Article:
Ibias, A., Nunez, M. and Hierons, R.M. orcid.org/0000-0002-4771-1446 (2021) Using mutual information to test from finite state machines: test suite selection. Information and Software Technology, 132. 106498. ISSN 0950-5849

https://doi.org/10.1016/j.infsof.2020.106498

Using mutual information to test from Finite State Machines: test suite selection[☆]

Alfredo Ibias[a], Manuel Núñez[a], Robert M. Hierons[b]

[a] *Instituto de Tecnología del Conocimiento, Universidad Complutense de Madrid, Madrid, Spain*
[b] *Department of Computer Science, The University of Sheffield, Sheffield, United Kingdom*

Abstract

Context: Mutual Information is an information theoretic measure designed to quantify the amount of similarity between two random variables ranging over two sets. In this paper, we adapt this concept and show how it can be used to select a *good* test suite to test from a Finite State Machine (FSM) based on a *maximise diversity* approach.
Objective: The main goal of this paper is to use Mutual Information in order to select test suites to test from FSMs and evaluate whether we obtain better results, concerning the quality of the selected test suite, than current state-of-the-art measures.
Method: First, we defined our scenario. We considered the case where we receive two (or more) test suites and we have to choose between them. We were interested in this scenario because it is a recurrent case in regression testing. Second, we defined our notion based on Mutual Information: Biased Mutual Information. Finally, we carried out experiments in order to evaluate the measure.
Results: We obtained experimental evidence that demonstrates the potential value of the measure. We also showed that the time needed to compute the measure is negligible when compare to the time needed to apply extra testing. We compared our measure with a state-of-the-art test selection measure and showed that our proposal outperforms it. Finally, we have compared our measure with a notion of transition coverage. Our experiments showed that our measure is slightly worse than transition coverage, as expected, but its computation is 10 times faster.
Conclusion: Our experiments showed that Biased Mutual Information is a good measure for selecting test suites, outperforming the current state-of-the-art measure, and having a (negative) correlation to fault coverage. Therefore, we can conclude that our new measure can be used to select the test suite that is likely to find more faults. As a result, it has the potential to be used to automate test generation.

Keywords: Formal approaches to testing, Information Theory, Mutual information, Finite State Machines.

1. Introduction

Software testing [1, 2] is the main technique to validate complex systems with the goal of increasing their reliability. Testing is a time consuming part of software development; it has been observed that testing can cost more than 50% of the development budget [2]. Therefore, it is important to devise methodologies that reduce the time taken without notably decreasing effectiveness. A good starting point is to reduce the number of tests (the size of the *test suite*) that we apply to the System Under Test (SUT)[1]. Therefore, we require approaches that *select* a test suite that is small enough to be used in practice and is likely to be effective in finding faults. We can rephrase this as the problem of maximising the expected test effectiveness for a given cost/time. Since the cost of test execution typically depends on the test suite size, we reduce this to the problem of choosing amongst different test suites, that have the same size, the one that is most likely to find faults.

The main aim of the work presented in this paper is to devise measures that can help testers, or testing tools, to choose between alternative test suites. We focus on the case where we want to choose between different finite test suites that have the same number of inputs (and so, most likely, the same execution cost). The problem studied is relevant in a number of contexts. For example, we might build a test suite in an incremental manner. When choosing a next test case to add we will be comparing test suites

[☆] This work has been supported by the Spanish MINECO/FEDER (grant FAME, RTI2018-093608-B-C31); the Region of Madrid (grant FORTE-CM, S2018/TCS-4314) co-funded by EIE Funds of the European Union; the Region of Madrid - Complutense University of Madrid (grant PR65/19-22452); and the UK EPSRC (grant InfoTestSS, EP/P006116/2).

Email addresses: aibias@ucm.es (Alfredo Ibias), mn@sip.ucm.es (Manuel Núñez), r.hierons@sheffield.ac.uk (Robert M. Hierons)

URL: https://alfredoibias.com/ (Alfredo Ibias), http://antares.sip.ucm.es/manolo/ (Manuel Núñez), https://robhierons.github.io/ (Robert M. Hierons)

[1] The number of tests needed to exhaustively test even the simplest systems is exorbitant. For example, exhaustive testing of a black-box implementation of a method adding two numbers on a 32-bit machine needs around $8 \cdot 10^{28}$ tests.

of the same size (the current test suite extended by the different test cases that could be added) and will choose the test suite that we expect to be most effective. Thus, incremental approaches to building a test suite involve the comparison of test suites of the same size. Measures that compare test suites might also be used to help guide test suite generation since, for example, the measures could form fitness functions to be used within a search-based approach. The problem is also relevant in the context of regression testing, where we typically have to repeatedly use a test suite. Ideally, one executes the entire test suite in regression testing but often this is too expensive, with this having led to a significant body of work regarding the problem of selecting a 'best' subset (see, for example, [3, 4, 5]).

This paper considers the situation in which the SUT is a black-box; we know its input and output alphabets but have no additional information. As a result, in choosing a test suite we cannot use information about the internal structure of the SUT. A side-effect of this is that we will not have access to information about the coverage of the SUT achieved by a test suite. However, we assume that we do have a specification of the system that we want to build. In order to simplify the presentation, we assume that the specification is given by a Finite State Machine (FSM) but the approach can be adapted to deal with other state-based formalisms, in particular, those containing data.

The choice of FSMs as a formalisation was motivated by the fact that they have been used in a number of areas. The early work largely concerned protocol conformance testing [6, 7] and hardware (processor) testing, since processor designs are FSMs (see, for example, [8]). FSM-based techniques have also been used in testing web-services and web-based applications [9, 10]. It is important to observe that it is not necessary for the user to produce an FSM specification; the specification may be in some other state-based formalism, such as state-charts, with a model being mapped to an FSM that represents its semantics (possibly after some abstraction). This makes FSM-based approaches applicable to a wide range of state-based specifications, such as those used in the embedded systems industry. The value of such an approach was also demonstrated when used to test a number of Microsoft Windows protocols [11]. Recent work has shown that FSM-based test generation techniques can be applied when testing from a class of rather more expressive models (reactive I/O-state-transition systems, RIOSTS) [12]. The benefit is that RIOSTS can also be used with a range of embedded systems [12], with the approach having been evaluated on part of the European Train Control System and also an airbag controller [13]. Finally, it is worth mentioning that there are several tools that can be used to specify and analyse FSMs (for example, fsmlib-cpp[2], automatalib [14] and OpenFST [15]).

[2] https://github.com/agbs-uni-bremen/fsmlib-cpp

This paper describes a novel *information theoretic measure* and proposes its use to inform the choice between test suites. Specifically, we define a notion, *Biased Mutual Information (BMI)*, inspired by Mutual Information [16], and use it to compare test suites of similar length. The idea behind the definition of BMI is that two tests that share a large amount of information will be more likely to explore the same paths of an FSM, applying the same inputs and expecting the same outputs. The intended goal of BMI is thus to *indirectly* maximise diversity and is motivated by it having been widely recognised that diversity has a strong impact on test quality [17, 18, 19, 20]. Our hypothesis was that a test suite with lower BMI will tend to be more effective (be more likely to find faults) because lower BMI implies that the tests in the test suite share less information and, therefore, they explore more behaviours of the FSM. In order to evaluate this hypothesis, we used mutants, that simulate faults. Specifically, given an FSM and a pair of test suites, we checked whether the test suite with lower BMI killed more mutants. Our experiments revealed that minimising BMI led to test suites that kill more mutants most of the time. Moreover, we obtained a (negative) correlation between BMI and mutation score. We also found that the time needed to compute BMI is negligible when compared to the time typically needed to apply a single test suite. The consequence of this fact is that it will often be worth *spending* some time to decide between two test suites, instead of applying both of them, if, as usual, resources and time are scarce. These results suggest that BMI can be used to guide testing towards test suites that are likely to be more effective.

Interestingly, in additional experiments it was found that BMI outperformed a previous information theoretic approach, the test set diameter (TSDm) measure [21], when selecting between two randomly generated test suites. We also compared our measure with a notion of transition coverage. Specifically, we explored the question of which measure was most effective when used to choose between two test suites: choosing the test suite with smaller BMI or the test suite with higher transition coverage. Since transition coverage has more information available (the states from which inputs are applied), it was not surprising that transition coverage was slightly more effective. However, BMI was found to be considerably faster. As a result, in the context of testing from an FSM, there is a trade-off between effectiveness and speed. Note that, although the comparison with transition coverage was interesting, the aim of the work described in this paper was to produce a general method that can be used in a range of scenarios. This includes scenarios in which, for example, we do not have a complete specification but we can estimate the frequency of input/output pairs, which is the only information required to compute BMI. There is potential to use (random) sampling to estimate such frequencies.

Although it will become clearer when we give the formal definition of BMI, we will briefly explain why we need a *bias* in our notion. Essentially, in order to compute

our measure we need to know the frequency of occurrence of each input/output pair in the underlying FSM. Since the formulation of Mutual Information applies a logarithm over that frequency, if we have only one occurrence of an input/output pair (i, o) in the FSM, then we obtain $\log(1) = 0$. This produces two undesirable consequences. First, we obtain the smallest possible value for (i, o) while we want to give it the highest weight. Second, when we combine this weight with other values by multiplying them, we obtain 0, which leads to final values that are uninformative. The introduction of a *bias* solves this problem.

The rest of the paper is structured as follows. In Section 2 we review basic concepts and notation used in the paper. In Section 3 we present related work. In Section 4 we formally define our measure and explore some of its properties. Section 5 reports on the experiments and Section 6 discusses threats to validity. In Section 7 we discuss some decisions that we took during the research presented in this paper. Finally, in Section 8 we provide conclusions and discuss future work.

2. Preliminaries

In this paper, systems will be modelled as *Finite State Machines* (FSMs). In order to define an FSM, we first introduce some notation. Given set A, A^* denotes the set of finite sequences of elements of A; A^+ denotes the set of non-empty finite sequences of elements of A; and $\epsilon \in A^*$ denotes the empty sequence. We let $|A|$ denote the size of set A. Given a sequence $\sigma \in A^*$, $|\sigma|$ denotes its length. Given a sequence $\sigma \in A^*$ and $a \in A$, we have that σa denotes the sequence σ followed by a and $a\sigma$ denotes the sequence σ preceded by a.

Throughout this paper we let $\mathcal{I}$ be the set of input actions and $\mathcal{O}$ be the set of output actions. It is important to differentiate between input actions and *inputs* of the system. An input of a system will be a non-empty sequence of input actions, that is, an element of $\mathcal{I}^+$ (similarly for outputs and output actions).

An FSM is a (finite) labelled transition system in which every transitions has a label in the form of an *input/output pair* (a pair containing an input action and an output action). We use this formalism to define specifications.

Definition 1. *A* Finite State Machine *(FSM) is represented by a tuple $M = (Q, q_{in}, \mathcal{I}, \mathcal{O}, T)$ in which Q is a finite set of states, $q_{in} \in Q$ is the initial state, $\mathcal{I}$ is a finite set of input actions, $\mathcal{O}$ is a finite set of output actions, and $T \subseteq Q \times (\mathcal{I} \times \mathcal{O}) \times Q$ is the transition relation. The meaning of a* transition *$(q, (i, o), q') \in T$, also denoted by $(q, i/o, q')$, is that if M receives input action i when in state q then it can move to state q' and produce output action o. We write $(i, o) \in_m M$ to denote that the pair (i, o) appears in m transitions of M.*

We say that M is deterministic *if for all $q \in Q$ and $i \in \mathcal{I}$ there exists at most one pair $(q', o) \in Q \times \mathcal{O}$ such that $(q, i/o, q') \in T$.*

We assume that FSMs are deterministic; this makes the work compatible with the previously devised information theoretic (white-box) TSDm measure [21]. In particular, it allows us to run experiments that compare the proposed approach with TSDm.

An FSM can be represented by a diagram in which nodes represent states of the FSM and transitions are represented by arcs between the nodes. We use an incoming edge with no source to denote the initial state. In our case, all states are final as long as they are reachable from the initial state.

We will assume the *minimal test hypothesis* [22]: the SUT can be modelled as an (unknown) object described in the same formalism as the specification (here, an FSM). Note that we do not need to have access to this description; we are in a black-box testing framework and only assume the existence of such an FSM. In principle, we could weaken this hypothesis to simply assume that each time the SUT receives a sequence of input actions, it returns a sequence of output actions.

Our main goal while testing is to decide whether the behaviour of an SUT conforms to the specification of the system that we would like to build. In order to detect differences between specifications and SUTs, we need to compare their behaviours and the main notion to define such behaviours is given by the concept of a *trace*.

Definition 2. *Let $M = (Q, q_{in}, \mathcal{I}, \mathcal{O}, T)$ be an FSM, $\sigma = (i_1, o_1) \dots (i_k, o_k) \in (\mathcal{I} \times \mathcal{O})^*$ be a sequence of pairs and $q \in Q$ be a state. We say that M can perform σ from q if there exist states $q_1 \dots q_k \in Q$ such that for all $1 \leq j \leq k$ we have $(q_{j-1}, i_j/o_j, q_j) \in T$, where $q_0 = q$. If $q = q_{in}$ then we say that σ is a* trace *of M. We denote by $\texttt{traces}(M)$ the set of traces of M. Note that $\epsilon \in \texttt{traces}(M)$ for every FSM M.*

Next we define the notion of test. As previously explained, a test is a sequence of (input action, output action) pairs. A test suite will be a set of tests.

Definition 3. *Let $M = (Q, q_{in}, \mathcal{I}, \mathcal{O}, T)$ be an FSM. We say that $t = (i_1, o_1) \dots (i_k, o_k) \in (\mathcal{I} \times \mathcal{O})^+$ is a* test *for M if $t \in \texttt{traces}(M)$. The* length *of t is the length of the sequence, that is, $|t| = k$. In addition, the sequence of input actions of t is $\lambda = i_1 \dots i_k$ and the sequence of output actions of t is $\mu = o_1 \dots o_k$. We will sometimes use the notation $t = (\lambda, \mu) \in (\mathcal{I}^+ \times \mathcal{O}^+)$. We write $(i, o) \in t$ to denote that the pair (i, o) appears in the test t; $(i, o) \in_n t$ denotes that the pair (i, o) appears n times in the test t.*

A test suite *for M is a set of tests for M. Given a test suite $\mathcal{T} = \{t_1, \dots, t_n\}$, the* length *of the test suite is the sum of the lengths of its tests, that is, $|\mathcal{T}| = \sum_{i=1,\dots,n} |t_i|$.*

Let $t = (\lambda, \mu)$ be a test for M. We say that the application of t to an FSM M' fails if there exists μ' such that $(\lambda, \mu') \in \texttt{traces}(M')$ and $\mu \neq \mu'$. Similarly, let $\mathcal{T}$ be a

test suite for M. We say that the application of $\mathcal{T}$ to an FSM *M' fails if there exists $t \in \mathcal{T}$ such that the application of t to M' fails.*

Intuitively, a test (λ, μ) for M denotes that the application of the sequence of input actions λ to a correct system (with respect to M) should lead to the sequence of output actions μ. Note that if we allowed non-determinism, then the previous inequality must be appropriately replaced to express that a behaviour of the SUT must be one of those of the specification, and we will have a notion of conformance similar to ioco [23].

The concept of Test Set Diameter (TSDm) [21], which will be used as the state-of-the-art measure to compare with, is derived from Kolmogorov complexity [24]. The *Kolmogorov complexity* of a string is the length of the shortest program that produces that string. It has been shown that Kolmogorov complexity can be approximated using Normalised Compression Distance [25].

Definition 4. *Let x and y be two strings and $C(x)$ be the length of the string x after being compressed by a chosen compression program. We denote by $\texttt{ncd}(x, y)$ the Normalised Compression Distance of x and y and we define it as*

$$\frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

where xy denotes the concatenation of x and y.

The previous distance can be naturally extended to deal with multisets of strings.

Definition 5. *Let X be a multi-set of strings with at least two elements and $C(x)$ be the length of the string $x \in X$ after being compressed by a chosen compression program. We denote by $\texttt{NCD}(X)$ the Normalised Compression Distance of X and we define it as*

$$\begin{cases} \texttt{ncd}(x_1, x_2) & \text{if } X = \{\!|x_1, x_2|\!\} \\ \max\{\texttt{NCD}_1(X), \max_{Y \subset X}\{\texttt{NCD}(Y)\}\} & \text{otherwise} \end{cases}$$

where

$$\texttt{NCD}_1(X) = \frac{C(X) - \min_{x \in X}\{C(x)\}}{\max_{x \in X}\{C(X \backslash \{x\})\}}$$

and where $C(X)$ is the length of the compression of the concatenation of the strings belonging to X in any specific order as long as we use it for all the concatenations.

The Test Set Diameter of a test suite is defined in terms of the NCD of a (multi-)set of tests. We can consider different *multisets* for computing this metric. For example, we could use the multiset of test inputs (Input-TSDm), the multiset of test outputs (Output-TSDm) or even the multiset of execution traces (Trace-TSDm). In this paper, as recommended in the original work [21], we used ITSDm to drive test selection.

It is important to note that Test Set Diameter is the current baseline for the use of Information Theory to direct the generation and selection of test suites in black-box testing. An extensive study [26] found that Test Set Diameter outperformed many alternatives. Specifically, the study shows that, for 5 well-known programs (Grep, Sed, Flex, Make and GZip), ITSDm is the best alternative, obtaining fault detection rates between 85% and 95%. Moreover, they observed that ITSDm is able to compete with white-box techniques, obtaining differences in fault detection rates of less than 2%, which is remarkable given the fact that ITSDm, as a black-box technique, works with less information than white-box techniques.

Finally, we recall the classical definition of mutual information [16], which we use as an inspiration for our measure.

Definition 6. *Let A and B be two sets and ξ_A and ξ_B be two discrete random variables ranging, respectively, over A and B. We denote by $I(\xi_A; \xi_B)$ the mutual information of ξ_A and ξ_B and we define it as*

$$\sum_{b \in B} \sum_{a \in A} \sigma_{\xi_{A,B}}(a, b) \cdot \log_2 \frac{\sigma_{\xi_{A,B}}(a, b)}{\sigma_{\xi_A}(a) \cdot \sigma_{\xi_B}(b)}$$

where $\xi_{A,B}$ is the joint probability distribution, defined as usual, of ξ_A and ξ_B.

3. Related Work

There are many techniques for generating a test suite from an FSM specification. Possibly the first work in this area was the seminal paper by Moore, published in 1956, that described an overall framework [27]. Later, in 1964, Hennie [28] published a test generation algorithm. Hennie's algorithm required the specification to be a deterministic, completely-specified FSM that has a special type of sequence, called a distinguishing sequence[3]. Later, a more general test generation algorithm was published [29, 30], with this only requiring that the specification is a deterministic, completely-specified FSM[4].

The area of FSM-based test generation has been extended in a number of directions. The techniques mentioned above are *complete* in the sense that they return test suites that determine correctness as long as the SUT is equivalent to an (unknown) FSM in some well-defined fault domain; the fault domain typically places an upper bound on the number of states of the minimal FSM that represents the behaviour of the SUT. A number of other complete test generation algorithms that have been devised, typically with a focus on producing relatively small complete

[3] A distinguishing sequence is an input sequence that produces different output sequences from the different states of the FSM.

[4] It also requires the specification to be minimal but any deterministic, completely-specified FSM can be converted into an equivalent minimal deterministic, completely-specified FSM.

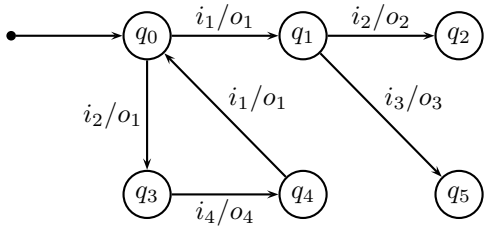

Figure 1: Example of FSM.

test suites (see, for example, [31, 32, 33]). There are also complete techniques for testing from other classes of FSM such as (possibly non-deterministic) partial FSMs (see, for example, [34, 35, 36]), FSMs representing distributed systems (see, for example, [37, 38]), probabilistic FSMs (see, for example, [39, 40]), and stochastic FSMs (see, for example, [41]). It is not always feasible to produce and use complete test suites and so there are also approaches that aim to cover the FSM specification in some way (see, for example, [42]). The interest in FSM-based testing is motivated by the ability of FSMs to suitably model many classes of system, possibly after an abstraction has been applied. Sometimes, however, it does not make sense to abstract out data (for example, where this data determines which transitions can be executed). There has thus been interest in testing from extended FSMs (EFSMs), which allow data (see, for example, [43, 44, 45, 46, 47]), including work on testing from Stream X-machines [48].

Interestingly, although there has been much work on testing from both FSMs and EFSMs, it appears that notions of diversity have not been utilised in these areas. A potential advantage of diversity-based approaches is that they can be used with any test budget: given a bound on the overall test execution cost/time available, one can aim to find the most diverse test suite whose cost does not exceed this bound. Diversity-based approaches are thus potentially extremely flexible and it should be possible to apply them with FSMs, EFSMs or FSMs extended with other features such as probability and time. However, as a first step this paper restricts attention to FSMs.

Information Theory has already been used in testing [49, 50, 51, 52, 53, 54, 55, 56, 57, 58]. In particular, the problem of choosing among different test suites has been addressed before [21, 17, 59]. However, as far as we are aware, this is the first work in which a measure inspired by Information Theory has been used to choose between test suites in a black-box testing framework, in particular, in testing from FSMs.

4. Biased Mutual Information

As previously explained, our goal is to increase test suite diversity with the hope that this will be reflected in the capability of the test suite to detect faults. Lower values of mutual information should be associated with higher diversity. We will consider both the input part of the test and the expected output with the goal of also increasing output diversity [60]. We can utilise output as well as input since we have a specification.

The intuition behind maximising diversity as a goal is very simple. Assume that we have an FSM with a set of input/output pairs labelling its transitions. If we select two different input/output pairs, and observe no failure, then we know that this selection traversed two different transitions of the FSM that represents the SUT. However, selecting one input/output pair twice leads to a scenario where we might traverse the same transition of the FSM twice (in particular, FSMs can have loops and so can return to a state met earlier). One might argue that this happens with probability $\frac{1}{m}$, where m is the number of times that the input/output pair labels a transition of the FSM; even for large m, we have a non-zero probability of traversing a transition more than once. This scenario also shows that we have to be careful when looking for measures of diversity, as the probability decreases when m increases. For example, the cases where $m = 2$ should be rather different to the case where $m = 200$. Therefore, one should not automatically discard test suites where a pair appears more than once. Instead, we should take into account the number of times a label appears in the FSM specification. We illustrate this with a simple example.

Example 1. *Consider the FSM given in Figure 1 and its test suites*

$$\mathcal{T}_1 = \{(i_2 i_4, o_1 o_4), (i_1 i_2, o_1 o_2)\}$$

and

$$\mathcal{T}_2 = \{(i_1 i_3, o_1 o_3), (i_1 i_2, o_1 o_2)\}$$

On the one hand, $\mathcal{T}_1$ has a mutual information of 0. Even though the input action i_2 appears twice in the test suite, we know that the pairs (i_2, o_1) and (i_2, o_2) represent different behaviours. On the other hand, $\mathcal{T}_2$ has a non-zero mutual information (applying the classical formula given in Definition 6 we have that this value is equal to 0.53). Therefore, a measure based on mutual information should choose the first test suite.

Initially, we would like to compute the mutual information of two tests. Each test is a sequence of input/output pairs. If we abstract out the position of the pairs in the sequence, we obtain a set of pairs. Given two tests t_1 and t_2, in order to compute the mutual information $I(\xi_{t_1}; \xi_{t_2})$ we need a definition of the probability distribution $\sigma_{\xi_t}(x)$ (see Definition 6). The first attempt was to give an *intuitive* definition of σ in which we used the uniform distribution as the probability distribution in the mutual information formula. That is, if a label appears in m transitions of the machine, then the probability of this label will be $\frac{1}{m}$; the probability that the pair x corresponds to a specific transition of specification M. Also, we replaced the notion of joint probability with a notion of *composition*. In this, if we have two transitions with the same input/output

pair (i, o), then the *composition* will return the common weight, that is, the weight of (i, o); otherwise, we multiply the weights. Alternatively, this composition can be seen as the product of the weights of each element reweighed by the number of repetitions of the input/output pair in the `FSM`.

Unfortunately, this choice does not induce a probability distribution over the pairs in the tests and so it is not a mathematically valid formulation of the mutual information between two tests. As a result, there is a need to explore alternatives. After several possibilities were considered (discussed in Section 7), we found that none of them consistently gave better results, so we kept the initial intuition to not restrict ourselves to random variables. Then, in the next definition the different occurrences of σ do not refer to probability distribution functions, but we keep this notation to retain the structure of the original formulation. As a result of the above, we will use the term $\sigma_t(x)$, even though t does not explicitly appear in the right hand side of the following formulae, because t is an implicit parameter; $\sigma_t(x)$ is only defined for x if $x \in t$. In the rest of this section, recall that we write $(i, o) \in_m M$ to denote that the pair (i, o) appears in m transitions of M and $(i, o) \in_n t$ denotes that the pair (i, o) appears n times in the test t.

Definition 7. *Let $M = (Q, q_{in}, \mathcal{I}, \mathcal{O}, T)$ be an `FSM`, t be a test for M and $x \in \mathcal{I} \times \mathcal{O}$ be an input/output pair such that $x \in t$. We let:*

$$\sigma_t(x) = \begin{cases} \frac{1}{m} & \text{if } x \in_m M,\ m \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

We define the composition of two tests t_1, t_2 of M, for input/output pairs $x_1 \in t_1$, $x_2 \in t_2$, as:

$$\sigma_{t_1,t_2}(x_1, x_2) = \begin{cases} \frac{1}{m_1} \cdot \frac{1}{\cancel{m_2}} \cdot \cancel{m_2} & \text{if } x_1 = x_2 \\ \frac{1}{m_1} \cdot \frac{1}{m_2} & \text{otherwise} \end{cases}$$

where $x_1 \in_{m_1} M$ and $x_2 \in_{m_2} M$. In the first case, note that $m_1 = m_2$ because we are looking for the same input/output pair in M. Also note that m_1 and m_2 are greater than zero because we request $x_1 \in t_1$ and $x_2 \in t_2$ to appear in M.

Finally, we redefine the mutual information of two tests as:

$$I(t_1; t_2) = \sum_{x_1 \in t_1} \sum_{x_2 \in t_2} \sigma_{t_1,t_2}(x_1, x_2) \cdot \log_2 \frac{\sigma_{t_1,t_2}(x_1, x_2)}{\sigma_{t_1}(x_1) \cdot \sigma_{t_2}(x_2)}$$

Note that the intuition assumes a uniform distribution over the set of transitions of M with the same label. We could choose another distribution for those probabilities by, for example, increasing the probability associated with transitions that are reached from the initial state after fewer transitions. However, this would complicate the computation of the measure and preliminary experiments did not show a significant improvement. Therefore, if the *real* distribution is not known then we use a uniform distribution in order to aid simplicity. Note that uniform distributions also have desirable properties (in particular, this distribution maximises entropy [61]). Naturally, we should not use uniform distributions if we have evidence that they are inappropriate (i.e. because using the true distribution is known to lead to better results). For example, this is the case if we have probabilistic user models indicating the probabilities with which users choose inputs [62]. Next we give a result allowing us to simplify the formulation.

Lemma 1. *Let M be an `FSM` and t_1, t_2 be tests for M. We have*

$$I(t_1; t_2) = \sum_{x \in t_2} n_x \cdot \frac{\log_2(m_x)}{m_x}$$

where m_x is such that $x \in_{m_x} M$ and n_x is such that $x \in_{n_x} t_1$.

Proof

In order to compute the terms of the sum defining mutual information, that is,

$$\sigma_{t_1,t_2}(x_1, x_2) \cdot \log_2 \frac{\sigma_{t_1,t_2}(x_1, x_2)}{\sigma_{t_1}(x_1) \cdot \sigma_{t_2}(x_2)} \tag{1}$$

we will distinguish two cases: $x_1 = x_2$ and $x_1 \neq x_2$. First, let us consider $x_1 = x_2$. Note that $\sigma_{t_1}(x_1) = \sigma_{t_2}(x_2)$, because these values depend only on M and x. In addition, the composition of an element of a test and itself is the probability of the element (as we stated in Definition 7). Therefore, $\sigma_{t_1,t_2}(x, x) = \sigma_{t_1}(x)$. Now, taking into account Definition 7 we have that if $x_1 = x_2$ then the previous term is equal to

$$\frac{1}{m_x} \cdot \frac{1}{m_x} \cdot m_x \cdot \log_2 \left(\frac{\frac{1}{m_x} \cdot \frac{1}{m_x} \cdot m_x}{\frac{1}{m_x} \cdot \frac{1}{m_x}} \right) = \frac{1}{m_x} \cdot \log_2 \left(\frac{1}{\frac{1}{m_x}} \right)$$

and, simplifying, we conclude that if $x_1 = x_2$, then the term given in Equation 1 is equal to

$$\frac{\log_2(m_x)}{m_x}$$

Now, let us consider $x_1 \neq x_2$. In this case, $\sigma_{t_1,t_2}(x_1, x_2) = \sigma_{t_1}(x_1) \cdot \sigma_{t_2}(x_2)$ and, therefore, the term given in Equation 1 becomes equal to 0 because we have $\log_2(1)$ as one of the factors.

Putting together these two cases in the Mutual Information formula given in Definition 7, we obtain the following expression:

$$I(t_1; t_2) = \sum_{x_2 \in t_2} \sum_{\substack{x_1 \in t_1 \\ x_1 = x_2}} \frac{\log_2(m_x)}{m_x}$$

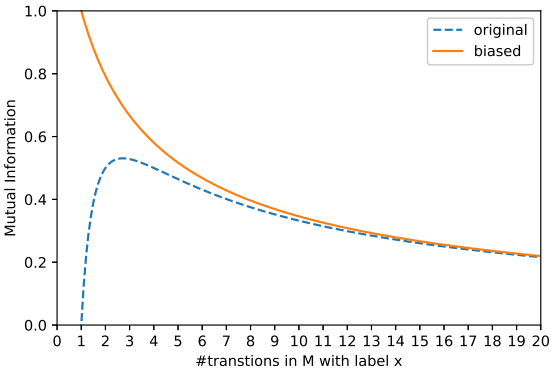

Figure 2: Measure comparison plot.

Note that in the inner sum, for a given x_2, we are always adding the same value. Therefore, we can simplify it as a multiplication of that value times the number of times it is added. This last factor corresponds to the number of times $x_1 = x_2$ appears in the test t_1, that is, the value n_x such that $x_2 \in_{n_x} t_1$. This results in the following expression:

$$I(t_1; t_2) = \sum_{x_2 \in t_2} n_x \cdot \frac{\log_2(m_x)}{m_x}$$

that can easily be rewritten as:

$$I(t_1; t_2) = \sum_{x \in t_2} n_x \cdot \frac{\log_2(m_x)}{m_x}$$

where m_x is such that $x \in_{m_x} M$ and n_x is such that $x \in_{n_x} t_1$. □

An important remark about this formula is that it is not monotonic and it is equal to 0 if all the transitions of the specification have different input/output pairs. Since we are interested in values that are useful when comparing test suites (therefore, we need monotonicity and we should avoid "division by zero"), we solve this problem with a simple transformation. The dashed curve in Figure 2 shows the behaviour of the previous formula. We make a small translation in the X axis of the logarithm of the formula, so that its behaviour is the one given by the solid curve.

Definition 8. *Let M be an* FSM *and t_1, t_2 be tests for M. We say that the* biased mutual information *(bmi) of t_1 and t_2 is given by*

$$bmi(t_1; t_2) = \sum_{x \in t_2} n_x \cdot \frac{\log_2(m_x + 1)}{m_x}$$

where m_x is such that $x \in_{m_x} M$ and n_x is such that $x \in_{n_x} t_1$.

In the following example we illustrate the importance of this translation.

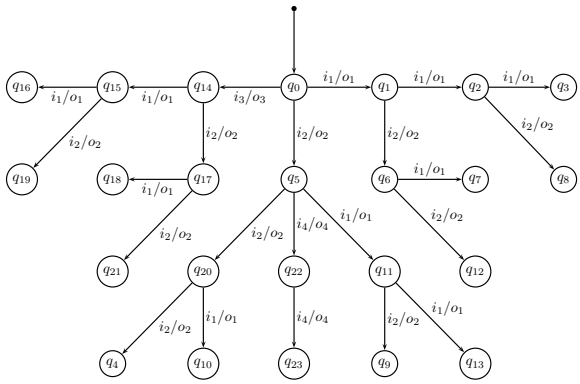

Figure 3: Another example of FSM.

Example 2. *Consider the* FSM *M depicted in Figure 3 and the test suites*

$$\mathcal{T}_1 = \{t_1 = (i_2 i_4 i_4, o_2 o_4 o_4),\ t_2 = (i_3 i_2 i_1, o_3 o_2 o_1)\}$$

and

$$\mathcal{T}_2 = \{t_3 = (i_3 i_1 i_1, o_3 o_1 o_1),\ t_4 = (i_3 i_2 i_2, o_3 o_2 o_2)\}$$

Note that the only pair appearing in both t_1 and t_2 is (i_2, o_2) (this input/output pair appears 9 times in M); similarly, the only common pair for t_3 and t_4 is (i_3, o_3) (this appears once in M). The (biased) mutual information of each test suite can be computed as follows:

$$bmi(t_1; t_2) = \frac{\log_2(9+1)}{9} = \frac{\log_2(10)}{9} \approx 0.3691$$
$$bmi(t_3; t_4) = \frac{\log_2(1+1)}{1} = \frac{\log_2(2)}{1} = 1$$
$$I(t_1; t_2) = \frac{\log_2(9)}{9} \approx 0.3522$$
$$I(t_3; t_4) = \frac{\log_2(1)}{1} = 0$$

Therefore, the first test suite would be better if we consider biased mutual information, but would be worse if we consider mutual information. In principle, we should prefer $\mathcal{T}_1$ because it is more likely that it will check more transitions than $\mathcal{T}_2$. In fact, in this example we know that the second test suite will traverse the same transition twice.

The biased mutual information between two tests will be used as the basis of a measure for a test suite. The idea is that we need to compute the cumulative amount of biased mutual information between all the pairs of tests. Given a test suite $\mathcal{T}$, this will be denoted by $\alpha(\mathcal{T})$ in the definition of the Biased Mutual Information of $\mathcal{T}$. Therefore, if we have a specification M and a test suite $\mathcal{T} = \{t_1, \ldots, t_k\}$, then we apply Definition 8 to all the pairs of tests included in $\mathcal{T}$:

$$\alpha(\mathcal{T}) = \sum_{i=1,\ldots,k} \sum_{j=i+1,\ldots,k} \sum_{x \in t_i} n_x \cdot \frac{\log_2(m_x + 1)}{m_x}$$

where m_x is such that $x \in_{m_x} M$ and n_x is such that $x \in_{n_x} t_j$.

In addition, we need to take into account repetitions in each test belonging to the test suite. Intuitively, we should penalise test suites that have tests with many repeated input/output pairs, even if these pairs do not appear in other tests of the suite. We present a simple example to motivate why we need to take this into account.

Example 3. *Consider the test suites*

$$\mathcal{T}_1 = \{t_1 = (i_1 i_1 i_1, o_1 o_1 o_1),\ t_2 = (i_2 i_3 i_4, o_2 o_3 o_4)\}$$

and

$$\mathcal{T}_2 = \{t_3 = (i_1 i_7 i_9, o_1 o_7 o_9),\ t_4 = (i_2 i_3 i_4, o_2 o_3 o_4)\}$$

There is no mutual information between t_1 and t_2 (similarly between t_3 and t_4). However, we should take into account the repetition in t_1 and we should prefer $\mathcal{T}_2$.

If we have a specification M and a test suite $\mathcal{T}$, then for each test $t \in \mathcal{T}$ we have that this *self-redundancy* factor, denoted by $\beta(t)$, is defined as:

$$\beta(t) = \sum_{x \in t} \frac{(n_x - 1) \cdot n_x}{2} \cdot \frac{\log_2(m_x + 1)}{m_x}$$

where m_x is such that $x \in_{m_x} M$ and n_x is such that $x \in_{n_x} t$.

In the previous formula, $\frac{(n_x-1)\cdot n_x}{2}$ is the sum of the first $n_x - 1$ integers. This represents the number of pairs (x_1, x_2) of input/output pairs such that x_1 and x_2 are both in the test and $x_1 \neq x_2$. In addition, $\frac{\log_2(m_x+1)}{m_x}$ is the biased mutual information between those pairs.

If we appropriately combine these factors, then we obtain our formula for the *Biased Mutual Information* (BMI) of a test suite.

Definition 9. *Let M be an FSM and $\mathcal{T} = \{t_1, \ldots, t_k\}$ be a test suite for M. We have*

$$BMI(\mathcal{T}) = \alpha(\mathcal{T}) + \sum_{i=1,\ldots,k} \beta(t_i)$$

In the next section we describe the experiments used to evaluate BMI.

5. Empirical evaluation

In this section we describe the experiments used to evaluate the proposed measure (BMI) and assess its ability to help the tester to choose *good* test suites. We performed several experiments with different models and in each we:

- *Derived test suites* by randomly traversing the specification.
- *Generated mutants* of the specification. The rationale is that test suite $\mathcal{T}_1$ is better than test suite $\mathcal{T}_2$ if $\mathcal{T}_1$ kills[5] more mutants than $\mathcal{T}_2$.

In the experiments we used 241 real FSMs from a benchmark [63]. We also used randomly generated FSMs, with this providing us with a larger set of subjects and also the ability to explore how performance changes as we vary FSM properties such as the alphabet size. There were two main reasons for us using randomly generated mutants in the experiments. First, the benchmark FSMs do not come with faulty versions and, indeed, we are not aware of any FSM benchmark that includes faulty programs or models. Second, we do not have a general *fault model* representing potential (faulty) implementations of a specification. If we had such a fault model, then we could have used it instead of the randomly generated mutants. Observe that since we used FSMs it is possible to identify equivalent mutants in low-order polynomial time and so we were able to remove them. Therefore, we have an implicit fault model, which is all non-equivalent mutants that can be generated using our mutation operators.

In the rest of this section, first, we state the research questions. We then explain the experimental design before discussing the results of the experiments and what these tell us about the research questions. All the code, benchmarks and results from the experiments are available at `https://github.com/Colosu/BMI-test-selection`.

5.1. Research questions

In order to *evaluate* BMI, we first checked how *well* it works.

The main motivation for the work described in this paper is that we would like to be able to choose between alternative test suites. Thus, as a first step we assessed whether BMI is effective in guiding such a choice.

Research Question 1. *Given a pair of test suites with the same length, will the one with lower Biased Mutual Information tend to have higher fault detection ability?*

We also wanted to address the related question of whether lower levels of Biased Mutual Information are associated with higher fault coverage; whether a correlation exists.

Research Question 2. *Are lower levels of Biased Mutual Information associated with higher fault coverage?*

If we have a positive answer (with statistical significance) to the above questions, then we would like to see how BMI compares to the currently proposed Information Theory approach, the test set diameter (TSDm) measure[6].

[5] A test suite kills a mutant if the test suite contains a test case such that the specification and mutant produce different outputs in response to this test case.

[6] In Section 5.5 we describe how TSDm has been used in test selection.

Set number	Set size	Range # states	Range outgoing transitions	Range input alphabet size	Range output alphabet size
1	241	[3, 156]	[0, 130]	[2, 130]	[2, 65]
2	100	50	[2, 5]	5	5
3	100	50	[2, 5]	25	25

Table 1: Properties of the FSM sets.

Value of A	# runs [0.5, 0.6)	# runs [0.6, 0.7)	# runs [0.7, 0.8)	min value	max value	% success (mean)
1/5	41	9	0	0.536232	0.616319	58.4039%
2/5	12	38	0	0.556391	0.649412	61.6790%
3/5	1	48	1	0.597360	0.703226	65.1793%
4/5	3	44	3	0.588235	0.712062	64.0050%

Table 2: Summary of the results of the experiment with real FSMs.

Research Question 3. *Do test suites selected by BMI have higher fault coverage than those selected by test set diameter (ITSDm)?*

Finally, we wanted to check whether the time taken to compute BMI might be better used in executing additional tests.

Research Question 4. *How does the time to execute the selection method scale as the length of the test suite increases? How does the time needed to compute the selection method relate to the time needed to apply a test suite?*

By the length of a test suite we mean the sum of the lengths of the tests (sequences).

5.2. Experimental subjects

We designed a series of experiments in order to address the research questions. We used three sets of FSMs.

- Set1 A benchmark of 335 FSMs recently collected [63]. From this, we selected the 241 deterministic FSMs. These FSMs represent real-world systems, ranging from the classical *coffee machine* to more complex systems such as ATMs, circuits, network protocols and X-ray systems.
- Set2 A set of 100 randomly generated FSMs with 50 states and input and output alphabets with 5 elements. Each state was given a random number, between 2 and 5, of outgoing transitions (except state 50, which had no outgoing transitions).
- Set3 Another set of 100 randomly generated FSMs with 50 states and input and output alphabets of size 25. Again, each state was given a random number, between 2 and 5, of outgoing transitions (except state 50, which had no outgoing transitions). These were used in order to explore the effect of having a larger input alphabet.

In Table 1 we summarise the main parameters of the FSMs in these three sets.

We used the first set (Set1) in order to evaluate BMI on real FSMs. This provided FSMs with varying numbers of states and varying alphabet sizes. While this is useful, there is only a limited number of FSMs and the use of these also limits the control we have (e.g. if we want to vary the size of the alphabets). This motivated the use of the other two sets of FSMs.

In order to assess the fault detection ability of a test suite, we generated mutants of the specification. We only use the mutation operator that modifies the target state of a transition because a preliminary experiment showed that faults produced by mutations on the labels of the transitions are much easier to detect during testing. The use of mutants is a standard approach to assessing fault detection effectiveness; see the discussion in the introduction. We say that a test t *kills* a mutant N of FSM M if either M and N produce different output sequences in response to t or $t = t'\ i/o\ t''$ for an input action i, output action o and prefix t' such that t' takes N to a state from which there is no transition with input action i.

For an FSM specification M, we generated test suites by randomly traversing M. The FSMs typically had at least one sink state, with no outgoing transitions: if such a sink state was reached then a new test was started (at the initial state).

We now describe the actual experiments used to address the research questions and the results of these.

5.3. Using BMI to help guide test suite choice

These experiments explored whether BMI is valid for our purposes. Initially, we used the real FSMs (Set1). For each FSM, we generated two test suites, from the FSM, of length $num\_states \times alphabet\_size \times A$ where we used the following values for A: $\{\frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}\}$. Recall that we generated a test suite by randomly traversing the FSM, starting a new test (sequence) whenever a sink state is met; the length of a test suite is thus the sum of the lengths of the individual tests (sequences). We let the test suite length depend on the number of states because the number of states of the FSMs in the benchmark varied, from 3 to 156 states; we used a length proportional to the potential maximum number of transitions of the FSM.

For each pair of test suites, we computed BMI values to determine which test suite would be selected if we chose the one with lower BMI. We also produced 100 mutants of the original FSM; recall that the mutation operator modifies the target state of a transition. We applied each test suite to all of the mutants to determine how many mutants were killed by a test suite. We repeated this procedure 10 times for each FSM, obtaining 10 pairs of test suites, and then computed how many times the test suite selected using BMI was the one that killed more mutants (the *percentage of success*). Where two test suites in a pair had the same mutation score or the same BMI, we replaced this pair of test suites with another randomly generated pair. We repeated this process 50 times for each value of A.

Type of test suite	# runs $[0.5, 0.6)$	# runs $[0.6, 0.7)$	# runs $[0.7, 0.8)$	min value	max value	% success (mean)
with repetitions	15	32	3	0.540816	0.714286	62.2402%
without repetitions	13	34	3	0.520408	0.737374	62.4924%

Table 3: Summary of the results of the experiment with controlled FSMs with alphabet of size 5.

Table 2 summarises the results (the full results are given in Tables A.6, A.7, A.8 and A.9 of the appendix). We always had a percentage of success that was higher than 58%, and in most cases it was higher than 60%. These results suggest that BMI performs better than random selection and that the difference increases for larger test suites.

We then carried out experiments to assess BMI in a more controlled scenario, using the following approach. We started with the 100 randomly generated FSMs in Set2 (50 states, between 2 and 5 outgoing transitions for each state, and alphabets of size 5). We used the process described above but this time generating 1000 mutants and fixing the test suite length to 100, because now we have a fixed number of states. We repeated the whole process 50 times, obtaining 50 different percentages of success. Since each percentage of success was obtained from 100 different FSMs, the calculation of a mean percentage of success required 5000 repetitions.

One issue in the design of the experiments was whether we should check for redundancy in a test suite, where *redundancy* corresponds to the case where one test (sequence) is a prefix of another. We carried out experiments to explore the effect of this choice on our results. In these experiments, we had two scenarios: 1) randomly generated test suites without checking for prefixes; and 2) discarding prefixes when randomly generating test suites. Table 3 provides a summary of the results (full results can be found in Table A.10 in the appendix), where we can see that very similar results were obtained in the two scenarios. Also, we can see that in both cases, all the values are in the range $[0.5, 0.8)$, with the majority of them belonging to the range $[0.6, 0.7)$. These observations suggest that the effectiveness of BMI is not affected by whether we allow the test suite to have redundancy. As a result, the remaining experiments consider the case where we do not allow the test suite to have redundant tests (in practice, this will typically be the case).

We repeated the experiments using the randomly generated FSMs with input alphabet of size 25 (Set3), using test suites without redundancy. The mean percentage of success was 75.0605% (full results can be found in Table A.12 in the appendix). These results are much better than the results from the previous experiments and this suggests that BMI works better in FSMs with larger input alphabets.

Over all the obtained results, we performed an homogeneity of variance check and a statistical hypothesis test whose null hypothesis was that random selection and ordering on BMI give similar results. The homogeneity check showed that there is no homogeneity, so we cannot use the ANOVA test. Instead, we applied a Kruskal-Wallis H-test where we tested whether the results of the experiment are distributed as the distribution that would arise from random selection. We assumed that the distribution that arises from random selection would be a Poisson distribution with $\lambda = 50$[7]. Then, we computed the p-values and, in all cases, the null hypothesis was rejected with a p-value extremely close to zero. We also performed an effect size measure[8]. We computed Cliff's delta statistic and we obtained large effect sizes (all higher than 1). This reinforces the conclusions derived from the previously computed p-values.

5.4. Assessing correlation

The results described above provide evidence that BMI works when choosing between two test suites. Ideally, we would also like a measure that correlates with the fault detection ability of a test suite.

We repeated the experiment described in the previous section using the same FSMs (Set2) but different test suites and mutants. For each FSM, we then computed the correlation between BMI and the mutation score (i.e. the number of killed mutants). We obtained a mean Pearson correlation of -0.369134 and a mean Spearman correlation of -0.356978. The fact that the correlations are negative implies that lower BMI is associated with higher mutation scores, addressing RQ2. These correlations are consistent with the results from the previous experiment. The full results are displayed in Table A.11 (see the appendix).

We repeated the experiment with the FSMs with alphabet size 25 (Set3). The results show a stronger (negative) correlation. Specifically, we obtained a mean Pearson correlation of -0.650256 and a mean Spearman correlation of -0.634711. The full results can be found in Table A.13 (see the appendix).

Again, we checked whether the results are statistically significant. Here our null hypothesis is that there is no correlation (that is, the correlation is equal to 0). First, we performed an homogeneity of variance check that told us that there is no homogeneity, and then we applied a Kruskal-Wallis H-test where we tested whether the results of the experiment are distributed as the distribution that would arise from random selection. We assumed that the distribution that arises from random selection would be a normal distribution of $\mu = 0$ and $\sigma = 0.1$. Then, we computed the final p-values that state the significance of our results. Again, the null hypothesis was rejected, with the returned p-values being very close to zero. We also performed an effect size measure, computing the Cliff's delta statistic. We obtained large effect sizes (all higher than 0.8). This reinforces the conclusions derived from the previously computed p-values.

[7] In each selection, we have a binomial distribution (50% probability of choosing the better test suite), and the repetition of this binomial distribution produces a Poisson distribution of $\lambda = 50$.

[8] https://github.com/txt/ase16/blob/master/doc/stats.md

5.5. Comparison with TSDm

We performed additional experiments to compare BMI with a previous information theoretic proposal: the Test Set Diameter (TSDm) measure.

In order to compare the measures, we started by using Set2 (100 FSMs with alphabets of size 5). For each FSM, we randomly generated two test suites, as in previous experiments. We then computed the ITSDm and BMI values of both test suites and recorded which one was better for each measure. We then produced 1000 mutants of the original FSM. As before, mutations changed the target state of a transition. Finally, we computed the mutation score of each test suite[9]. We performed this procedure for each FSM and then computed how many times the test suite selected using BMI had the higher mutation score, how many times it was the one selected using ITSDm, and how many times both measures selected the same test suite. Additionally, in this last case we checked how many times the test suite selected was the one with the higher mutation score, and how many times it was the one with lower mutation score. We presented the results as percentages. We repeated the whole process until we obtained 50 different sets of the corresponding four percentages (each percentage was obtained from 100 different FSMs and 1000 mutants per FSM).

Additionally, we computed the mean execution time for the computation of each measure, in order to compare the differences in performance with the differences in execution time.

In the experiments, on average, 55.84% of the time BMI selected the best test suite, while ITSDm selected the best test suite only 46.72% of the time. Moreover, in 30.16% of cases, BMI chose the best test suite and ITSDm chose the worst one, while the other way around only occurred 21.04% of the time. Finally, 23.12% of the time both measures failed to select the best test suite. Full results are displayed in Table A.14 of the Appendix.

Regarding execution time, we found that the mean time taken to compute BMI was 0.043 seconds, while the corresponding value for ITSDm was 0.174 seconds. Thus, the proposed measure required 75.29% less time than ITSDm, while at the same time obtaining better results. These results are promising: the proposed measure tended to be a better guide, than ITSDm, when choosing between alternative test suites and took less time to compute.

We repeated this experiment using Set3 (100 FSMs with alphabets of size 25) and we again obtained better results for BMI. On average, 75.78% of the time BMI selected the best test suite, while for ITSDm this happened only 46.66% of the time. Moreover, 40% of the time BMI chose the best test suite while ITSDm chose the worst one, while the other way around only happened 10.88% of the time. Finally, 13.34% of the time both measures failed to select the best test suite. Full results are displayed in Table A.15 of the Appendix. With respect to time, the execution time results are better too, with the computation of BMI taking a mean time of 0.045 seconds, while the corresponding time for ITSDm was 0.32 seconds. This corresponds to a 85.94% saving in execution time.

Similar to the comparison with random ordering, we analysed the statistical significance of the results of the experiments, with the null hypothesis being that the two approaches (using ITSDm and BMI) give similar results. Again, we performed an homogeneity of variance check that told us that for the experiment with Set2 there is homogeneity, but for the experiment with Set3 there is none. Therefore, we applied an upper-tailed ANOVA test to the results for the experiments with Set2 (where we tested whether the results concerning BMI and the ones concerning ITSDm come from the same distribution) and we applied a Kruskal-Wallis H-test to the results for the experiment with Set3 (where we tested if the BMI results come from the same distribution as the ITSDm results). We computed the p-values, obtaining p-values close to zero, so we reject the hypothesis. We performed two effect size measures (one for each experiment), computing the Cliff's delta statistic. We obtained large effect sizes (greater than 0.89).

5.6. Execution time

On average we needed 0.00083786 seconds to compute BMI for test suites of length 100.[10] In order to simulate the use of BMI, we extended the experiment for RQ1 as follows. First, we recorded the (mean) time needed to compute BMI for test suites of length 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000. We also determined the mean time needed to apply these test suites to mutants (averaged over 1000 mutants) assuming that the time needed to execute a transition is 0, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1 and 1 seconds (test execution terminated once a failure occurred). We compared the mean time to compute BMI with the mean time needed to apply a test suite, obtaining the results in Figure 4.

Note that the mean time needed to compute BMI includes the time needed to determine how many times each input/output pair appears in the specification. This is computed only once for a given FSM. Therefore, although we included this in the time needed to compute BMI, when comparing test suites we need to carry out this computation only once.

As we can see, as long as the time needed to execute a transition is greater than 0.001 seconds, computing BMI is much faster than applying test suites. If this time is higher than 0.1 seconds then we need minutes (or even hours if it is higher than 1 second) to apply a test suite. As an additional result of the experiments, we validated that

[9] A mutant is killed if an input action is received in a state where it is not defined or the wrong output action is observed.

[10] The experiments were run on a GNU/Linux machine with an AMD® Ryzen threadripper 1920X at 3.50GHz × 12 cores and with 32GB of RAM (although only one core was running at a time and we did not use more than 4Gb of RAM).

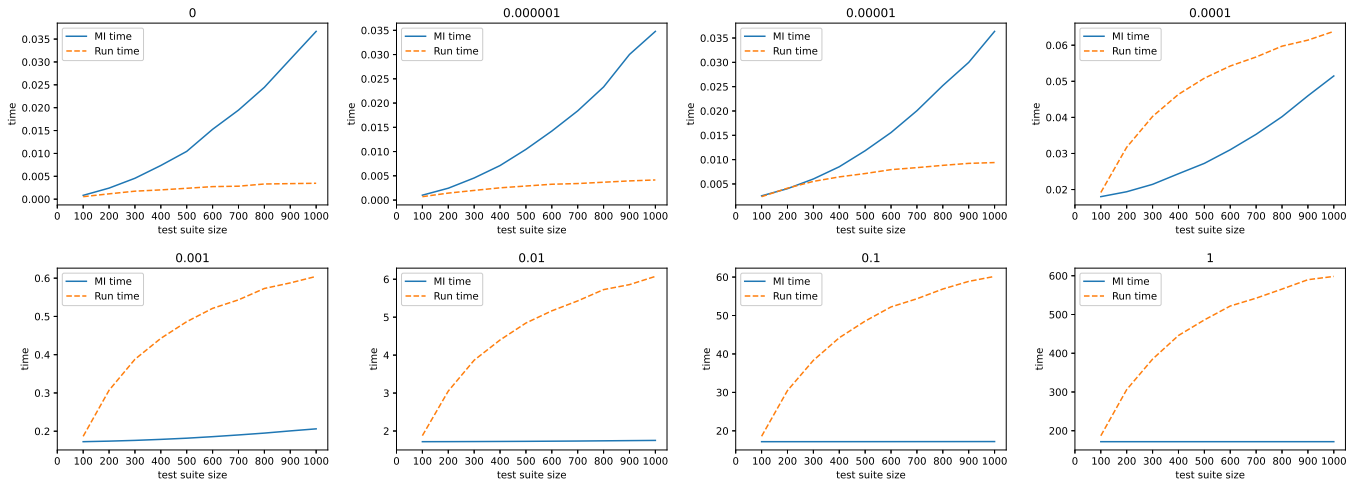

Figure 4: Time comparison plots (left to right, from top to bottom, with transition time of 0, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1 seconds).

the time needed to compute BMI scales (approximately) quadratically with respect to the test suite length.

5.7. Assessing fault complexity and test suites coverage

We decided to assess the fault complexity and test suite coverage of the randomised elements we use in our experiments. In order to do so, we performed a new experiment. For the FSMs in Set2 and Set3, we randomly generated 100 test suites and 1000 mutants. We then computed, for each mutant, how many test suites killed this mutant and the coverage of each test suite, both in terms of transitions traversed and states visited, with respect to each original FSM.

For Set2, each mutant was killed by between 52.78% and 96.01% of the test suites, with a mean value of 73.45%. Full results are displayed in Tables A.16 and A.17 of the Appendix. For Set3, the results range from a minimum of 0% of test suites killing the mutant to a maximum of 90.21%, with a mean of 43.63%. Full results are displayed in Tables A.18 and A.19 (minimum results are not displayed this time because there was always a mutant that is not killed by any test suite).

Regarding coverage, we first computed the state coverage and transition coverage of each test suite. In addition, in order to normalise the values we required an upper bound on the *maximum achievable coverage* since this would allow us to compute the ratio between obtained coverage and an upper bound. Let us illustrate these concepts with a simple example.

Example 4. *Consider an FSM with* 50 *states and* 150 *transitions and a test suite of length* 75*. An upper bound of the maximum transition coverage that we can reach is* 50% *(the test suite will traverse at most* 75 *of the* 150 *different transitions) while the maximum state coverage that we can reach is* 100%*. Note that these values are upper bounds but need not be least upper bounds because the structure of the FSM might induce smaller real maxima. For example, if we consider the FSM depicted in Figure 1, then any test suite with* 6 *inputs will have a maximum transition coverage of* $\frac{6}{7} \cdot 100\%$ *although our estimate will be* 100% *because the FSM has* 6 *transitions.*

Let us suppose that the test suite has a 60% *state coverage and* 40% *transition coverage. Then we know that our test suite covers at least* 60% *of the maximum number of states that can be covered by any test suite of its size (as already said, the real value might be higher). Similarly, our test suite covers at least* 80% *of the maximum number of transitions that can be covered with a test suite of its size.*

We did not compute a true maximum value of coverage because of the time required. For example, it is straightforward to prove that the problem of computing the maximum state coverage for a given FSM and test suite length is NP-hard (by reduction from the Hamiltonian path problem [64]).

For Set2 we obtained an average state coverage of 77.42%, while the estimate of maximum achievable state coverage was always 100%. For transition coverage, Set2 obtained an average of 40.39% total coverage. If we *normalise* this number with respect to the estimate of the maximum achievable transition coverage we have 69.49%. Regarding Set3, we have 78.54% state coverage (again, the estimate of the maximum achievable state coverage was 100%), 41.16% total transition coverage, and 69.91% transition coverage with respect to the estimate of the maximum achievable coverage. The full results can be found at Tables A.20 and A.21 for Set2, and Tables A.22 and A.23 for Set3.

5.8. Comparison with coverage guided selection

During the analysis of coverage, a concern was raised: whether our BMI measure was simply a proxy for transition coverage. In order to determine whether this was the case, we developed a new experiment. This experiment was essentially the same as the experiment where we compared BMI with TSDm, but this time we compared BMI with an approach in which we select the test suite with higher transition coverage (over the specification) instead of the one with a higher ITSDm value.

We took Set2 and Set3. For each `FSM` in one of these sets, we generated two test suites of length 100, we computed both BMI and the transition coverage of each test suite, and marked the one chosen by each method. Then, we generated 1000 mutants of the `FSM`, computed how many mutants were killed by each test suite, and compared them. Then, we computed how many times both methods select the same test suite and how many times they did not. When they coincide, we consider two cases: they selected the test suite that killed more mutants or they selected the test suite that killed fewer mutants. When they differed, we determined which test suite killed more mutants and which one was the method that choose this test suite. We repeated this full experiment 50 times.

The results for Set2 are interesting: in 65.5% of the cases, BMI selected the same test suite as coverage. In 54.78% of the cases BMI selected the test suite that killed more mutants while coverage selected that test suite 62.16% of the time. In 13.56% of the experiments BMI selected the test suite that killed more mutants while coverage selected the other test suite. In 20.94% of the experiments the situation was the other way round. Finally, both measures failed to select the best test suite 24.28% of the time.

Regarding computation time, we have that BMI needed a mean of 0.00295048 seconds while coverage required a mean of 0.0583516 seconds. This constitutes a 94.94% saving.

The results for Set3 are not so different: BMI and coverage selected the same test suite 85.78% of the time. In addition, in 75.96% of the experiments BMI selected the test suite that killed more mutants, while coverage selected this test suite 81.78% of the time. In 4.2% of cases, BMI selected the test suite that killed more mutants and coverage selected the other test suite; the situation was the other way round 10.02% of the time. Finally, in 14.02% of the experiments, both measures failed to select the test suite that killed more mutants. Regarding computation time, BMI took 0.00137012 seconds on average and coverage took 0.016663 seconds on average. This is a saving of 91.78%.

Similar to the other comparisons, we analysed the statistical significance of the results of the experiments assuming as the null hypothesis that the two approaches (using transition coverage and BMI) give similar results. Again, we performed an homogeneity of variance check that told us that there is no homogeneity for our results. Therefore, we applied a Kruskal-Wallis H-test to the results, where we tested if the BMI results come from the same distribution as the transition coverage results. We computed the p-values, obtaining p-values close to zero, so we reject the hypothesis that the results are statistically equivalent. We also computed the effect-size using Cliff's delta statistic. We obtained large effect sizes (greater than 0.82).

Based on the above, we can conclude that BMI is not a proxy for transition coverage. In addition, although BMI is not as effective as coverage, it takes less time to compute, which can be a critical fact in some industrial size cases. There is therefore a trade-off between the effectiveness of the approaches and the time taken.

In some ways it is not too surprising that transition coverage can be more effective than BMI since transition coverage uses more information about the `FSM`: it uses information about the states associated with each input/output pair rather than just input/output pair frequency. It is therefore, for example, able to identify cases where the same input/output pair appears but they represent different parts (transitions) of the specification.

In principle, it should be possible to extend the definition of BMI to use information about the transitions executed instead of input/output pair frequency. However, this would have at least two disadvantages. The first disadvantage is simply that it would be necessary to traverse the `FSM` specification and this would increase the computation time. The second, and rather more important, disadvantage is that such a revised definition of BMI would only be applicable in situations in which we have an `FSM` specification. This would go against the aim of developing a measure that can be used in a range of scenarios. For example, in principle it should be possible to use BMI without having a specification, as long as it is possible to include some initial random testing in order to provide estimates of input/output pair frequency; clearly, transition coverage cannot be applied in such situations.

5.9. Summary

We now summarise what the results tell us about the research questions.

Research Question 1. *Given a pair of test suites with the same length, will the one with lower Biased Mutation Information tend to have higher fault detection ability?*

The answer to this question is affirmative: BMI tended to select test suites with higher fault coverage than random selection. In the experiments BMI selected the best test suite 62.4924% of the time when we used `FSM`s with an alphabet of size 5 (Tables 3 and A.10 in the appendix) and 75.0605% of the times when we used `FSM`s with an input alphabet of size 25 (Table A.12 in the appendix). BMI also selected test suites with higher fault coverage in the scenario with real `FSM`s.

Research Question 2. *Are lower levels of Biased Mutual Information associated with higher fault coverage?*

In the experiments, lower levels of BMI were correlated with higher fault coverage. We can conclude this from Tables A.11 and A.13 (see the appendix), where we can observe that BMI was negatively correlated with the mutation score of the tests, with a mean correlation of -0.369134 for FSMs with an alphabet of size 5 and a mean correlation of -0.650256 for FSMs with an alphabet of size 25.

Research Question 3. *Do test suites selected by BMI have higher fault coverage than those selected by test set diameter (ITSDm)?*

The answer was again positive: BMI selected test suites with higher fault coverage than ITSDm. We can conclude this from Tables A.14 and A.15 (see the appendix), where we can observe that BMI outperformed the ITSDm measure when selecting the best test suite from a set of 2 randomly generated test suites.

Research Question 4. *How does the time to execute the selection method scale as the length of the test suite increases? How does the time needed to compute the selection method relate to the time needed to apply a test suite?*

Figure 4 shows that the time needed to compute BMI increased (approximately) quadratically with respect to the length of the test suite. Also, the time needed to compute the selection method was smaller than the time needed to apply a test suite as long as we need at least 0.001 seconds to execute each transition (Table A.12 in the appendix).

6. Threats to validity

In this section we discuss the possible threats to the validity of the results of the experiments.

Concerning threats to internal validity, which consider uncontrolled factors that might be responsible for the obtained results, the main threat is associated with possible faults in the tools. In order to reduce the impact of this threat we tested the code with carefully constructed examples for which we could manually check the results. In addition, we repeated the experiments many times to reduce the impact of randomisation. Another important threat was the processor reschedule policy, which can affect the recorded times. In order to reduce the impact of this threat, we abstracted the time computation and only computed small enough time values so that the reschedule policy does not affect them. In addition, we repeated the tests and computed mean values. Another threat was that Normalised Compress Distance (NCD), used in TSDm, performs poorly with short strings. We therefore used relatively long strings; for a test suite of length 100 we have strings of $(100\ \textit{input actions} + 100\ \textit{output actions}) \times 2\ \textit{characters per action} = 400\ \textit{characters}$. We made this choice because it seemed to be a reasonable test length for the FSMs used and also because previous work noted that compression did not work for strings of length less than 128 [21]. It is possible that longer test sequences would lead to more effective compression and so NCD being a better guide; this is an issue that could be addressed by further experiments.

The main threat to external validity, which concerns conditions that allow us to generalise our findings to other situations, is the choice of FSMs. Such a threat cannot be entirely addressed since the population of FSMs is unknown and it is not possible to sample from this (unknown) population. In order to reduce the impact of this threat we used randomly generated FSMs and a carefully constructed benchmark. A minor external threat is that if we use large alphabets for randomly generated FSMs then very few input/output pairs will be repeated. In order to address this threat we performed the experiments with different alphabet sizes, as shown in Section 5.

Finally, we considered threats to construct validity, which are related to whether we are measuring properties of interest. The aim of testing is to find faults and so it is clear that the fault detection ability of a test suite is of interest, as is the time used to obtain a solution (since there are finite resources). We used mutants to assess fault detection ability and ideally we would have also used real faults. However, we are not aware of benchmark FSMs with faulty versions; state-based specifications are widely used in certain areas of industry (e.g. automotive and avionics) but associated companies appear not to have provided faulty versions. The open source community provides a source of faulty code but not faulty models.

7. Final discussion: alternative definitions

We have shown that BMI is interesting, potentially useful and that the time needed to compute it is negligible when compared to the time needed to apply extra testing. However, it is possible that some of the design decisions were not optimal and, indeed, there were alternative choices. In this section we describe some such alternatives and the results of additional experiments carried out to evaluate these. The decisions made, when designing BMI, fall into the following classes:

1. The formula used to define $bmi(t_1; t_2)$ in terms of the 'distributions';
2. Whether the variables used in the mutual information formula were probability distributions.

We now describe some alternatives to the choices made, along with the results of experiments that evaluated these.

7.1. The definition of $bmi(t_1; t_2)$

During the rest of this section, remember that we write $(i, o) \in_m M$ to denote that the pair (i, o) appears in m transitions of M and $(i, o) \in_n t$ denotes that the pair (i, o) appears n times in the test t.

The definition of $bmi(t_1;t_2)$ used a transformation of the X-axis and we might have chosen a different transformation. In order to explore the impact of using a larger transformation, we considered:

$$bmi_2(\xi_{t_1};\xi_{t_2}) = \sum_{x \in t_2} n_x \cdot \frac{\log_2(m_x + 2)}{m_x + 2}$$

where m_x is such that $x \in_{m_x} M$ and n_x is such that $x \in_{n_x} t_1$.

Another option is to use the formula $\frac{n}{m}$, instead of $\frac{1}{m}$, to compute the values of $\sigma_t(x)$ (where $x \in_n t$ and $x \in_m M$). This way, we take into account also how many times the input/output pair is repeated in the test. This leads to the following formula:

$$bmi_3(\xi_{t_1};\xi_{t_2}) = \sum_{x \in t_2} n_1 \cdot n_1 \cdot n_2 \cdot \frac{\log_2(m_x + 1)}{m_x}$$

where m_x is such that $x \in_{m_x} M$, n_1 is such that $x \in_{n_1} t_1$ and n_2 is such that $x \in_{n_2} t_2$.

Finally, we applied the previous variations with two different approaches to compute the values of $\sigma_t(x)$. Instead of using the probability explained in Section 4, we could use the number of times an input/output pair appears in the test suite:

$$\sigma_{\xi_A}(x) = \frac{1}{\#\text{test suite } \mathcal{I}/\mathcal{O} \text{ pairs with label } x}$$

The results of experiments, using Set3 (50 states and alphabets of size 25), are given in Table 4. These show that from the seven possible combinations, five are more or less equally *good*, and the other two are clearly worse. Therefore, we decided to retain our approach, since it appears to keep a good balance between intuition and being faithful to the original Information Theory formulae.

7.2. Variables used in the mutual information formula

Another important choice was the decision to not use true random variables and corresponding probability distributions in the Mutual Information formula. We now describe some alternatives considered. The alternatives that we explored used combinations of the following mechanisms for generating the random variables used in the definition of BMI.

1. Whether normalisation is used;
2. Whether one takes into account the number of times a pair appears in a test;
3. How to consider the case where an input/output pair appears in both tests (i.e. how one defines the "joint probability", corresponding to $\sigma_{\xi_{t_1,t_2}}(x_1, x_2)$, for two tests t_1, t_2 when considering the same pair ($x_1 = x_2$)).

Regarding the first point, normalisation would lead to a probability distribution (i.e. with values that sum to 1). Normalisation can be achieved by taking the sum of the values for the input/output pairs of the test and then dividing the value given for each input/output pair by this factor. This way, the sum of the probabilities of all the input/output pairs of the test is equal to 1.

Regarding the second mechanism, it would have been possible to use the *number of times* each input/output pair appears within a single test when defining the corresponding probability distribution. The resultant probabilities depend both on the test and on the `FSM`. The downside of this mechanism was that we lose the intuition that we previously followed, which is that the weight of each input/output pair in a test should be the probability of this pair corresponding to a particular transition of the specification (see paragraph after Example 1). Despite this, we evaluated alternatives that take into account the number of times an input/output pair appears in a test.

Finally, we considered the approach taken to define the joint probability distribution for tests t_1 and t_2. We explored an alternative approach in which we start by giving a random variable and its probability distribution for t_1 and t_2. Then, we compute the joint probability of both tests with the uncorrelated input/output pairs of each test (the case where the input/output pairs are different) and add all these values. This is straightforward because the joint probability of uncorrelated input/output pairs is simply the product of the probabilities of each input/output pair. As the sum of all the values of the joint probability should sum up to 1, we know the amount of probability corresponding to the correlated input/output pairs (we will call this P). Then, we defined s to be the product of the probabilities of each input/output pair modified by a factor and we define the joint probability of the correlated input/output pairs as s divided by the sum of all the s's and multiplied by P. This gives us a joint probability for each pair of correlated input/output pairs and we could call this "a joint probability of correlated input/output pairs". It is important to note that this joint probability is different from the product of the probabilities of each input/output pair; otherwise, in the mutual information formula we would get $\log_2(1)$ and since this is equal to 0, we would have mutual information of 0.

We tried several different combinations of these mechanisms. These alternatives are displayed in Table 5. The column "dist" corresponds to the probability distribution formula, while "joint" corresponds to the joint distribution formula for the correlated input/output pairs (for the uncorrelated input/output pairs, the joint distribution is the product of the individual distributions). In the table we assume $x_1 \in_{n_1} t_1$, $x_2 \in_{n_2} t_2$, $x_1 \in_m M$, $x_2 \in_m M$ and $P = 1 - S_1$.

As can be seen in Table 5, all the alternative formulations considered were outperformed by the proposed approach. This is despite some of the alternatives being notably more involved than the proposed approach. All of the alternatives achieve a mean score between 50% and 65%, with only one distribution getting more than 60%. In con-

Test	# **0.4**%	# **0.5**%	# **0.6**%	# **0.7**%	# **0.8**%	**min value**	**max value**	% **success (mean)**
MI based on spec	4	33	13	0	0	0.459184	0.680851	56.9662%
MI based on test suite	0	0	3	41	6	0.666667	0.818182	75.0757%
BMI based on spec	**0**	**0**	**5**	**36**	**9**	**0.673469**	**0.838384**	**75.3883**%
BMI based on test suite	0	0	5	32	13	0.666667	0.848485	76.4054%
BMI_2 based on spec	0	0	14	31	5	0.66	0.816327	74.2696%
BMI_2 based on test suite	0	0	1	43	6	0.670103	0.848485	75.1613%
BMI_3	3	22	24	1	0	0.42268	0.7	58.9412%

Table 4: Comparing different alternative approaches.

#	**dist**	**joint**	$\mathbf{s_1}$	$\mathbf{s_2}$	$\mathbf{S_1}$	$\mathbf{S_2}$	**success**
1	$\frac{1}{m \cdot s}$	$\frac{n_1}{m \cdot s_1} \cdot \frac{n_2}{m \cdot s_2} \cdot \frac{P}{S2}$	$\sum_{\substack{x_1 \in t_1 \\ x_1 \in M}} \frac{1}{m_1}$	$\sum_{\substack{x_2 \in t_2 \\ x_2 \in M}} \frac{1}{m_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 \neq x_2}} \frac{1}{m_1 \cdot s_1} \cdot \frac{1}{m_2 \cdot s_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 = x_2}} \frac{n_1}{m_1 \cdot s_1} \cdot \frac{n_2}{m_2 \cdot s_2}$	64%
2	$\frac{n}{s}$	$\frac{min(n_1, n_2)}{S_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_1 \in M}} n_1$	$\sum_{\substack{x_2 \in t_2 \\ x_2 \in M}} n_2$	0	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 = x_2}} min(n_1, n_2)$	55%
3	$\frac{n}{s}$	$\frac{n_1}{s_1} \cdot \frac{n_2}{s_2} \cdot \frac{1}{m_1} \cdot \frac{P}{S2}$	$\sum_{\substack{x_1 \in t_1 \\ x_1 \in_{m_1} M}} n_1$	$\sum_{\substack{x_2 \in t_2 \\ x_2 \in_{m_2} M}} n_2$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 \neq x_2}} \frac{n_1}{s_1} \cdot \frac{n_2}{s_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 = x_2}} \frac{n_1}{s_1} \cdot \frac{n_2}{s_2} \cdot \frac{1}{m_1}$	54%
4	$\frac{n}{s}$	$\frac{n_1}{s_1} \cdot \frac{n_2}{s_2} \cdot m_1 \cdot \frac{P}{S2}$	$\sum_{\substack{x_1 \in t_1 \\ x_1 \in_{m_1} M}} n_1$	$\sum_{\substack{x_2 \in t_2 \\ x_2 \in_{m_2} M}} n_2$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 \neq x_2}} \frac{n_1}{s_1} \cdot \frac{n_2}{s_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 = x_2}} \frac{n_1}{s_1} \cdot \frac{n_2}{s_2} \cdot m_1$	58%
5	$\frac{1}{m \cdot s}$	$\frac{1}{m_1 \cdot s_1} \cdot \frac{1}{m_2 \cdot s_2} \cdot \frac{1}{m_1} \cdot \frac{P}{S2}$	$\sum_{\substack{x_1 \in t_1 \\ x_1 \in_{m_1} M}} \frac{1}{m_1}$	$\sum_{\substack{x_2 \in t_2 \\ x_2 \in_{m_2} M}} \frac{1}{m_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 \neq x_2}} \frac{1}{m_1 \cdot s_1} \cdot \frac{1}{m_2 \cdot s_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 = x_2}} \frac{1}{m_1 \cdot s_1} \cdot \frac{1}{m_2 \cdot s_2} \cdot \frac{1}{m_1}$	57%
6	$\frac{1}{m \cdot s}$	$\frac{1}{m_1 \cdot s_1} \cdot \frac{1}{m_2 \cdot s_2} \cdot m_1 \cdot \frac{P}{S2}$	$\sum_{\substack{x_1 \in t_1 \\ x_1 \in_{m_1} M}} \frac{1}{m_1}$	$\sum_{\substack{x_2 \in t_2 \\ x_2 \in_{m_2} M}} \frac{1}{m_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 \neq x_2}} \frac{1}{m_1 \cdot s_1} \cdot \frac{1}{m_2 \cdot s_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 = x_2}} \frac{1}{m_1 \cdot s_1} \cdot \frac{1}{m_2 \cdot s_2} \cdot m_1$	55%
7	$\frac{n}{m \cdot s}$	$\frac{n_1}{m_1 \cdot s_1} \cdot \frac{n_2}{m_2 \cdot s_2} \cdot \frac{1}{m_1} \cdot \frac{P}{S2}$	$\sum_{\substack{x_1 \in t_1 \\ x_1 \in_{m_1} M}} \frac{n_1}{m_1}$	$\sum_{\substack{x_2 \in t_2 \\ x_2 \in_{m_2} M}} \frac{n_2}{m_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 \neq x_2}} \frac{n_1}{m_1 \cdot s_1} \cdot \frac{n_2}{m_2 \cdot s_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 = x_2}} \frac{n_1}{m_1 \cdot s_1} \cdot \frac{n_2}{m_2 \cdot s_2} \cdot \frac{1}{m_1}$	56%
8	$\frac{n}{m \cdot s}$	$\frac{n_1}{m_1 \cdot s_1} \cdot \frac{n_2}{m_2 \cdot s_2} \cdot m_1 \cdot \frac{P}{S2}$	$\sum_{\substack{x_1 \in t_1 \\ x_1 \in_{m_1} M}} \frac{n_1}{m_1}$	$\sum_{\substack{x_2 \in t_2 \\ x_2 \in_{m_2} M}} \frac{n_2}{m_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 \neq x_2}} \frac{n_1}{m_1 \cdot s_1} \cdot \frac{n_2}{m_2 \cdot s_2}$	$\sum_{\substack{x_1 \in t_1 \\ x_2 \in t_2 \\ x_1 = x_2}} \frac{n_1}{m_1 \cdot s_1} \cdot \frac{n_2}{m_2 \cdot s_2} \cdot m_1$	55%

Table 5: Comparing different probability distributions.

trast, our proposed approach had a score of 75.0605%.

Then, with all this information, we are able to clearly state that the alternative a tester should use is the one presented in Section 4.

8. Conclusions and future work

The selection of a test suite can be a critical task because the time and resources devoted to testing are limited. In this paper we considered the problem of choosing between two alternative test suites. Solutions to this problem might be used to directly compare alternative test suites (e.g. for different sets of features). They might also be used to guide the generation of test suites in an iterative manner or to inform the choice of which subset of a regression test suite to use. We observed that diverse test suites have been found to be effective and proposed the use of a novel measure, BMI, based on Mutual Information to assess diversity.

Having developed BMI, and analysed a number of its properties, we reported on experiments that evaluated it. First, we randomly generated pairs of test suites and used BMI to order the test suites in each pair. We then determined how many mutants of the `FSM` specification were killed by each test suite. In these experiments we found that test suites with lower BMI tend to kill more mutants. This provides evidence that BMI can be used as the basis for choosing between test suites. There was also a (negative) correlation between the fault detecting ability of a test suite (i.e. the number of mutants that it killed) and the BMI of the test suite. Interestingly, we also found that BMI outperformed the previous information theor-

etic measure, Test Set Diameter (TSDm), when selecting a test suite from a set of two randomly generated test suites.

The positive results, when comparing BMI with TSDm, suggest that the use of diversity is improved if we introduce knowledge about the specification; previous work using diversity has concerned white-box testing and assumed that a specification is not available. The results suggest that measures of test suite diversity can be improved if one has knowledge about the *rarity* of events. This knowledge can be extracted from the specification of the developed system. Naturally, if additional information is available then it should be possible to improve on measures. As a result, we found that transition coverage was (slightly) more effective than BMI, although the computation of transition coverage took much more time than the computation of BMI. Therefore, there is potentially a trade-off between effectiveness and time taken. Importantly, however, BMI can be used whenever we have information about input/output pair frequency and is therefore more widely applicable than transition coverage. In fact, there is potential to use BMI even if there is no specification, since it should be possible to estimate input/output pair frequency using sampling.

The results presented in this paper have some clear practical ramifications. First, testers can directly use the novel BMI measure if they have a number of test suites; they can compute the BMI of each test suite and take the one (or ones) with lower BMI. Testers can also use BMI to drive test subset selection; they can select the subset with lower BMI. In addition, if there is a limited budget for regression testing then the tester faces the problem of choosing a subset of the regression test suite; the test subset selection problem has been addressed based on white-box coverage information (see, for example, [3, 4, 5]) but, where this is not available, it is possible to instead use BMI. BMI can be used by testers when they have limited information about the specification. In particular, it can be used instead of methods based on coverage when we do not have a complete specification of the system but we have the frequency of each input/output pair.

An intuition that explains why BMI is better than a true Information Theory based measure can be the following one: BMI gives a proportional value to each input/output pair independently of the rest of the test. That is, we are giving the same weight to the same input/output pair independently of the test length. However, when using Information Theory based measures, we require a probability distribution over the input/output pairs of the test. Therefore, the weights of an input/output pair will be different in two different tests. This produces undesirable effects like the decrease of the weight of an input/output pair due to it being in a longer test than if it were in a shorter test. This can lead to situations where, for example, a test suite with a longer test with many repeated input/output pairs could be preferred to a test suite with many shorter tests with only one repeated input/output pair between all of them.

There are several possible lines of future work. First, it would be interesting to explore the use of BMI in the task of generating new test suites from scratch. Second, we would like to perform additional experiments to compare BMI and ITSDm. Third, there is potential to apply BMI in more complex scenarios, with one such scenario being when the specification is an Extended Finite State Machine (`EFSM`). Note that an `EFSM` can be mapped to an `FSM` through expanding out the data, possibly after applying an abstraction. Thus, `EFSM` faults that lead to incorrect variable values map nicely to the type of mutation used in the experiments, in which only the final state of a transition is changed. Another interesting scenario is given when we consider distributed systems, possibly with asynchronous communications, whose specifications are represented as a variant of an `FSM` [65, 66]. In order to confront these more complicated formalisms, we can use current work that make it possible to apply a systematic approach to the generation of mutants [67, 68]. In principle, it should also be possible to apply BMI even when there is no specification, since an initial random testing phase could be used to produce estimates of input/output pair frequency. For this scenario, there is a need for experiments that explore the process of producing estimates of input/output pair frequency and also the impact of using estimates on the effectiveness of BMI. This line of work is particularly important because it should make it possible to apply BMI in a context in which we do not have access to a specification and so we cannot apply a method based on the coverage of the available test suites.

Acknowledgements

We would like to thank the anonymous reviewers for the careful reading of the paper and the many constructive comments, which have helped us to further strengthen the paper.

References

[1] P. Ammann, J. Offutt, Introduction to Software Testing, 2nd Edition, Cambridge University Press, 2017.

[2] G. J. Myers, C. Sandler, T. Badgett, The Art of Software Testing, 3rd Edition, John Wiley & Sons, 2011.

[3] G. Rothermel, M. J. Harrold, Analyzing regression test selection techniques, IEEE Transactions on Software Engineering 22 (8) (1996) 529–551.

[4] Z. Li, M. Harman, R. M. Hierons, Search algorithms for regression test case prioritization, IEEE Transactions on Software Engineering 33 (4) (2007) 225–237.

[5] A. Arrieta, J. A. Agirre, G. Sagardui, Seeding strategies for multi-objective test case selection: an application on simulation-based testing, in: 22nd Annual Conf. on Genetic and Evolutionary Computation, GECCO'20, ACM, 2020, pp. 1222–1231.

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

[7] K. Sabnani, A. Dahbura, A protocol test generation procedure, Computer Networks and ISDN Systems 15 (1988) 285–297.

[8] I. Pomeranz, S. M. Reddy, Test generation for multiple state-table faults in finite-state machines, IEEE Transactions on Computers 46 (7) (1997) 783–794.
[9] A. Benharref, R. Dssouli, M. Serhani, A. En-Nouaary, R. Glitho, New approach for EFSM-based passive testing of web services, in: Joint 19th IFIP TC6/WG6.1 Int. Conf. on Testing of Software and Communicating Systems, TestCom'07, and 7th Int. Workshop on Formal Approaches to Software Testing, FATES'07, LNCS 4581, Springer, 2007, pp. 13–27.
[10] M. Haydar, A. Petrenko, H. Sahraoui, Formal verification of web applications modeled by communicating automata, in: 24th IFIP WG 6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems, FORTE'04, LNCS 3235, Springer, 2004, pp. 115–132.
[11] W. Grieskamp, N. Kicillof, K. Stobie, V. Braberman, Model-based quality assurance of protocol documentation: tools and methodology, Software Testing, Verification and Reliability 21 (1) (2011) 55–71.
[12] W. Huang, J. Peleska, Complete model-based equivalence class testing for nondeterministic systems, Formal Aspects of Computing 29 (2) (2017) 335–364.
[13] F. Hübner, W. Huang, J. Peleska, Experimental evaluation of a novel equivalence class partition testing strategy, Software and Systems Modeling 18 (1) (2019) 423–443.
[14] M. Isberner, F. Howar, B. Steffen, The open-source learnlib, in: 27th Int. Conf. on Computer Aided Verification, CAV'15, LNCS 9206, Springer, 2015, pp. 487–495.
[15] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, M. Mohri, OpenFst: A general and efficient weighted finite-state transducer library, in: 9th Int. Conf. on Implementation and Application of Automata, CIAA'07, LNCS 4783, Vol. 4783, Springer, 2007, pp. 11–23.
[16] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal 27 (1948) 379–423, 623–656.
[17] R. Feldt, R. Torkar, T. Gorschek, W. Afzal, Searching for cognitively diverse tests: Towards universal test diversity metrics, in: 1st IEEE Int. Conf. on Software Testing Verification and Validation Workshops, IEEE Computer Society, 2008, pp. 178–186.
[18] E. G. Cartaxo, P. D. L. Machado, F. G. de Oliveira Neto, On the use of a similarity function for test case selection in the context of model-based testing, Software Testing, Verification and Reliability 21 (2) (2011) 75–100.
[19] H. Hemmati, A. Arcuri, L. Briand, Achieving scalable model-based testing through test case diversity, ACM Transactions on Software Engineering and Methodology 22 (1) (2013) 6:1–6:42.
[20] H. Hemmati, Z. Fang, M. V. Mantyla, Prioritizing manual test cases in traditional and rapid release environments, in: 8th IEEE Int. Conf. on Software Testing, Verification and Validation, ICST'15, IEEE Computer Society, 2015, pp. 1–10.
[21] R. Feldt, S. M. Poulding, D. Clark, S. Yoo, Test set diameter: Quantifying the diversity of sets of test cases, in: 9th IEEE Int. Conf. on Software Testing, Verification and Validation, ICST'16, IEEE Computer Society, 2016, pp. 223–233.
[22] ISO/IEC JTCI/SC21/WG7, ITU-T SG 10/Q.8, Information Retrieval, Transfer and Management for OSI; Framework: Formal Methods in Conformance Testing. Committee Draft CD 13245-1, ITU-T proposed recommendation Z.500. ISO – ITU-T (1996).
[23] J. Tretmans, Model based testing with labelled transition systems, in: Formal Methods and Testing, LNCS 4949, Springer, 2008, pp. 1–38.
[24] M. Li, P. M. B. Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition, Springer, 2019.
[25] R. Cilibrasi, P. M. B. Vitányi, Clustering by compression, IEEE Transactions on Information Theory 51 (4) (2005) 1523–1545.
[26] C. Henard, M. Papadakis, M. Harman, Y. Jia, Y. L. Traon, Comparing white-box and black-box test prioritization, in: 38th Int. Conf. on Software Engineering, ICSE'16, ACM Press, 2016, pp. 523–534.
[27] E. P. Moore, Gedanken experiments on sequential machines, in: C. Shannon, J. McCarthy (Eds.), Automata Studies, Princeton University Press, 1956.
[28] F. C. Hennie, Fault-detecting experiments for sequential circuits, in: 5th Annual Symposium on Switching Circuit Theory and Logical Design, IEEE Computer Society, 1964, pp. 95–110.
[29] T. S. Chow, Testing software design modeled by finite state machines, IEEE Transactions on Software Engineering 4 (1978) 178–187.
[30] M. P. Vasilevskii, Failure diagnosis of automata, Cybernetics 4 (1973) 653–665.
[31] R. M. Hierons, H. Ural, Optimizing the length of checking sequences, IEEE Transactions on Computers 55 (5) (2006) 618–629.
[32] F. Ipate, Bounded sequence testing from deterministic finite state machines, Theoretical Computer Science 411 (16-18) (2010) 1770–1784.
[33] A. Simão, A. Petrenko, N. Yevtushenko, On reducing test length for FSMs with extra states, Software Testing, Verification and Reliability 22 (6) (2012) 435–454.
[34] R. M. Hierons, Testing from partial finite state machines without harmonised traces, IEEE Transactions on Software Engineering 43 (11) (2017) 1033–1043.
[35] R. M. Hierons, FSM quasi-equivalence testing via reduction and observing absences, Science of Computer Programming 177 (2019) 1–18.
[36] A. Petrenko, N. Yevtushenko, Testing from partial deterministic FSM specifications, IEEE Transactions on Computers 54 (9) (2005) 1154–1165.
[37] R. M. Hierons, M. Núñez, Implementation relations and probabilistic schedulers in the distributed test architecture, Journal of Systems and Software 132 (2017) 319–335.
[38] R. M. Hierons, M. G. Merayo, M. Núñez, Bounded reordering in the distributed test architecture, IEEE Transactions on Reliability 67 (2) (2018) 522–537.
[39] I. Hwang, A. R. Cavalli, Testing a probabilistic FSM using interval estimation, Computer Networks 54 (7) (2010) 1108–1125.
[40] N. López, M. Núñez, I. Rodríguez, Specification, testing and implementation relations for symbolic-probabilistic systems, Theoretical Computer Science 353 (1–3) (2006) 228–248.
[41] R. M. Hierons, M. G. Merayo, M. Núñez, Testing from a stochastic timed system with a fault model, Journal of Logic and Algebraic Programming 78 (2) (2009) 98–115.
[42] A. V. Aho, A. T. Dahbura, D. Lee, M. Ü. Uyar, An optimization technique for protocol conformance test generation based on UIO sequences and Rural Chinese Postman Tours, IEEE Transactions on Communications 39 (11) (1991) 1604–1615.
[43] A. Y. Duale, M. Ü. Uyar, A method enabling feasible conformance test sequence generation for EFSM models, IEEE Transactions on Computers 53 (5) (2004) 614–627.
[44] K. Derderian, R. M. Hierons, M. Harman, Q. Guo, Generating feasible input sequences for extended finite state machines (EFSMs) using genetic algorithms, in: 7th Genetic and Evolutionary Computation Conference, GECCO'05, ACM Press, 2005, pp. 1081–1082.
[45] A. S. Kalaji, R. M. Hierons, S. Swift, Generating feasible transition paths for testing from an extended finite state machine (EFSM), in: 2nd Int. Conf. on Software Testing Verification and Validation, ICST'09, IEEE Computer Society, 2009, pp. 230–239.
[46] A. Petrenko, S. Boroday, R. Groz, Confirming configurations in EFSM testing, IEEE Transactions on Software Engineering 30 (1) (2004) 29–42.
[47] A. Turlea, F. Ipate, R. Lefticaru, A test suite generation approach based on EFSMs using a multi-objective genetic algorithm, in: 19th Int.Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC'17, IEEE Computer Society, 2017, pp. 153–160.
[48] K. Bogdanov, M. Holcombe, F. Ipate, L. Seed, S. Vanak, Testing methods for X-machines: a review, Formal Aspects of Computing 18 (2006) 3–30.
[49] K. Androutsopoulos, D. Clark, H. Dan, R. Hierons, M. Harman,

An analysis of the relationship between conditional entropy and failed error propagation in software testing, in: 36th Int. Conf. on Software Engineering, ICSE'14, ACM Press, 2014, pp. 573–583.
[50] J. K. Blundell, M. L. Hines, J. Stach, The measurement of software design quality, Annals of Software Engineering 4 (1–4) (1997) 235–255.
[51] D. Clark, R. Feldt, S. M. Poulding, S. Yoo, Information transformation: An underpinning theory for software engineering, in: 37th IEEE/ACM International Conference on Software Engineering, ICSE'15, 2015, pp. 599–602.
[52] D. Clark, R. M. Hierons, Squeeziness: An information theoretic measure for avoiding fault masking, Information Processing Letters 112 (8-9) (2012) 335–340.
[53] A. Ibias, R. M. Hierons, M. Núñez, Using Squeeziness to test component-based systems defined as Finite State Machines, Information & Software Technology 112 (2019) 132–147.
[54] A. V. Miranskyy, M. Davison, R. M. Reesor, S. S. Murtaza, Using entropy measures for comparison of software traces, Information Sciences 203 (2012) 59–72.
[55] K. R. Pattipati, M. G. Alexandridis, Application of heuristic search and information theory to sequential fault diagnosis, IEEE Transactions on Systems, Man, and Cybernetics 20 (4) (1990) 872–887.
[56] K. R. Pattipati, S. Deb, M. Dontamsetty, A. Maitra, START: System testability analysis and research tool, IEEE Aerospace and Electronic Systems Magazine 6 (1) (1991) 13–20.
[57] R. Sagarna, A. Arcuri, X. Yao, Estimation of distribution algorithms for testing object oriented software, in: 9th IEEE Congress on Evolutionary Computation, CEC'07, IEEE Computer Society, 2007, pp. 438–444.
[58] S. Yoo, M. Harman, D. Clark, Fault localization prioritization: Comparing information-theoretic and coverage-based approaches, ACM Transanctions on Software Enginnering and Methodology 22 (3) (2013) 19:1–19:29.
[59] A. González-Sanchez, É. Piel, H.-G. Groß, A. J. C. van Gemund, Prioritizing tests for software fault localization, in: 10th Int. Conf. on Quality Software, QSIC'10, IEEE Computer Society, 2010, pp. 42–51.
[60] N. Alshahwan, M. Harman, Coverage and fault detection of the output-uniqueness test selection criteria, in: 24th ACM SIGSOFT Int. Symposium on Software Testing and Analysis, ISSTA'14, ACM Press, 2014, pp. 181–192.
[61] T. M. Cover, J. A. Thomas, Elements of Information Theory, Wiley Interscience, 1991.
[62] C. Andrés, M. G. Merayo, M. Núñez, Supporting the extraction of timed properties for passive testing by using probabilistic user models, in: 9th Int. Conf. on Quality Software, QSIC'09, IEEE Computer Society, 2009, pp. 145–154.
[63] D. Neider, R. Smetsers, F. W. Vaandrager, H. Kuppens, Benchmarks for automata learning and conformance testing, in: T. Margaria, S. Graf, K. G. Larsen (Eds.), Models, Mindsets, Meta: The What, the How, and the Why Not? - Essays Dedicated to Bernhard Steffen on the Occasion of His 60th Birthday, Springer, 2019, pp. 390–416.
[64] M. R. Garey, D. S. Johnson, Computers and Intractability, W. H. Freeman and Company, 1979.
[65] M. G. Merayo, R. M. Hierons, M. Núñez, Passive testing with asynchronous communications and timestamps, Distributed Computing 31 (5) (2018) 327–342.
[66] M. G. Merayo, R. M. Hierons, M. Núñez, A tool supported methodology to passively test asynchronous systems with multiple users, Information & Software Technology 104 (2018) 162–178.
[67] P. Gómez-Abajo, E. Guerra, J. de Lara, M. G. Merayo, A tool for domain-independent model mutation, Science of Computer Programming 163 (2018) 85–92.
[68] P. Gómez-Abajo, E. Guerra, J. de Lara, M. G. Merayo, WodelTest: a model-based framework for language-independent mutation testing, Software and Systems Modeling (in press).

Appendix A. Appendix: Additional material

This appendix does not constitute an integral part. It includes the full results of all the experiments reported in this paper.

Run #	Percentage of success
1	0.5761467889908257
2	0.5682242990654206
3	0.5600739371534196
4	0.5918727915194346
5	0.5875
6	0.6102941176470589
7	0.5652173913043478
8	0.5804701627486437
9	0.5652985074626866
10	0.5958254269449715
11	0.5952813067150635
12	0.5882352941176471
13	0.5892547660311959
14	0.5863309352517986
15	0.5678571428571428
16	0.5948905109489051
17	0.569620253164557
18	0.563963963963964
19	0.6133333333333333
20	0.5893805309734513
21	0.6077348066298343
22	0.5871212121212122
23	0.6007604562737643
24	0.5868372943327239
25	0.5979202772963604
26	0.5769944341372912
27	0.5989010989010989
28	0.5362318840579711
29	0.6040145985401459
30	0.5471014492753623
31	0.5886939571150097
32	0.585278276481149
33	0.6046099290780141
34	0.6163194444444444
35	0.5594795539033457
36	0.5659050966608085
37	0.5981981981981982
38	0.5730129390018485
39	0.5801801801801801
40	0.5884543761638734
41	0.5748613678373382
42	0.5545774647887324
43	0.5893805309734513
44	0.5981818181818181
45	0.5822550831792976
46	0.5801801801801801
47	0.5640569395017794
48	0.5786713286713286
49	0.6106870229007634
50	0.6083032490974729

Table A.6: Percentages of success of our selected test suite in the experiment with real FSMs and $A = 1/5$.

Run #	Percentage of success
1	0.6290726817042607
2	0.5892857142857143
3	0.6421319796954315
4	0.6048192771084338
5	0.6456310679611651
6	0.6059113300492611
7	0.625
8	0.592964824120603
9	0.6132075471698113
10	0.6363636363636364
11	0.6122931442080378
12	0.6222222222222222
13	0.6144278606965174
14	0.6123456790123457
15	0.6327014218009479
16	0.6015037593984962
17	0.6374407582938388
18	0.5949367088607594
19	0.6186666666666667
20	0.6363636363636364
21	0.6155717761557178
22	0.6425
23	0.6289156626506024
24	0.6491228070175439
25	0.5924932975871313
26	0.6428571428571429
27	0.6109785202863962
28	0.5772946859903382
29	0.6169665809768637
30	0.5815602836879432
31	0.6388140161725068
32	0.6341463414634146
33	0.6486486486486487
34	0.6275
35	0.645933014354067
36	0.6265356265356266
37	0.5944584382871536
38	0.6268656716417911
39	0.5965770171149144
40	0.6216867469879518
41	0.6035353535353535
42	0.6191646191646192
43	0.5891089108910891
44	0.5758293838862559
45	0.628140703517588
46	0.6116504854368932
47	0.6494117647058824
48	0.556390977443609
49	0.596401028277635
50	0.6218274111675127

Table A.7: Percentages of success of our selected test suite in the experiment with real FSMs and $A = 2/5$.

Run #	Percentage of success
1	0.6346153846153846
2	0.6633663366336634
3	0.6482758620689655
4	0.6782334384858044
5	0.5973597359735974
6	0.6382252559726962
7	0.6054421768707483
8	0.6611295681063123
9	0.6331168831168831
10	0.6820987654320988
11	0.6254071661237784
12	0.6452702702702703
13	0.6528662420382165
14	0.6807817589576547
15	0.6666666666666666
16	0.6511627906976745
17	0.6524590163934426
18	0.6372881355932203
19	0.6910828025477707
20	0.6461038961038961
21	0.6918032786885245
22	0.625
23	0.6587837837837838
24	0.6
25	0.6221498371335505
26	0.7032258064516129
27	0.6710963455149501
28	0.6514084507042254
29	0.6478405315614618
30	0.6366666666666667
31	0.6577181208053692
32	0.6741935483870968
33	0.6806451612903226
34	0.6326530612244898
35	0.6697530864197531
36	0.6825396825396826
37	0.6317567567567568
38	0.6419354838709678
39	0.6489028213166145
40	0.6477987421383647
41	0.6390728476821192
42	0.6861538461538461
43	0.6515151515151515
44	0.6495176848874598
45	0.6513157894736842
46	0.6389776357827476
47	0.6457564575645757
48	0.6332288401253918
49	0.6261980830670927
50	0.6914498141263941

Table A.8: Percentages of success of our selected test suite in the experiment with real FSMs and $A = 3/5$.

Run #	Percentage of success
1	0.6359832635983264
2	0.622568093385214
3	0.6475409836065574
4	0.7021276595744681
5	0.6509803921568628
6	0.7120622568093385
7	0.636734693877551
8	0.61003861003861
9	0.6126126126126126
10	0.673469387755102
11	0.6204081632653061
12	0.6653061224489796
13	0.6282051282051282
14	0.664
15	0.6613545816733067
16	0.5933609958506224
17	0.6363636363636364
18	0.6428571428571429
19	0.6336206896551724
20	0.6170212765957447
21	0.6223175965665236
22	0.6386554621848739
23	0.6788617886178862
24	0.5882352941176471
25	0.648068669527897
26	0.6283185840707964
27	0.64
28	0.6529680365296804
29	0.6567796610169492
30	0.6491228070175439
31	0.6533333333333333
32	0.6074380165289256
33	0.6260504201680672
34	0.6582278481012658
35	0.6443514644351465
36	0.6317991631799164
37	0.6125
38	0.6313725490196078
39	0.7086614173228346
40	0.6440677966101694
41	0.6173913043478261
42	0.6592920353982301
43	0.6390041493775933
44	0.6942148760330579
45	0.6220472440944882
46	0.6473029045643154
47	0.5966386554621849
48	0.6091954022988506
49	0.6115702479338843
50	0.6111111111111112

Table A.9: Percentages of success of our selected test suite in the experiment with real FSMs and $A = 4/5$.

Run #	With repetition of tests	without repetition of tests
1	0.65	0.555556
2	0.59596	0.62
3	0.666667	0.535354
4	0.602041	0.62
5	0.71	0.642857
6	0.58	0.59596
7	0.66	0.612245
8	0.56	0.65
9	0.65	0.585859
10	0.656566	0.71
11	0.69697	0.520408
12	0.59	0.646465
13	0.63	0.656566
14	0.63	0.636364
15	0.602041	0.737374
16	0.58	0.71
17	0.663265	0.636364
18	0.606061	0.646465
19	0.616162	0.581633
20	0.65	0.608247
21	0.585859	0.686869
22	0.59	0.57
23	0.540816	0.666667
24	0.653061	0.61
25	0.565657	0.626263
26	0.602041	0.61
27	0.61	0.587629
28	0.670103	0.63
29	0.639175	0.618557
30	0.632653	0.636364
31	0.540816	0.59596
32	0.626263	0.57
33	0.618557	0.632653
34	0.68	0.585859
35	0.6	0.646465
36	0.606061	0.64
37	0.58	0.57
38	0.66	0.61
39	0.59	0.626263
40	0.69697	0.663265
41	0.656566	0.69697
42	0.545455	0.656566
43	0.59	0.63
44	0.714286	0.656566
45	0.6	0.540816
46	0.571429	0.642857
47	0.63	0.61
48	0.606061	0.61
49	0.61	0.628866
50	0.714286	0.68

Table A.10: Percentages of success of our selected test suite in the experiment with controlled `FSM`s with alphabet of size 5.

Run #	Pearson correlation	Spearman correlation
1	−0.378529	−0.447121
2	−0.407239	−0.308503
3	−0.305347	−0.299361
4	−0.338248	−0.257336
5	−0.342747	−0.374436
6	−0.634282	−0.541008
7	−0.731647	−0.731102
8	−0.273694	−0.312782
9	−0.247209	−0.220384
10	−0.395459	−0.409929
11	−0.259926	−0.298081
12	−0.170457	−0.0308619
13	−0.414295	−0.456907
14	−0.507343	−0.693233
15	−0.58775	−0.624765
16	−0.44744	−0.545865
17	0.123024	0.232103
18	−0.383787	−0.300752
19	−0.534142	−0.471783
20	0.00977185	−0.0647103
21	−0.505013	−0.486649
22	−0.354695	−0.484575
23	−0.492013	−0.408578
24	−0.357769	−0.355907
25	−0.460478	−0.391877
26	−0.584937	−0.660399
27	−0.425394	−0.415194
28	−0.224411	−0.227905
29	−0.495339	−0.596992
30	−0.674246	−0.603391
31	−0.0656146	0.00300865
32	−0.468773	−0.438511
33	−0.433851	−0.395637
34	−0.564793	−0.456735
35	−0.553955	−0.548872
36	−0.383386	−0.40271
37	−0.285205	−0.221302
38	0.110759	0.0721805
39	−0.689946	−0.61203
40	−0.664481	−0.54778
41	−0.333295	−0.275188
42	−0.328676	−0.34501
43	−0.158634	−0.202484
44	−0.0459975	0.0428894
45	−0.245797	−0.300113
46	−0.380434	−0.359398
47	−0.325745	−0.26968
48	−0.462855	−0.37594
49	−0.132061	−0.17833
50	−0.242898	−0.248966

Table A.11: Correlation between BMI and mutation score with alphabet size of 5.

Run #	Percentage of success	Elapsed Time
1	0.76	0.000842141
2	0.721649	0.000793936
3	0.85	0.000822785
4	0.767677	0.000821019
5	0.826531	0.000813211
6	0.680412	0.000838903
7	0.77	0.000841091
8	0.74	0.000789195
9	0.806122	0.000826917
10	0.707071	0.000835529
11	0.717172	0.000853626
12	0.74	0.00082674
13	0.8	0.000840832
14	0.787879	0.000840821
15	0.76	0.00086412
16	0.75	0.000841077
17	0.714286	0.000850253
18	0.69	0.00082661
19	0.747475	0.0008403
20	0.714286	0.000846815
21	0.707071	0.000871455
22	0.65	0.000806566
23	0.757576	0.000827078
24	0.7	0.000842363
25	0.77	0.000819572
26	0.76	0.000836346
27	0.693878	0.00083186
28	0.717172	0.00085203
29	0.74	0.000836536
30	0.76	0.00085497
31	0.767677	0.000844704
32	0.795918	0.000847846
33	0.767677	0.000852514
34	0.78	0.000832202
35	0.71	0.000828946
36	0.795918	0.000863082
37	0.767677	0.000832604
38	0.83	0.000833852
39	0.83	0.000831776
40	0.838384	0.000874901
41	0.636364	0.000850119
42	0.737374	0.000826633
43	0.83	0.000844154
44	0.767677	0.000823715
45	0.686869	0.000835953
46	0.747475	0.000843442
47	0.69	0.000844199
48	0.73	0.000821044
49	0.7	0.000852965
50	0.814433	0.000873662

Table A.12: Results of BMI computation time.

Run #	Pearson correlation	Spearman correlation
1	−0.571537	−0.426476
2	−0.517691	−0.501696
3	−0.692478	−0.634825
4	−0.539586	−0.556391
5	−0.705762	−0.602183
6	−0.671802	−0.555305
7	−0.704822	−0.77924
8	−0.589333	−0.631064
9	−0.56587	−0.519744
10	−0.642352	−0.643851
11	−0.828483	−0.817908
12	−0.614509	−0.62147
13	−0.749209	−0.749906
14	−0.649231	−0.659135
15	−0.583118	−0.360286
16	−0.620321	−0.729323
17	−0.783434	−0.809184
18	−0.761556	−0.733358
19	−0.810923	−0.838661
20	−0.531118	−0.562406
21	−0.347714	−0.341353
22	−0.561493	−0.454306
23	−0.558292	−0.62754
24	−0.662369	−0.72009
25	−0.804059	−0.774436
26	−0.723854	−0.748683
27	−0.798174	−0.7567
28	−0.624052	−0.647347
29	−0.605207	−0.543675
30	−0.525772	−0.561324
31	−0.73993	−0.827379
32	−0.520028	−0.697744
33	−0.541205	−0.496989
34	−0.643269	−0.58443
35	−0.789753	−0.864459
36	−0.843549	−0.767784
37	−0.787704	−0.774436
38	−0.706133	−0.678706
39	−0.666575	−0.715789
40	−0.596623	−0.585844
41	−0.721583	−0.762406
42	−0.725915	−0.708804
43	−0.528187	−0.541008
44	−0.596918	−0.491347
45	−0.694988	−0.527109
46	−0.583214	−0.487585
47	−0.730042	−0.712782
48	−0.60586	−0.64812
49	−0.744956	−0.613996
50	−0.402239	−0.340986

Table A.13: Correlation between BMI and mutation score with alphabet size of 25.

Run #	BMI Wins	ITSDm Wins	Draw Winning	Draw Losing
1	0.29	0.26	0.18	0.27
2	0.3	0.17	0.24	0.29
3	0.39	0.18	0.2	0.23
4	0.32	0.18	0.28	0.22
5	0.24	0.21	0.27	0.28
6	0.36	0.15	0.25	0.24
7	0.33	0.22	0.24	0.21
8	0.28	0.23	0.25	0.24
9	0.31	0.17	0.26	0.26
10	0.31	0.24	0.26	0.19
11	0.33	0.22	0.25	0.2
12	0.29	0.19	0.23	0.29
13	0.33	0.27	0.16	0.24
14	0.27	0.23	0.23	0.27
15	0.36	0.24	0.23	0.17
16	0.32	0.24	0.23	0.21
17	0.26	0.22	0.3	0.22
18	0.31	0.23	0.26	0.2
19	0.26	0.16	0.29	0.29
20	0.25	0.23	0.23	0.29
21	0.31	0.29	0.23	0.17
22	0.27	0.24	0.26	0.23
23	0.27	0.24	0.29	0.2
24	0.32	0.25	0.26	0.17
25	0.29	0.21	0.28	0.22
26	0.35	0.24	0.2	0.21
27	0.31	0.22	0.19	0.28
28	0.31	0.25	0.23	0.21
29	0.28	0.23	0.29	0.2
30	0.22	0.21	0.35	0.22
31	0.3	0.17	0.29	0.24
32	0.35	0.18	0.26	0.21
33	0.29	0.21	0.22	0.28
34	0.34	0.16	0.27	0.23
35	0.32	0.15	0.3	0.23
36	0.36	0.14	0.24	0.26
37	0.31	0.19	0.3	0.2
38	0.35	0.18	0.29	0.18
39	0.34	0.16	0.22	0.28
40	0.32	0.23	0.26	0.19
41	0.23	0.21	0.35	0.21
42	0.23	0.3	0.17	0.3
43	0.25	0.24	0.27	0.24
44	0.27	0.2	0.29	0.24
45	0.31	0.19	0.3	0.2
46	0.29	0.18	0.32	0.21
47	0.21	0.22	0.33	0.24
48	0.28	0.24	0.27	0.21
49	0.37	0.19	0.21	0.23
50	0.32	0.16	0.26	0.26

Table A.14: Percentages from the comparison of BMI with ITSDm (Set2).

Run #	BMI Wins	ITSDm Wins	Draw Winning	Draw Losing
1	0.38	0.11	0.38	0.13
2	0.43	0.15	0.31	0.11
3	0.38	0.12	0.38	0.12
4	0.36	0.1	0.4	0.14
5	0.41	0.06	0.39	0.14
6	0.33	0.2	0.34	0.13
7	0.5	0.1	0.29	0.11
8	0.39	0.14	0.35	0.12
9	0.39	0.07	0.37	0.17
10	0.36	0.12	0.38	0.14
11	0.42	0.13	0.32	0.13
12	0.41	0.05	0.4	0.14
13	0.44	0.1	0.33	0.13
14	0.37	0.1	0.38	0.15
15	0.38	0.07	0.37	0.18
16	0.36	0.13	0.37	0.14
17	0.51	0.09	0.31	0.09
18	0.43	0.09	0.38	0.1
19	0.36	0.16	0.37	0.11
20	0.35	0.1	0.4	0.15
21	0.4	0.1	0.35	0.15
22	0.44	0.13	0.32	0.11
23	0.41	0.06	0.41	0.12
24	0.44	0.11	0.29	0.16
25	0.39	0.1	0.38	0.13
26	0.4	0.11	0.34	0.15
27	0.45	0.07	0.31	0.17
28	0.34	0.14	0.41	0.11
29	0.45	0.14	0.31	0.1
30	0.41	0.06	0.36	0.17
31	0.38	0.12	0.36	0.14
32	0.42	0.12	0.32	0.14
33	0.38	0.07	0.42	0.13
34	0.4	0.1	0.32	0.18
35	0.43	0.09	0.28	0.2
36	0.4	0.13	0.35	0.12
37	0.26	0.12	0.47	0.15
38	0.43	0.09	0.34	0.14
39	0.42	0.15	0.34	0.09
40	0.5	0.1	0.28	0.12
41	0.41	0.15	0.34	0.1
42	0.44	0.08	0.4	0.08
43	0.42	0.06	0.4	0.12
44	0.31	0.09	0.45	0.15
45	0.46	0.09	0.31	0.14
46	0.42	0.17	0.31	0.1
47	0.43	0.1	0.32	0.15
48	0.31	0.13	0.42	0.14
49	0.37	0.14	0.37	0.12
50	0.32	0.13	0.39	0.16

Table A.15: Percentages from the comparison of BMI with ITSDm (Set3).

FSM #	Min.	Max.	Average
1	0.41	0.99	0.66156
2	1.0	1.0	1.0
3	0.28	0.97	0.59814
4	0.65	0.98	0.79892
5	0.58	0.95	0.76067
6	0.81	0.99	0.91935
7	0.91	1.0	0.94743
8	0.53	0.97	0.77722
9	0.49	0.98	0.74142
10	0.56	0.98	0.74353
11	0.95	1.0	0.97229
12	0.87	1.0	0.93984
13	0.58	0.98	0.74424
14	0.75	0.98	0.85946
15	0.77	1.0	0.87572
16	0.54	0.93	0.73653
17	0.0	0.79	0.42637
18	0.76	1.0	0.86361
19	0.51	1.0	0.72173
20	0.0	0.89	0.40575
21	0.34	1.0	0.63675
22	0.97	1.0	0.98332
23	0.95	1.0	0.97183
24	0.51	0.97	0.71475
25	0.19	0.9	0.53744
26	0.74	0.98	0.85649
27	0.47	0.95	0.69854
28	0.98	1.0	0.9874
29	0.46	0.95	0.69807
30	0.84	0.99	0.91555
31	0.76	1.0	0.87161
32	0.0	0.92	0.43817
33	0.17	0.94	0.51964
34	0.26	0.97	0.57787
35	0.0	0.98	0.41146
36	0.51	0.94	0.71073
37	0.53	0.99	0.7301
38	0.71	0.96	0.83734
39	0.62	0.94	0.77565
40	0.0	0.82	0.42639
41	0.94	1.0	0.96297
42	0.88	1.0	0.93942
43	0.0	0.92	0.44462
44	0.0	0.9	0.42206
45	0.0	0.88	0.43675
46	0.45	0.96	0.67961
47	0.59	0.95	0.78192
48	0.46	0.98	0.69809
49	0.47	0.96	0.71049
50	0.62	0.99	0.78636

Table A.16: Percentage of test suites that kill the mutants (Set2 Part I).

FSM #	Min.	Max.	Average
51	0.66	0.99	0.80505
52	0.4	0.97	0.65833
53	0.86	1.0	0.92803
54	0.49	0.91	0.71166
55	0.59	1.0	0.79225
56	0.2	0.97	0.5383
57	0.48	0.94	0.69313
58	0.65	1.0	0.80878
59	0.78	0.99	0.87713
60	0.64	0.96	0.79091
61	0.85	1.0	0.93616
62	0.42	0.94	0.68205
63	0.9	1.0	0.94805
64	0.64	0.98	0.80052
65	0.35	0.96	0.63074
66	0.08	0.94	0.48562
67	0.26	0.86	0.6002
68	0.47	0.95	0.71338
69	0.37	0.96	0.69181
70	0.34	0.93	0.61436
71	0.17	0.91	0.5281
72	0.62	0.94	0.77926
73	0.0	0.89	0.45093
74	0.92	1.0	0.95179
75	0.24	0.85	0.53192
76	0.79	0.98	0.87988
77	0.92	1.0	0.96011
78	0.78	1.0	0.8672
79	0.25	0.92	0.56085
80	0.92	1.0	0.95496
81	0.48	0.98	0.70715
82	0.57	0.98	0.76196
83	0.75	0.98	0.8651
84	0.45	0.99	0.70628
85	0.77	1.0	0.89163
86	0.76	0.98	0.86909
87	0.0	0.9	0.44222
88	0.58	0.95	0.75736
89	0.61	0.96	0.77888
90	0.32	0.91	0.59404
91	0.81	0.99	0.88447
92	0.41	0.95	0.66994
93	0.51	0.95	0.74146
94	0.15	0.91	0.52903
95	0.3	0.94	0.60817
96	0.63	0.95	0.79545
97	0.74	0.98	0.84585
98	0.65	0.95	0.79298
99	0.3	0.94	0.606
100	0.58	0.96	0.77294

Table A.17: Percentage of test suites that kill the mutants (Set2 Part II).

FSM #	Max.	Average
1	0.91	0.44829
2	0.9	0.44203
3	0.86	0.43297
4	0.87	0.44348
5	0.88	0.41344
6	0.95	0.43757
7	0.93	0.43475
8	0.89	0.40389
9	0.96	0.43645
10	0.84	0.42938
11	0.91	0.42769
12	0.94	0.45951
13	0.86	0.43165
14	0.83	0.42816
15	0.92	0.4193
16	0.83	0.44864
17	0.89	0.48418
18	0.88	0.43903
19	0.95	0.43351
20	0.92	0.4278
21	0.92	0.41964
22	0.9	0.44669
23	0.84	0.42021
24	0.83	0.43035
25	0.87	0.41209
26	0.91	0.42674
27	0.84	0.45367
28	0.96	0.40972
29	0.89	0.41803
30	0.96	0.43839
31	0.89	0.4641
32	0.97	0.46382
33	0.9	0.44115
34	0.91	0.44495
35	0.91	0.43584
36	0.8	0.4282
37	0.92	0.44832
38	0.9	0.44736
39	0.91	0.45373
40	0.88	0.40431
41	0.87	0.41552
42	0.95	0.43048
43	0.8	0.41188
44	0.88	0.47413
45	0.96	0.43364
46	0.91	0.41166
47	0.9	0.45991
48	0.94	0.45115
49	0.97	0.39983
50	0.92	0.45365

Table A.18: Percentage of test suites that kill the mutants (Set3 Part I). Minimum is always 0.

FSM #	Max.	Average
51	0.96	0.44439
52	0.84	0.39549
53	0.96	0.43365
54	0.97	0.42208
55	0.95	0.46957
56	0.96	0.4291
57	0.94	0.42368
58	0.9	0.41643
59	0.85	0.45532
60	0.88	0.44115
61	0.92	0.43872
62	0.95	0.44664
63	0.88	0.48217
64	0.87	0.45015
65	0.93	0.42311
66	0.88	0.44841
67	0.93	0.44696
68	0.91	0.42443
69	0.83	0.44163
70	0.82	0.41825
71	0.88	0.44828
72	0.9	0.43722
73	0.96	0.43312
74	0.85	0.4523
75	0.85	0.4469
76	0.84	0.41888
77	0.87	0.43075
78	0.93	0.42524
79	0.98	0.41455
80	0.92	0.40092
81	0.99	0.43639
82	0.9	0.44402
83	0.92	0.45994
84	0.89	0.44517
85	0.97	0.46387
86	0.87	0.41392
87	0.9	0.43428
88	0.99	0.4524
89	0.9	0.41782
90	0.87	0.4276
91	0.94	0.44664
92	0.87	0.42702
93	0.95	0.40269
94	0.84	0.42797
95	0.89	0.46605
96	0.91	0.48529
97	0.91	0.43881
98	0.92	0.43231
99	0.88	0.44156
100	0.86	0.43291

Table A.19: Percentage of test suites that kill the mutants (Set3 Part II). Minimum is always 0.

FSM #	State coverage	Transition coverage	PMATC
1	0.7452	0.390893	0.6567
2	0.7672	0.404909	0.6681
3	0.766	0.38388	0.7025
4	0.7568	0.399706	0.6795
5	0.7912	0.409429	0.716499
6	0.7974	0.412286	0.721499
7	0.761	0.408863	0.682801
8	0.7236	0.402134	0.6595
9	0.7736	0.408647	0.6947
10	0.7458	0.377457	0.653
11	0.788	0.420706	0.7152
12	0.763	0.405353	0.6891
13	0.7338	0.382743	0.6698
14	0.803	0.404309	0.7318
15	0.7208	0.403291	0.6372
16	0.79	0.409829	0.717199
17	0.7866	0.399945	0.727899
18	0.7304	0.421753	0.649499
19	0.724	0.398503	0.665501
20	0.8022	0.375026	0.7238
21	0.7558	0.405522	0.661
22	0.7714	0.390471	0.6638
23	0.7632	0.385706	0.6827
24	0.7878	0.4148	0.725899
25	0.7706	0.410542	0.6815
26	0.814	0.388229	0.745401
27	0.7474	0.382102	0.6725
28	0.7958	0.402944	0.725299
29	0.775	0.409415	0.700101
30	0.794	0.399011	0.726199
31	0.8078	0.427134	0.7005
32	0.7938	0.427824	0.7273
33	0.7868	0.392528	0.698699
34	0.7786	0.403121	0.6974
35	0.7692	0.401861	0.691201
36	0.7616	0.390387	0.7066
37	0.7714	0.401221	0.690101
38	0.7874	0.431205	0.7158
39	0.7882	0.394365	0.7138
40	0.7884	0.406514	0.7114
41	0.7444	0.378352	0.688599
42	0.763	0.401198	0.670001
43	0.7324	0.403187	0.6451
44	0.7898	0.396448	0.7255
45	0.796	0.399034	0.7023
46	0.8	0.394837	0.7265
47	0.7954	0.415176	0.7058
48	0.762	0.406871	0.6632
49	0.7852	0.4345	0.6952
50	0.7738	0.40681	0.6631

Table A.20: Test suite coverage evaluation results in percentages (Set2 Part I). We include Percentages with respect to Maximal Achievable Transition Coverage.

FSM #	State coverage	Transition coverage	PMATC
51	0.7616	0.400526	0.684901
52	0.7772	0.418788	0.691
53	0.7418	0.388171	0.679299
54	0.805	0.418439	0.7239
55	0.7834	0.392286	0.686499
56	0.7932	0.389945	0.7136
57	0.7956	0.403218	0.701599
58	0.7328	0.364674	0.671
59	0.7758	0.404768	0.696201
60	0.7766	0.396033	0.7287
61	0.7652	0.396171	0.693299
62	0.7596	0.403095	0.6772
63	0.7674	0.398343	0.6732
64	0.7868	0.409882	0.6927
65	0.7922	0.408363	0.698301
66	0.7748	0.411797	0.687701
67	0.8096	0.436348	0.728701
68	0.8042	0.431151	0.711399
69	0.7686	0.388187	0.706499
70	0.7644	0.388022	0.706199
71	0.8192	0.395769	0.720299
72	0.8152	0.406966	0.724399
73	0.81	0.425	0.714
74	0.7542	0.381768	0.691
75	0.7552	0.378737	0.7196
76	0.7622	0.417665	0.697501
77	0.7768	0.408855	0.6787
78	0.7262	0.386509	0.6532
79	0.7854	0.388989	0.7313
80	0.7648	0.423418	0.669001
81	0.7426	0.400248	0.6444
82	0.7654	0.415305	0.6811
83	0.7908	0.430307	0.7014
84	0.7984	0.435901	0.7018
85	0.7668	0.400059	0.6761
86	0.7816	0.40092	0.6976
87	0.8054	0.437143	0.7038
88	0.7792	0.406994	0.7041
89	0.7882	0.411345	0.703401
90	0.756	0.387598	0.6938
91	0.7746	0.390945	0.7037
92	0.765	0.407651	0.6767
93	0.8102	0.434573	0.7127
94	0.7446	0.415938	0.6655
95	0.7514	0.419063	0.6705
96	0.7968	0.411136	0.7236
97	0.7452	0.376667	0.6893
98	0.7638	0.405907	0.694101
99	0.7834	0.413392	0.706901
100	0.7802	0.405989	0.698301

Table A.21: Test suite coverage evaluation results in percentages (Set2 Part II). We include Percentages with respect to Maximal Achievable Transition Coverage.

FSM #	State coverage	Transition coverage	PMATC
1	0.7778	0.415671	0.6817
2	0.7662	0.405353	0.6891
3	0.7938	0.413977	0.707901
4	0.7974	0.424458	0.7046
5	0.7744	0.399435	0.707
6	0.8006	0.415407	0.714501
7	0.788	0.405029	0.7007
8	0.7914	0.404413	0.7239
9	0.7674	0.423727	0.6822
10	0.7926	0.403295	0.7098
11	0.7738	0.393779	0.677301
12	0.8128	0.411916	0.687901
13	0.78	0.404128	0.695101
14	0.7956	0.422083	0.7091
15	0.7538	0.397337	0.6715
16	0.7974	0.393757	0.7127
17	0.8096	0.425154	0.693
18	0.7804	0.411183	0.6949
19	0.7802	0.403489	0.694001
20	0.7968	0.405114	0.713
21	0.7718	0.418235	0.711
22	0.7718	0.432911	0.684
23	0.7926	0.408864	0.7196
24	0.8026	0.424551	0.709001
25	0.7774	0.409538	0.7085
26	0.7462	0.391737	0.654201
27	0.7988	0.419461	0.700501
28	0.773	0.3925	0.7222
29	0.7804	0.399385	0.7149
30	0.7782	0.388506	0.675999
31	0.8176	0.419581	0.700701
32	0.7576	0.424494	0.6707
33	0.797	0.413684	0.707401
34	0.7876	0.411579	0.7038
35	0.799	0.399375	0.7029
36	0.801	0.405967	0.7348
37	0.8006	0.439062	0.7025
38	0.7938	0.408855	0.6787
39	0.799	0.40578	0.702
40	0.7884	0.387263	0.7358
41	0.7696	0.393559	0.6966
42	0.7598	0.390284	0.6869
43	0.7528	0.406023	0.694301
44	0.7468	0.422581	0.655
45	0.7734	0.41	0.6724
46	0.7776	0.392198	0.713799
47	0.815	0.421006	0.7115
48	0.7652	0.408963	0.6707
49	0.781	0.396497	0.7018
50	0.7974	0.42018	0.701701

Table A.22: Test suite coverage evaluation results in percentages (Set3 Part I). We include Percentages with respect to Maximal Achievable Transition Coverage.

FSM #	State coverage	Transition coverage	PMATC
51	0.77	0.410424	0.6772
52	0.7694	0.403932	0.718999
53	0.7644	0.429814	0.692
54	0.7758	0.40386	0.690601
55	0.7988	0.438853	0.689
56	0.7748	0.388258	0.691099
57	0.8038	0.408324	0.7309
58	0.7874	0.395281	0.703599
59	0.782	0.436346	0.6807
60	0.7952	0.405402	0.705399
61	0.8026	0.439752	0.708
62	0.7938	0.427048	0.7089
63	0.7782	0.431503	0.6602
64	0.8098	0.435151	0.717999
65	0.7708	0.401395	0.690401
66	0.8094	0.447	0.7152
67	0.7968	0.418537	0.6864
68	0.7756	0.399535	0.687201
69	0.7998	0.42	0.714
70	0.7984	0.411573	0.732599
71	0.8216	0.424269	0.725501
72	0.8052	0.409714	0.716999
73	0.7742	0.422638	0.6889
74	0.8286	0.441212	0.727999
75	0.799	0.416667	0.7125
76	0.7854	0.382989	0.7047
77	0.7798	0.413721	0.711601
78	0.7704	0.409177	0.6956
79	0.7348	0.397831	0.6604
80	0.7762	0.398962	0.7301
81	0.7524	0.412468	0.6517
82	0.7512	0.42075	0.6732
83	0.7704	0.414479	0.6756
84	0.8016	0.412486	0.7136
85	0.7934	0.420311	0.6767
86	0.7848	0.401808	0.7112
87	0.8016	0.401932	0.7074
88	0.7922	0.40503	0.6845
89	0.7618	0.405497	0.693401
90	0.807	0.390909	0.731001
91	0.733	0.397469	0.6439
92	0.7916	0.428036	0.7191
93	0.7738	0.39676	0.7102
94	0.7826	0.412126	0.7171
95	0.8102	0.415882	0.707
96	0.807	0.452467	0.6787
97	0.79	0.425864	0.6899
98	0.794	0.395304	0.7155
99	0.8116	0.427679	0.7185
100	0.7898	0.415385	0.702

Table A.23: Test suite coverage evaluation results in percentages (Set3 Part II). We include Percentages with respect to Maximal Achievable Transition Coverage.