to denote the set of input sequences used to show that $s_i$ is strongly separable from other states of $M$.

**Definition 11.** *Given state $s_1$ of $M$, we let $W(s_1)$ denote the* state identification *set*

$$W(s_1) = \{w(s_1, s_2) | s_2 \in S \setminus \{s_1\}\}$$

Note that only a single witness is used for a pair of states and typically one will use a relatively short such witness. Later (Proposition 11) we provide a polynomial upper bound on the lengths of shortest such witnesses.

*Example 8.* Consider FSM *Specification* in Example 2 (Figure 2). Both $r$ and $s$ pairwise separate the states of *Specification* but we have seen that only $s$ strongly separates these states: $Off \#_s^w On$ and $\neg Off \#_r^w On$. Since $Off \#_s^w On$, we can choose $\{s\}$ to be the state identification set for *Off* (and also for *On*).

Recall that we do not know the FSM $M_I$ that models the SUT but we will want to reason about the states of $M_I$ that are met during testing. Given state $s_i$ of $M$ and input sequence $\bar{x}$, we use $B_{\bar{x}}(s_i)$ to denote the ball (set) of states of $M_I$ that are not separated from $s_i$ by $\bar{x}$.

**Definition 12.** *Given state $s_i$ of $M$, input sequence $\bar{x}$, and FSM $M_I$ with state set $Q$, $B_{\bar{x}}(s_i) = \{q \in Q | s_i \approx_{\bar{x}} q\}$.*

*Example 9.* Consider the *Specification* and *Implementation*$_0$ FSMs in Example 2 (Figure 2), comprising the set of states $\{Off, On, Off_0, On_0\}$. We have that $Q = \{On_0, Off_0\}$. Given identification set $\{s\}$, we have that $B_s(Off) = \{Off_0\}$ and $B_s(On) = \{On_0\}$. Clearly we have that $B_s(On)$ and $B_s(Off)$ are disjoint.

The following is a consequence of strong separability and tells us that if we take two states $s_1$ and $s_2$ of $M$ that are strongly separated by $\bar{x}$ then $B_{\bar{x}}(s_1)$ and $B_{\bar{x}}(s_2)$ are pairwise disjoint.

**Proposition 7.** *Given states $s_1$ and $s_2$ of $M$ with $s_1 \#^w_{\bar{x}} s_2$, if $\bar{x}$ strongly separates $s_1$ and $s_2$, $s'_1 \in B_{\bar{x}}(s_1)$ and $s'_2 \in B_{\bar{x}}(s_2)$ then $s'_1 \neq s'_2$.*

Observe that $B_{\bar{x}}(s_i)$ is defined in terms of separability and *not* strong separability. However, Proposition 7 concerns input sequences that strongly separate states. The following shows what can happen if we were to instead use input sequences that separate, but do not strongly separate, the states of the specification.

*Example 10.* Consider again the *Specification* and *Implementation*$_0$ FSMs in Example 2 (Figure 2), comprising the set of states $\{Off, On, Off_0, On_0\}$, with $Q = \{On_0, Off_0\}$. Now consider the input $r$, which separates the states of the specification, and the sets $B_1 = \{q \in Q | q \sim_r Off\}$ and $B_2 = \{q \in Q | q \sim_r On\}$. Here $B_1 = \{Off_0\}$ and $B_2 = \{On_0, Off_0\}$, so $B_1$ and $B_2$ are not disjoint.

The test generation approach will be based on placing a lower bound on the number of states $M_I$ must have if the SUT is faulty and it passes all the test cases in a given test suite. The approach will use Proposition 7, which tells us that if input sequences $\bar{x}_1$ and $\bar{x}_2$ reach states $s_1$ and $s_2$ of $M$ respectively, $w$ strongly separates $s_1$ and $s_2$ (ie $s_1 \#^w_w s_2$), and the SUT conforms to the specification on both $\bar{x}_1.w$ and $\bar{x}_2.w$ then $\bar{x}_1$ and $\bar{x}_2$ must reach *different* states of $M_I$.

We now consider what we know about the states of the SUT reached by the state cover $V$ if $M_I$ conforms to $M$ on the set of input sequences formed by following each sequence in $V$ by the corresponding state identification set.

**Proposition 8.** *Let $V$ be a state cover for FSM $M$ whose states are pairwise strongly separable. Further, let us suppose that for every $v \in V$ we have that the states $s_i = \delta(s_0, v)$ and $q_i = \delta_I(q_0, v)$ satisfy $s_i \approx_{W(s_i)} q_i$. Then $V$ reaches $n$ separate states of $M_I$.*

*Example 11.* Following up on Examples 7, 8, and 9, consider the state cover set $\{\varepsilon, s\}$ and the state identification set $\{s\}$ for which the states of the specification are strongly separable. Now consider $Implementation_0$; the state cover set of the specification will take $Implementation_0$ to $Off_0$ and $On_0$, which are clearly different states. The same holds for $Implementation_1$, which has the same number of states.

For $Implementation_2$, since it has more states, we need more inputs to reach the remaining states: using $s$, *Specification* and $Implementation_2$ will arrive in state $(On, On_{20})$, further inputs are needed to bring the pair of *Specification* and $Implementation_2$ into $(On, On_{21})$ and then identify the output fault on input $r$. This is formalised in the following test generation algorithm.

The test generation techniques that we build upon use input sequences of the form $v.\bar{x}.w$ such that $v$ is a sequence in the state cover $V$, $\bar{x}$ is any input sequence of a given length $\ell$ (determined by the maximum number of extra states in the SUT), and $w$ is a state identifier for the state $\delta(s_0, v.\bar{x})$ of $M$. There are two ways in which such an input sequence can lead to a failure being observed.

1. The SUT $M_I$ does not conform to the specification $M$ on the initial sequence $v.\bar{x}$. If the SUT has output faults (one or more transitions produce the wrong output) then these output faults will lead to such failures.
2. There are $v$, $\bar{x}$, and $w$ such that the SUT $M_I$ conforms $M$ on the initial sequence $v.\bar{x}$ but does not conform to the specification on $v.\bar{x}.w$. Here, a state identifier for $\delta(s_0, v.\bar{x})$ separates the state of the SUT $M_I$ reached by $v.\bar{x}$ from the state of the specification $M$ reached by $v.\bar{x}$ (the test finds a state transfer fault).

The following introduces corresponding notation, for a fixed length $\ell$ of input sequence $\bar{x}$.

**Definition 13.** *Let us suppose that all states of FSM $M$ are strongly separable and $M_I$ is an FSM with the same input and output alphabets as $M$. Further, let us suppose that $V$ is a state-cover for $M$ and for all $s_i \in S$ we have that $W(s_i)$ is a state identification set for $s_i$. Given integer $\ell \geq 0$:*

$$\begin{aligned} D(M, M_I, V, \ell) &= \{(v, \bar{x}) | v \in V \wedge |\bar{x}| = \ell \wedge M \not\approx_{v\bar{x}} M_I\} \\ DW(M, M_I, V, W, \ell) &= \{(v, \bar{x}) | v \in V \wedge |\bar{x}| = \ell \wedge \exists w \in W(\delta(s_0, v\bar{x})). \\ &\qquad M \not\approx_{v\bar{x}w} M_I\} \end{aligned}$$

We now provide two results that show what one can deduce from the value of the *smallest* integer $\ell$ such that $D(M, M_I, V, \ell) \cup DW(M, M_I, V, W, \ell) \neq \emptyset$. Given this $\ell$, we will place a *lower* bound on the number of different states of $M_I$ reached by $V$ and sequences of the form $v.\bar{x}$, for $v \in V$ and input sequence $\bar{x}$ of length at most $\ell$ (Proposition 9). We then show that this implies an *upper* bound on the value of $\ell$ that we need to use in testing (Proposition 10).

**Lemma 1.** *Let us suppose that all states of FSM $M$ are strongly separable, $M_I$ does not conform to $M$, and $\ell > 0$ is the smallest value such that $D(M, M_I, V, \ell) \cup$*

*$DW(M, M_I, V, W, \ell) \neq \emptyset$. If $(v, \bar{x}) \in D(M, M_I, V, \ell) \cup DW(M, M_I, V, W, \ell)$ and $\bar{x}_1$ is a non-empty proper prefix of $\bar{x}$ then the state of $M_I$ reached by $v\bar{x}_1$ is not a state of $M_I$ reached by an input sequence in $V$.*

**Lemma 2.** *Let us suppose that all pairs of states of FSM $M$ are strongly separable, $M_I$ does not conform to $M$, $\ell > 0$, and $\ell$ is the smallest value such that $D(M, M_I, V, \ell) \cup DW(M, M_I, V, W, \ell) \neq \emptyset$. If $(v, \bar{x}) \in D(M, M_I, V, \ell) \cup DW(M, M_I, V, W, \ell)$ and $\bar{x}_1, \bar{x}_2$ are different non-empty proper prefixes of $\bar{x}$ then $v\bar{x}_1$ and $v\bar{x}_2$ reach different states of $M_I$.*

*Example 12.* For non-conforming implementation *Implementation*$_1$, we have that it is sufficient to take $\bar{x}$ to be $s$ since after $s \in V$ and $\bar{x} = s$, a further input $s \in W$ can reach a non-conformance verdict between the specification and the implementation. This is because the faulty implementation does not have any additional state with respect to the specification. In terms of Definition 13 this is summarised as follows, where, and for brevity we refer to the specification as $M$ and to *Implementation*$_1$ as $M_I$:

$$\begin{aligned} D(M, M_I, V, 1) &= \emptyset \\ DW(M, M_I, V, W, 1) &= \{(s, s)\} \end{aligned}$$

$D(M, M_I, V, 1)$ is the empty set because the pair with the shortest sequence to distinguish $M$ and $M_I$ is $(s, s.s)$, of which the second component is beyond the limit of $l = 1$. However, $DW(M, M_I, V, W, 1)$ contains the pair $(s, s)$, because after $s.s$ by performing a further $s \in W$, we can distinguish the two systems.

For non-conforming implementation *Implementation*$_2$, the situation is different because it contains more states than the specification. Namely, we need three further inputs, in addition to the state cover, to identify the output fault (on input $r$) at state $On_{21}$: after performing input $s$ from the state cover, we need the additional inputs $s.s.r$ to first bring the implementation to $On_{21}$ (using $s.s$) and then identify the faulty output by performing an additional input $r$. This is generalised in the following results, where we identify an upper bound on the length of intermediate sequences, between the state cover and the state identification.

In terms of Definition 13 this is summarised as follows, where, and for brevity we refer to the specification as $M$ and to *Implementation*$_1$ as $M_I$:

$$\begin{aligned} D(M, M_I, V, 1) &= D(M, M_I, V, 2) = \emptyset \\ D(M, M_I, V, 3) &= \{(s, s.s.r)\} \end{aligned}$$

$D(M, M_I, V, 1)$ and $D(M, M_I, V, 2)$ are both the empty set because the pair with the shortest sequence to distinguish $M$ and $M_I$ is $(s, s.s.r)$, of which the second component is beyond the limit of $l = 2$.

However $DW(M, M_I, V, W, 3)$ is the empty set; had we had a transition fault, instead of an output fault at the final state, then $DW(M, M_I, V, W, 3)$ would have been non-empty.

We can use the above-given Lemmas to place a lower bound on the number of states an SUT must have if the SUT is faulty and we know the smallest value $\ell$ such that $D(M, M_I, V, \ell) \cup DW(M, M_I, V, W, \ell) \neq \emptyset$.

**Proposition 9.** *Let us suppose that all pairs of states of FSM $M$ are strongly separable, $M_I$ does not conform to $M$, and for all $v \in V$ and $w \in W(\delta(s_0, v))$ we have that $M \approx_{vw} M_I$. If $\ell$ is the smallest value such that $D(M, M_I, V, \ell) \cup DW(M, M_I, V, W, \ell) \neq \emptyset$ and $\ell > 0$ then $M_I$ has at least $n + \ell - 1$ states.*

Now consider the standard FSM-based testing context in which we assume that the FSM $M_I$ that models the SUT has at most $n + k$ states and we wish to generate sufficient tests to determine whether $M_I$ conforms to $M$. We can use Proposition 9 to provide an upper bound on the value of $\ell$ that we require to use when testing with sequences of the form $v.\bar{x}$ and $v.\bar{x}.w$, with $|\bar{x}| \leq \ell$.

**Proposition 10.** *Let us suppose that all pairs of states of FSM $M$ are strongly separable and $M_I$ does not conform to $M$. If $M_I$ has at most $k$ more states than $M$ then there exists some $v \in V$, and $\bar{x} \in X^*$ such that $|\bar{x}| \leq k + 1$ and one of the following holds:*

- *$M_I \not\approx_{v\bar{x}} M$; or*
- *there is some $w \in W(\delta(s_0, v\bar{x}))$ such that $M_I \not\approx_{v\bar{x}w} M$.*

We now show what this implies regarding test completeness.

**Theorem 1.** *Let us suppose that the states of FSM $M$ are pairwise strongly separable, $M$ has $n$ states, and $M_I$ has at most $m = n+k$ states. If $D(M, M_I, V, \ell) \cup D(M, M_I, V, \ell) = \emptyset$ for all $\ell \leq k + 1$ then $M_I$ conforms to $M$.*

This result shows how test generation can proceed: one simply uses the test sequences implicit in the definitions of $D(M, M_I, V, \ell)$ and $DW(M, M_I, V, W, \ell)$. Algorithm 1 gives the corresponding test generation algorithm. Note that for a state $s_i$ of $M$, $W(s_i)$ is a Harmonized State Identifier for $s_i$ as used in the HSI-method except that we require $W(s_i)$ to strongly separate $s_i$ from other states of $M$; it is not sufficient for $W(s_i)$ to separate $s_i$ from other states of $M$. As a result, Algorithm 1 is essentially the HSI-method, with the only difference being how we define conformance and the $W(s_i)$. Since the test set produced by the HSI-method is a subset of those produced by the W-method and the Wp-method, the above demonstrates that these test generation techniques also produce $m$-complete test sets when we have a notion of similarity of outputs rather than equality and we use state identifiers that strongly separate states.

*Example 13.* Consider the *Specification* FSM in Figure 2. We established in Example 7 that $V = \{\varepsilon, s\}$ and in Example 8 that the state-identification set that strongly separates the specification states is $W = \{s\}$. Hence, assuming that we are only interested in implementations that have the same state count as the specification, by applying Algorithm 1, we obtain:

$$\begin{aligned} T &= \{v.\bar{x}.w | v \in \{\varepsilon, s\} \wedge \bar{x} \in \{\varepsilon, r, s\} \wedge w \in \{s\}\} \\ T &= \{\varepsilon.\varepsilon.s, \varepsilon.r.s, \varepsilon.s.s, s.r.s, s.s.s\} \\ &= \{s, r.s, s.s, s.r.s, s.s.s\} \end{aligned}$$

**Algorithm 1** Test Generation Using Strong Separability

Input: FSM $M = (S, s_0, X, Y, \delta, \lambda)$ where $S = \{s_1, \ldots, s_n\}$, $m \geq n$
Derive state cover $V$
Derive state identification sets $W(s_1), \ldots, W(s_n)$ based on strong separability
Derive $T = \{v.\bar{x}.w | v \in V, \bar{x} \in X^*, 0 \leq |\bar{x}| \leq m - n + 1, w \in W(\delta(s_0, v.\bar{x}))\}$
Remove from $T$ all prefixes
**return** $T$

If we wish to find faulty implementations such as $Implementation_2$ then we need to use a larger value for $k$. If we use $k = 2$ then we obtain the following.

$$T = \{v.\bar{x}.w | v \in \{\varepsilon, s\} \wedge \bar{x} \in \{\varepsilon, r, s, rr, rs, sr, ss, rrr, rrs, rsr, rss, srr, srs, ssr, sss\} \wedge w \in \{s\}\}$$

Note that the key difference between these sets of generated inputs and the traditional HSI method is that the adapted algorithm forces the state-identification set to provide strong separability. Otherwise, the traditional HSI method can use $r$ to separate states and generate $r.r$, $s.s.r$ and $s.r.r$ as the longest sequences (for $k = 0$) and these sequences will miss the fault in $Implementation_1$. This demonstrates that the HSI method, with state identifiers that do not strongly separate the states of the specification, is not complete. Using this specification FSM, state cover, and state identifiers, the W-method and Wp-method return the same test suite and so are also incomplete.

A final observation is that, as usual, if a test suite contains input sequences $\bar{x}_1$ and $\bar{x}_2$ and $\bar{x}_2$ is a prefix of $\bar{x}_1$ then we can remove $\bar{x}_2$ from the test suite without reducing effectiveness. In the first example above, this final optimisation reduces the test suite to $\{r.s, s.r.s, s.s.s\}$.

It is now worth considering how large the test suite can be for an FSM specification $M$ with $n$ states. Clearly $V$ contains $n$ input sequences and these have length at most $n - 1$: we can generate such input sequences through a breadth-first search. Further, each $W(s_i)$ contains at most $n - 1$ sequences. The following places an upper bound on the lengths of sequences in $W(s_i)$, assuming we use shortest sequences that suffice.

**Proposition 11.** *Let us suppose that $M$ is an FSM with $n$ states, which are pairwise strongly separable. For every pair $s_1, s_2$ of distinct states of $M$, there is an input sequence of length at most $\frac{n(n-1)}{2}$ that strongly separates $s_1$ and $s_2$.*

Thus, the sizes of the state cover and state identification sets are low-order polynomial. Naturally, the size of the test suite grows exponentially as the upper bound, $k$, on the number of extra states grows. However, this is also the case with techniques such as the HSI-method when applied in the classical FSM context. In practice, the value of $k$ used might depend upon domain knowledge and a cost/benefit analysis.

It is finally worth commenting on the computational complexity of deriving the state cover and state identifiers. Clearly, a state cover can be derived in low-order polynomial time since breadth-first search takes linear time. It is also possible to derive the state identifiers in polynomial time. In particular, it is possible to define a simple iterative algorithm based on the proof of Proposition 11, with this operating as follows. First, we determine which pairs of states can be strongly separated by a single input and record these (we could say that these are strongly 1-separable). We then determine which of the remaining pairs $\{s_i, s_j\}$ of states are strongly separated by input sequences of length 2 (they are strongly 2-separable). In order to check this, for each pair $\{s_i, s_j\}$ of states (that are not strongly 1-separable) we check whether there is an input $x$ such that states $\delta(s_i, x)$ and $\delta(s_j, x)$ are strongly 1-separable. This process continues: in each iteration, for each pair $\{s_i, s_j\}$ of states not already strongly separated, we determine whether there is an input $x$ such that states $\delta(s_i, x)$ and $\delta(s_j, x)$ have already been strongly separated in an earlier iteration. Clearly, each iteration takes polynomial time and, from Proposition 11, we know that there are at most $\frac{n(n-1)}{2}$ iterations. Bringing together the above information, we have that Algorithm 1 has polynomial time complexity as long as we bound $k$. Note, however, that the test suite can size grow exponentially with $k$ but this is true also of classical FSM test generation techniques such as the W, Wp, and HSI-methods.

Finally, we identify a condition under which separability and strong separability coincide.

**Definition 14 (Strongly different).** *Two outputs $y_1, y_2 \in Y$ are* strongly different *if for all $y_3 \in Y$ we have that either $y_1 \not\sim y_3$ or $y_2 \not\sim y_3$. Further, an FSM $M = (S, s_0, X, Y, \delta, \lambda)$ has* strongly different outputs *if for all $s_1, s_2 \in S$ and $x \in X$ we have that either $\lambda(s_1, x) = \lambda(s_2, x)$ or $\lambda(s_1, x)$ and $\lambda(s_2, x)$ are strongly different.*

**Proposition 12.** *Given an FSM $M = (S, s_0, X, Y, \delta, \lambda)$, if $M$ has strongly different outputs then an input sequence $\bar{x}$ strongly separates states $s_1$ and $s_2$ of $M$ if and only if $\bar{x}$ separates $s_1$ and $s_2$.*

Recall that Theorem 1 tells us that we can use the HSI-method (and so the W and Wp-methods) to produce $m$-complete test suites as long as the state identification sequences strongly separate the states of the specification $M$. Proposition 12 thus tells us that if $M$ has strongly different outputs then we can use the W-method, Wp-method and HSI-method in the usual way (and so any tools that implement these). In some scenarios, we might be able to design the testing approach (eg the number of times that test execution is repeated) in order to reduce the threshold so that the specification has strongly different outputs.

## 5 Conclusions

This paper proposed the notion of strong separability, inspired by apartness in constructive mathematics. This allows for separating states according to approximate notions of conformance, that allow for a margin of error, in quantitative

finite-state machine models of systems. We showed the applicability of our notion by adopting it in a well-known model-based testing technique, the HSI method. We proved that using our notion, the HSI method is complete for approximate notions of conformance. To illustrate our approach, we used a simple thermostat example and three implementations. We also demonstrated that complete test suites need not be produced if we use separability rather than strong separability. We gave (low-order) complexity results for test generation and polynomial upper bound on test size.

Recall that the proposed approach operates by reasoning about the number of states of the SUT met during testing and does this by strongly separating states. We have seen that this is consistent with the W-method, the Wp-method and the HSI-method. These three approaches differ in the choice of state identification sequences used but they keep this choice fixed throughout test generation. Dorofeeva et al. [13] introduced the H-method, which allows different state identification sequences to be used for different input sequences. It may be possible to extend the results given in this paper to the H-method by allowing the input sequence $w(s_1, s_2)$ used to strongly separate two states of $M$ to vary.

There are several lines of possible future work corresponding to the following limitations. First, we abandoned the use of the observation tree used by Vaandrager [31] since a correct SUT might produce traces that are not traces of the specification $M$; it may be possible to reason about observation trees that are 'similar' to those of $M$. Second, we considered deterministic, completely-specified FSMs and the test generation algorithm assumes that the states of $M$ are pairwise strongly separable. Although these requirements are weaker than those imposed by the work that uses apartness, there is the question of how they might be further weakened. Finally, we will look into using strong separability to develop novel automata learning algorithms by integrating algorithms that use apartness [32, 33] with those that consider quantitative extensions of state machines [23, 3, 27]. Our threshold would allow for adjusting the level of abstraction we admit in learning and can provide an adjustable trade-off between the accuracy and the size of the learned model.

**Acknowledgements** Robert M. Hierons and Mohammad Reza Mousavi have been partially supported by the UKRI Trustworthy Autonomous Systems Node in Verifiability, Grant Award Reference EP/V026801/2. Mohammad Reza Mousavi has been partially supported by the EPSRC project on Verified Simulation for Large Quantum Systems (VSL-Q), grant reference EP/Y005244/1 and the EPSRC project on Robust and Reliable Quantum Computing (RoaRQ), Investigation 009 Model-based monitoring and calibration of quantum computations (ModeMCQ), grant reference EP/W032635/1 and ITEA/InnovateUK projects GENIUS and GreenCode.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References


1. Abbas, H., Mittelmann, H.D., Fainekos, G.: Formal property verification in a conformance testing framework. In: Twelfth ACM/IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE 2014, Lausanne, Switzerland, October 19-21, 2014. pp. 155–164. IEEE (2014). https://doi.org/10.1109/MEMCOD.2014.6961854, https://doi.org/10.1109/MEMCOD.2014.6961854
2. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987). https://doi.org/10.1016/0890-5401(87)90052-6, https://doi.org/10.1016/0890-5401(87)90052-6
3. Bacci, G., Ingólfsdóttir, A., Larsen, K.G., Reynouard, R.: Active learning of markov decision processes using baum-welch algorithm. In: Wani, M.A., Sethi, I.K., Shi, W., Qu, G., Raicu, D.S., Jin, R. (eds.) 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021, Pasadena, CA, USA, December 13-16, 2021. pp. 1203–1208. IEEE (2021). https://doi.org/10.1109/ICMLA52953.2021.00195, https://doi.org/10.1109/ICMLA52953.2021.00195
4. Biewer, S., D'Argenio, P.R., Hermanns, H.: Doping tests for cyber-physical systems. ACM Trans. Model. Comput. Simul. **31**(3), 16:1–16:27 (2021). https://doi.org/10.1145/3449354, https://doi.org/10.1145/3449354
5. Biewer, S., Dimitrova, R., Fries, M., Gazda, M., Heinze, T., Hermanns, H., Mousavi, M.R.: Conformance relations and hyperproperties for doping detection in time and space. Log. Methods Comput. Sci. **18**(1) (2022). https://doi.org/10.46298/LMCS-18(1:14)2022, https://doi.org/10.46298/lmcs-18(1:14)2022
6. van Breugel, F., Hermida, C., Makkai, M., Worrell, J.: An accessible approach to behavioural pseudometrics. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3580, pp. 1018–1030. Springer (2005). https://doi.org/10.1007/11523468_82, https://doi.org/10.1007/11523468_82
7. van Breugel, F., Worrell, J.: Towards quantitative verification of probabilistic transition systems. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2076, pp. 421–432. Springer (2001). https://doi.org/10.1007/3-540-48224-5_35, https://doi.org/10.1007/3-540-48224-5_35
8. Broy, M., Jonsson, B., Katoen, J.P., Leucker, M., Pretschner, A.: Model-Based Testing of Reactive Systems, Lecture Notes in Computer Science, vol. 3472. Springer (2005)
9. Chaudhuri, S., Gulwani, S., Lublinerman, R.: Continuity and robustness of programs. Commun. ACM **55**(8), 107–115 (Aug 2012). https://doi.org/10.1145/2240236.2240262, https://doi.org/10.1145/2240236.2240262
10. Chow, T.S.: Testing software design modelled by finite state machines. IEEE Transactions on Software Engineering **4**, 178–187 (1978)
11. Desharnais, J., Edalat, A., Panangaden, P.: Bisimulation for labelled markov processes. Information and Computation **179**(2), 163–193 (2002). https://doi.org/https://doi.org/10.1006/inco.2001.2962, https://www.sciencedirect.com/science/article/pii/S0890540101929621

12. Deshmukh, J.V., Majumdar, R., Prabhu, V.S.: Quantifying conformance using the skorokhod metric. Formal Methods Syst. Des. **50**(2-3), 168–206 (2017). https://doi.org/10.1007/S10703-016-0261-8, https://doi.org/10.1007/s10703-016-0261-8
13. Dorofeeva, R., El-Fakih, K., Yevtushenko, N.: An improved conformance testing method. In: 25th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2005). Lecture Notes in Computer Science, vol. 3731, pp. 204–218. Springer (2005)
14. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009). https://doi.org/10.1016/J.TCS.2009.06.021, https://doi.org/10.1016/j.tcs.2009.06.021
15. Fujiwara, S., v. Bochmann, G., Khendek, F., Amalou, M., Ghedamsi, A.: Test selection based on finite state models. IEEE Transactions on Software Engineering **17**(6), 591–603 (1991)
16. Gaudel, M.C.: Testing can be formal too. In: 6th International Joint Conference CAAP/FASE Theory and Practice of Software Development (TAPSOFT'95). Lecture Notes in Computer Science, vol. 915, pp. 82–96. Springer (1995)
17. Hennie, F.C.: Fault-detecting experiments for sequential circuits. In: Proceedings of Fifth Annual Symposium on Switching Circuit Theory and Logical Design. pp. 95–110. Princeton, New Jersey (November 1964)
18. Hierons, R.M., Bogdanov, K., Bowen, J.P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Kapoor, K., Krause, P., Lüttgen, G., Simons, A.J.H., Vilkomir, S.A., Woodward, M.R., Zedan, H.: Using formal specifications to support testing. ACM Computing Surveys **41**(2), 9:1–9:76 (2009)
19. Khakpour, N., Mousavi, M.R.: Notions of conformance testing for cyber-physical systems: Overview and roadmap (invited paper). In: Aceto, L., de Frutos-Escrig, D. (eds.) 26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015. LIPIcs, vol. 42, pp. 18–40. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015). https://doi.org/10.4230/LIPICS.CONCUR.2015.18, https://doi.org/10.4230/LIPIcs.CONCUR.2015.18
20. Lee, D., Yannakakis, M.: Testing finite-state machines: State identification and verification. IEEE Transactions on Computers **43**(3), 306–320 (1994)
21. Lee, D., Yannakakis, M.: Principles and methods of testing finite-state machines - a survey. Proceedings of the IEEE **84**(8), 1089–1123 (1996)
22. Luo, G., Petrenko, A., v. Bochmann, G.: Selecting test sequences for partially-specified nondeterministic finite state machines. In: The 7th IFIP Workshop on Protocol Test Systems. pp. 95–110. Chapman and Hall, Tokyo, Japan (November 8–10 1994)
23. Medhat, R., Ramesh, S., Bonakdarpour, B., Fischmeister, S.: A framework for mining hybrid automata from input/output traces. In: Girault, A., Guan, N. (eds.) 2015 International Conference on Embedded Software, EMSOFT 2015, Amsterdam, Netherlands, October 4-9, 2015. pp. 177–186. IEEE (2015). https://doi.org/10.1109/EMSOFT.2015.7318273, https://doi.org/10.1109/EMSOFT.2015.7318273
24. Miller, T., Strooper, P.A.: A case study in model-based testing of specifications and implementations. Software Testing, Verification and Reliability **22**(1), 33–63 (2012)

25. Mohd-Shafie, M.L., Kadir, W.M.N.W., Lichter, H., Khatibsyarbini, M., Isa, M.A.: Model-based test case generation and prioritization: a systematic literature review. Software and Systems Modeling **21**(2), 717–753 (2022). https://doi.org/10.1007/s10270-021-00924-8, https://doi.org/10.1007/s10270-021-00924-8
26. Sankaranarayanan, S., Kumar, S.A., Cameron, F., Bequette, B.W., Fainekos, G., Maahs, D.M.: Model-based falsification of an artificial pancreas control system. SIGBED Rev. **14**(2), 24–33 (2017). https://doi.org/10.1145/3076125.3076128, https://doi.org/10.1145/3076125.3076128
27. Tappler, M., Aichernig, B.K., Bacci, G., Eichlseder, M., Larsen, K.G.: $L^*$-based learning of markov decision processes (extended version). Formal Aspects Comput. **33**(4-5), 575–615 (2021). https://doi.org/10.1007/S00165-021-00536-5, https://doi.org/10.1007/s00165-021-00536-5
28. Tretmans, J.: Model based testing with labelled transition systems. In: Formal Methods and Testing. Lecture Notes in Computer Science, vol. 4949, pp. 1–38. Springer (2008)
29. Tuncali, C.E., Pavlic, T.P., Fainekos, G.: Utilizing s-taliro as an automatic test generation framework for autonomous vehicles. In: 19th IEEE International Conference on Intelligent Transportation Systems, ITSC 2016, Rio de Janeiro, Brazil, November 1-4, 2016. pp. 1470–1475. IEEE (2016). https://doi.org/10.1109/ITSC.2016.7795751, https://doi.org/10.1109/ITSC.2016.7795751
30. Utting, M., Pretschner, A., Legeard, B.: A taxonomy of model-based testing approaches. Software Testing, Verification and Reliability **22**(5), 297–312 (2012)
31. Vaandrager, F.W.: A new perspective on conformance testing based on apartness. In: Capretta, V., Krebbers, R., Wiedijk, F. (eds.) Logics and Type Systems in Theory and Practice - Essays Dedicated to Herman Geuvers on The Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 14560, pp. 225–240. Springer (2024). https://doi.org/10.1007/978-3-031-61716-4_15, https://doi.org/10.1007/978-3-031-61716-4_15
32. Vaandrager, F.W., Garhewal, B., Rot, J., Wißmann, T.: A new approach for active automata learning based on apartness. In: Fisman, D., Rosu, G. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13243, pp. 223–243. Springer (2022). https://doi.org/10.1007/978-3-030-99524-9_12, https://doi.org/10.1007/978-3-030-99524-9_12
33. Vaandrager, F.W., Sanders, M.: $L^\#$ for dfas. In: Jansen, N., Junges, S., Kaminski, B.L., Matheja, C., Noll, T., Quatmann, T., Stoelinga, M., Volk, M. (eds.) Principles of Verification: Cycling the Probabilistic Landscape - Essays Dedicated to Joost-Pieter Katoen on the Occasion of His 60th Birthday, Part III. Lecture Notes in Computer Science, vol. 15262, pp. 155–172. Springer (2024). https://doi.org/10.1007/978-3-031-75778-5_8, https://doi.org/10.1007/978-3-031-75778-5_8
34. Vasilevskii, M.P.: Failure diagnosis of automata. Cybernetics **4**, 653–665 (1973)