Using Squeeziness to test component-based systems defined as Finite State Machines

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

[a] *Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Madrid, 28040, Spain*
[b] *Department of Computer Science, The University of Sheffield, Sheffield, S1 4DP, United Kingdom*

Abstract

Context: Testing is the main validation technique used to increase the reliability of software systems. The effectiveness of testing can be strongly reduced by *Failed Error Propagation*. This situation happens when the System Under Test executes a faulty statement, the state of the system is affected by this fault, but the expected output is observed. Squeeziness is an information theoretic measure designed to quantify the likelihood of Failed Error Propagation and previous work has shown that Squeeziness correlates strongly with Failed Error Propagation in white-box scenarios. Despite its usefulness, this measure, in its current formulation, cannot be used in a black-box scenario where we do not have access to the source code of the components.
Objective: The main goal of this paper is to adapt Squeeziness to a black-box scenario and evaluate whether it can be used to estimate the likelihood that a component of a software system introduces Failed Error Propagation.
Method: First, we defined our black-box scenario. Specifically, we considered the Failed Error Propagation that a component introduces when it receives its input from another component. We were interested in this since such fault masking makes it more difficult to find faults in the *previous* component when testing. Second, we defined our notion of Squeeziness in this framework. Finally, we carried out experiments in order to evaluate our measure.
Results: Our experiments showed a strong correlation between the likelihood of Failed Error Propagation and Squeeziness.
Conclusion: We can conclude that our new notion of Squeeziness can be used as a measure that estimates the probability of Failed Error Propagation being introduced by a component. As a result, it has the potential to be used as a measure of testability, allowing testers to assess how easy it is to test either the whole system or a single component. We considered a simple model (Finite State Machines) but the notions and results can be extended/adapted to deal with more complex state-based models, in particular, those containing data.

1. Introduction

Software testing [3, 35] is the main validation technique used to increase the reliability of complex software systems. Software testing has traditionally been considered to be an *informal* technique [18]. However, it is now known that testing activities can have a formal basis. Formal testing is an active research area [7, 10, 25] and the existence of several tools that support formal testing has led to the recognition that the combination of formal methods and testing facilitates test automation [42].

Failed Error Propagation (FEP) is a situation in which a faulty statement in the *System Under Test (SUT)* is executed during testing, the fault *corrupts* the internal state of the SUT, but the expected output is observed. Naturally, in order for a statement to be a fault there must be at least one input under which FEP does not occur. FEP is a form of fault masking and can reduce the effectiveness of testing: we might fail to find a fault despite executing the faulty statement in testing. Empirical studies have shown that many systems suffer from FEP [4, 33]. For example, Masri et al. [33] found that in 13% of the programs that they examined, a total of 60% or more of the tests suffered from FEP.

Recent work introduced the notion of Squeeziness [4, 13] to capture FEP, with Squeeziness being a measure of the information (entropy) lost by a channel (the SUT) that takes input and returns output. The essential idea is that if the SUT maps two or more inputs to the same output then this channel (the SUT) can lead to a loss of information: if we know the program output then we may not know the program input that caused this (this is the loss of information). The motivation for looking at Squeeziness was that FEP can be caused by two program states, a *correct* program state and a *faulty* program state, being mapped to the same output, which is exactly this type of loss of information. In experiments, there was a rank correlation of close to 0.95 between measures of Squeeziness and the likelihood of FEP [4]. In addition, it has been found that the likelihood of FEP more strongly correlates with Squeeziness than with the Domain to Range

*Corresponding author.
Email addresses: aibias@ucm.es (Alfredo Ibias), r.hierons@sheffield.ac.uk (Robert M. Hierons), manuelnu@ucm.es (Manuel Núñez)

Ratio [13].

The goal of this paper is to adapt the notion of Squeeziness to a black box testing scenario in which a software system is composed of components and we have models of these components. We consider the situation in which we have a component C with model M and this component receives a sequence of inputs from another component C_P.[1] The sequence received by C is an input sequence for C and an output sequence for C_P and might result, for example, from communications through an internal network, a sequence of method/function calls, or shared storage/memory. We assume that these values (sent by C_P to C) are not directly observed by the tester. Further, C produces a sequence of outputs that are either observed during testing or are received by another component. A graphical representation of this type of systems can be found in Figure 1. It is entirely possible that C_P produces an unexpected sequence but component C maps the expected and unexpected sequences to the same output sequence. If this occurs, C introduces a form of FEP that makes it more difficult to find faults in C_P.

In this paper we concentrate on a particular type of model, the Finite State Machine (`FSM`). An `FSM` has a finite set of states and transitions between the states, with each transition having a label: an input/output pair. The behaviour represented by an `FSM` is the set of input/output sequences that label paths from the initial state of the `FSM`. One of the main reasons for our interest in the `FSM` formalism is that it has been widely used as the basis for model-based testing (MBT). This line of work (MBT from `FSM` specifications) started in the 1950s with Moore's seminal paper [34], with Hennie [23] later (in the 1960s) introducing the first `FSM` based test generation algorithm. Many `FSM` based test generation algorithms have since been devised (see, for example [12, 31, 39, 30]). The initial work was largely in the context of testing hardware, since processors are typically specified as FSMs. Since the 1980s FSMs have also been used in the testing of communications protocols. More generally, FSMs are used as the basis for testing a wide variety of systems including embedded systems [9, 36] and parts of operating systems [19, 20]. Although `FSM`s do not directly model data, an `FSM` is typically extracted from a model (in a richer language) by either applying an abstraction or expanding out the data (possibly after applying an abstraction).

In this paper we assume that we have an `FSM`specification of the component C being analysed. In this setting, component C can introduce FEP, and so potentially make testing more difficult, if there is a case where a component C_P should send sequence α to C, instead C_P sends $\alpha' \neq \alpha$ to C, and C produces the same output sequence β in response to α and α'. Since an `FSM` receives a sequence of inputs and produces a sequence of outputs, in our setting, α and α' will be potential inputs of C and β will be a potential output of C. Naturally, since α and α' are sent to C, and as we already mentioned, they are not directly observed by the tester. This situation is illustrated in Figure 2. Assume that we want to implement the component C_p given in the upper part of Figure 2 and that this component will be paired with component C. In this setting, it will be difficult to unmask a faulty implementation of C_p, such as the one shown in the lower part of Figure 2, because C returns the same response, the sequence z_1z_1, to the sequences y_1y_1 (produced by a correct implementation of C_p receiving x_1x_1) and y_2y_2 (produced by a faulty implementation of C_p also receiving x_1x_1). Note, as we already said, that a tester will not be able to observe whether the sequence provided to C is y_1y_1 or y_2y_2.

Unfortunately, we cannot simply reuse the previous approach [13] and results; we are considering a different scenario and also a different source of FEP. The following are the key differences.

1. We are interested in a *different type of FEP*. Previous work looked at the FEP within a program: the potential for a program to mask faults within itself. In contrast, we are interested in the potential for one program (component) C to mask faults in another component C_P.
2. We are interested in *programs that are state-based*. In contrast, previous work looked at programs that retain no state information (after processing an input). As a result, the input domain of an SUT is the set of all possible input sequences. This complicates the underlying theory since:
   (a) The input domain is infinite (previous work assumed finite input domains [4]).
   (b) We cannot consider arbitrary probability distributions over the set of input sequences.

   Regarding the second point, we need to carefully consider how we can assign probabilities to input sequences since, for example, it may not make sense to have input sequences σ_1 and σ_2 such that σ_1 is a prefix of σ_2 and σ_2 is given a higher probability than σ_1.
3. Previous work considered the source code of a program and we instead base the analysis on *models*. This brings a number of benefits, including the potential to apply the analysis at an earlier stage.

There were several reasons for us reconsidering the previous decision to base the analysis on the source code. A first practical concern is that approaches that analyse the source code are less likely to scale to situations in which there are multiple components. There is also the issue that for state-based systems there is a need to reason about the change in state caused when we execute the SUT with an input: state-based models make this explicit. Moreover, the source code of a component might not be available (e.g. if the development of a component has been outsourced).

[1] C might actually receive input from multiple components; allowing this will not affect the underlying approach but complicates the exposition.

Figure 1: Representation of our testing scenario

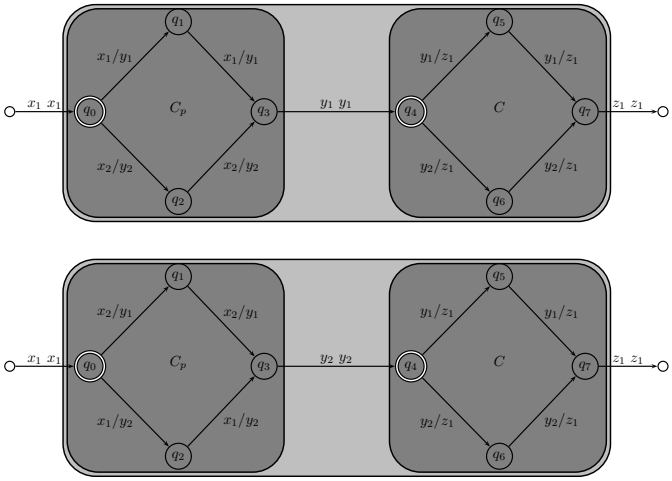


Figure 2: A case of fault masking

Finally, analysis based on models might be applied at an earlier stage of the development process. Note that detailed state-based models are used in a number of important application domains such as automotive and avionic systems; here the models are typically sufficiently detailed to be executable and also for code to be automatically generated from them[2]. As a result, we chose to consider the case where we have *models* of the components and wish to analyse these models. Note that, as explained above, this means that our scenario (and source of FEP) differs from the previous work, as does the entity being analysed.

Although the FSM formalism is relatively simple, we establish the basis of a framework to test in more complex black-box contexts because the basis of testing is similar: we apply a sequence of inputs and decide whether the observed sequence of outputs is consistent with the specification of the system [20]. Further, an FSM might represent the semantics of a model written in a more expressive language. In addition to extending the notion of Squeeziness to a black box scenario, we evaluated this through two types of experiments. The first approach used was to model the component C in terms of the sizes of the inverse images of the possible output sequences; this was used to compare the probability of FEP with our proposed metrics and also the Domain To Range Ratio of C. A weakness of this approach, however, is that we cannot guarantee that the simulations correspond to potential FSM models. As a result, we also carried out experiments using randomly generated FSMs.

The overall results were encouraging, with there being a high correlation between our proposed measures and the probability of FEP. As a result, the proposed measures could act as testability measures for state-based testing and have the potential to help direct testing. There are two practical reasons for the interest in measures associated with FEP. First, there may be potential to generate test cases that achieve a given test purpose, such as testing a component C_P, and that have a low probability of FEP. Second, such measures might be used to estimate testability; we might expect it to be particularly difficult for testing to find a fault in a component C_P that sends its output to another component C that is likely to introduce FEP. Measures of testability might be used to direct additional testing towards difficult to test areas of a system.

As we have explained, there are several differences between the original scenario [13] and ours and these include the type of FEP considered and the entity being analysed. These differences introduced a number of technical challenges. First, we had to reshape the definition of Squeeziness because inputs and outputs have a different treatment in each scenario. In the previously considered white-box case, a program receives an input (a tuple of values) and returns an output (again, a tuple of values). Inputs and outputs were drawn from finite sets allowing, for example, the use of uniform distributions. In the scenario that we consider in this paper, an *input* is a sequence of input actions while an output is also a sequence, in this case of output actions. This leads to two issues, the first of which is that the 'input set' is infinite (it is the set of all input sequences), as is the 'output set'. The second issue is that, even if we bound sequences to make the sets finite, we cannot define a uniform distribution over the sets of inputs and outputs because, for example, a prefix of a sequence should have a higher probability than the whole sequence.

There is a significant body of work on FEP and fault masking for white-box testing [6, 33, 44, 46] and black-box testing [21, 37, 38, 45]. As mentioned, previous work has also defined Squeeziness in a white-box scenario [4, 13]. However, we are not aware of any work that uses an Information Theory foundation for addressing FEP in a black-box context. Naturally, there is work that looks at testing systems that are composed of components for which we have FSM models [5, 15, 41] but this previous work has considered rather different concerns. There has also been previous work that uses information theoretic measures in testing. For example, Information Theory has been used to devise new measures of *test diversity* [16, 17], with the potential to either direct test generation towards diverse test suites or facilitate the development of new test criteria. Another line of work, though not one that uses Information Theory, aims to find diverse tests where diversity refers to the test outputs [2]. It is interesting to note that recent work found that white-box and black-box notions of diversity were both effective when searching for a good order of the test cases in a given test suite (the test prioritisation problem) [22].

The rest of the paper is organised as follows. In Sec-

[2]There are a number of widely used tools such as STATEMATE and STATEFLOW that support this.

tion 2 we introduce concepts and terminology used in the rest of the paper. Section 3 develops our novel information theoretic measures for FSMs. Section 4 then describes the empirical evaluation carried out and the results of the different sets of experiments. Finally, in Section 5 we present our conclusions and some lines of future work.

2. Preliminaries

In this section we present the main definitions and concepts, regarding Finite State Machines (FSMs), that we use throughout this paper. The material presented in this section is based on classical work on testing from FSMs [32]. Most of the concepts are based on the original sources while some notation is adapted to facilitate the formulation of subsequent definitions.

Given a set A, we let A^* denote the set of finite sequences of elements of A; $\epsilon \in A^*$ denotes the empty sequence. We let A^+ denote the set of non-empty sequences of elements of A. A^k denotes the set of sequences with length $k \geq 1$. We let $|A|$ denote the cardinal of set A. Given a sequence $\sigma \in A^*$, we have that $|\sigma|$ 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 I be the set of input actions and O be the set of output actions. It is important to differentiate between input actions and *inputs* of the system. In our context an input of a system will be a non-empty sequence of input actions, that is, an element of I^+ (similarly for outputs and output actions).

A *Finite State Machine* is a (finite) labelled transition system in which transitions are labelled by an input/output pair. We use this formalism to define processes.

Definition 1 *We say that $M = (Q, q_{in}, I, O, T)$ is a* Finite State Machine *(FSM), where Q is a finite set of states, $q_{in} \in Q$ is the initial state, I is a finite set of input actions, O is a finite set of output actions, and $T \subseteq Q \times (I \times O) \times Q$ is the transition relation. A transition $(q, (i, o), q') \in T$, also denoted by $q \xrightarrow{i/o} q'$ or by $(q, i/o, q')$, means that from state q after receiving input i it is possible to move to state q' and produce output o.*

We say that M is deterministic *if for all $q \in Q$ and $i \in I$ there exists at most one pair $(q', o) \in Q \times O$ such that $(q, i/o, q') \in T$. In this paper we consider deterministic FSMs.*

We say that M is input-enabled *if for all $q \in Q$ and $i \in I$ there exists $(q', o) \in Q \times O$ such that $(q, i/o, q') \in T$.*

We let $\mathtt{FSM}(I, O)$ denote the set of FSMs with input set I and output set O.

A process can be identified with its initial state and we can define a process corresponding to a state q of M by making q the initial state. Thus, we use states and processes and their notation interchangeably. 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 a double circle to denote the initial state.

As usual, we assume that the System Under Test (SUT) is input-enabled: the SUT should be able to react, somehow, to any external stimulus. In particular, if the tester applies an input action at a certain stage, then the system should be able to provide a response (that is, an output action). Actually, if an input cannot be applied in a state of the SUT, then we can assume that there is a response to the input that reports that this input is blocked and so an FSM that is not input-enabled can be converted into one that is. In addition, it has been shown that the problem of testing from an FSM that is not input-enabled can be mapped to the problem of testing from an input-enabled FSM [24, 40]. As a result, the assumption that FSMs are input-enabled is not a significant restriction. However, we do not force specifications to be input-enabled. In particular, all the definitions and results concerning Squeeziness will not assume input-enableness. As stated in the previous definition, we consider the case where both specifications and SUTs are deterministic. This is similar to the previously explored white-box scenario that assumed that programs are deterministic.

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 the behaviours of specifications and SUTs and the main notion to define such behaviours is given by the concept of *trace*.

Definition 2 *Let $M = (Q, q_{in}, I, O, T)$ be an FSM. We use the following notation.*

1. *Let $\sigma = (i_1, o_1) \dots (i_k, o_k) \in (I \times O)^*$ be a sequence of input/output actions and 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 $q_0 = q$. We denote this by either $q \stackrel{\sigma}{\Longrightarrow} q_k$ or $q \stackrel{\sigma}{\Longrightarrow}$. If $q = q_{in}$ then we say that σ is a* trace *of M. We denote by $\mathtt{traces}(M)$ the set of traces of M. Note that for every state q we have that $q \stackrel{\epsilon}{\Longrightarrow} q$ holds. Therefore, $\epsilon \in \mathtt{traces}(M)$ for every FSM M.*
2. *Let $\alpha = i_1 \dots i_k \in I^*$ be a sequence of input actions and q be a state. We define $\mathtt{out}_M(q, \alpha)$ as the set*

$$\{o_1 \dots o_k \in O^* | q \stackrel{(i_1,o_1)\dots(i_k,o_k)}{\Longrightarrow}\}$$

   Note that if M is deterministic then this set is either empty or a singleton. In the last case we will sometimes write $\mathtt{out}_M(q, \alpha) = o_1, \dots, o_k$.
3. *Let $q \in Q$ be a state. We define $\mathtt{dom}_M(q)$ as the set*

$$\{\alpha \in I^* | \mathtt{out}_M(q, \alpha) \neq \emptyset\}$$

If $q = q_{in}$ then we simply write $\mathtt{dom}_M$. Similarly, we define $\mathtt{image}_M(q)$ as the set

$$\{o_1 \dots o_k \in O^* | \exists i_1 \dots i_k \in I^* : q \xRightarrow{(i_1,o_1)\dots(i_k,o_k)}\}$$

If $q = q_{in}$ then we simply write $\mathtt{image}_M$. We denote by $\mathtt{dom}_{M,k}$ the set $\mathtt{dom}_M \cap I^k$. Similarly, We denote by $\mathtt{image}_{M,k}$ the set $\mathtt{image}_M \cap O^k$.

Note that if M is input-enabled then for all $k > 0$ we have that $\mathtt{dom}_{M,k} = I^k$ and, therefore, for all $\alpha \in I^k$ we have that $\mathtt{out}_M(q, \alpha) \neq \emptyset$.

3. Squeeziness for FSMs

In this section we show how the notion of Squeeziness can be adapted to the situation in which we would like to reason about the FEP introduced by a component C that has FSM specification M. As previously discussed, such FEP affects the testing of previous components (components that send output to C) since it might lead to a faulty sequence from a previous component C_P being mapped to the expected output sequence by C.

An FSM M can be seen as a function transforming sequences of input actions belonging to $\mathtt{dom}_M$ into sequences of output actions belonging to $\mathtt{image}_M$. Therefore, we could say that M receives an *input* (an element of I^*) and returns an *output* (an element of O^*, with the same length as the input). We define *projections* of this function: for a natural number k, we restrict the function to the set of sequences of input actions that are of length k. In particular, these projections will allow us to consider finite sets of inputs (all the sequences of inputs of a certain length). We also introduce the notion of *collision*: two inputs collide if they produce the same output.

Definition 3 *Let $M = (Q, q_{in}, I, O, T)$ be an FSM. We define $f_M : \mathtt{dom}_M \longrightarrow \mathtt{image}_M$ as the function such that for all $\alpha \in \mathtt{dom}_M$ we have $f_M(\alpha) = \beta$ for β such that $\mathtt{out}_M(q_{in}, \alpha) = \{\beta\}$.*

Let $k > 0$. We define $f_{M,k}$ to be the function $f_M \cap (I^k \times O^k)$, where we use the function f_M to denote the associated set of pairs. Let $\beta \in \mathtt{image}_M$. We define $f_M^{-1}(\beta)$ to be the set $\{\alpha \in I^* | f_M(\alpha) = \beta\}$.

Let $\alpha_1, \alpha_2 \in I^*$. We say that α_1 and α_2 collide for M if $\alpha_1 \neq \alpha_2$ and $f_M(\alpha_1) = f_M(\alpha_2)$.

Note that if two sequences of input actions collide then they must have the same length (otherwise, the returned sequences of output actions would have different length and, therefore, cannot be equal). Next we introduce some notation for random variables and recall the concept of *entropy* [43] associated with a random variable and Squeeziness [13] of a function. The concept of entropy is a "measure of the average uncertainty in the random variable. It is the number of bits on average required to describe the random variable" [43]. In other words, entropy is a measure of the amount of information of a given set with a random variable ranging over it. The concept of Squeeziness then is defined as the amount of information lost after the application of a given function, that is, the difference between the amount of information (entropy) of the domain of the function and the amount of information (entropy) of the range of the function. In a broader sense, we can consider it to measure the difference between the amount of information that we have before applying the function and the amount of information that remains after applying the function. We are interested in total functions since we consider input-enabled FSMs[3].

Definition 4 *Let A be a set and ξ_A be a random variable over A. We denote by σ_{ξ_A} the probability distribution induced by ξ_A. The* entropy *of the random variable ξ_A, denoted by $\mathcal{H}(\xi_A)$, is defined as:*

$$\mathcal{H}(\xi_A) = -\sum_{a \in A} \sigma_{\xi_A}(a) \cdot \log_2(\sigma_{\xi_A}(a))$$

Let $f : A \longrightarrow B$ be a total function and consider two random variables ξ_A and ξ_B ranging, respectively, over A and B. The Squeeziness *of f, denoted by $\mathtt{Sq}(f)$, is defined as the loss of information after applying f to A, that is, $\mathcal{H}(\xi_A) - \mathcal{H}(\xi_B)$.*

As we said, Squeeziness represents the amount of information lost by a given function. Since we have shown that FSMs can be seen as functions from a set of sequences of input actions to a set of sequences of output actions, we can adapt Squeeziness to deal with FSMs. First, we need to define how inputs are chosen and outputs are returned. We consider a probabilistic view where a random variable associated with each set of relevant inputs/outputs is taken into account. We studied two possible alternatives:

- We associate a random variable with the whole set of inputs/outputs (that is, a random variable induces a probability distribution over I^* and O^*, respectively).
- We associate a random variable with the set of inputs/outputs of a certain length (that is, there are different random variables associated with I^1, I^2, $\dots$, O^1, O^2, $\dots$).

In this paper we consider the second approach because it gives us an incremental procedure to compute a sequence of consecutive values of Squeeziness so that we can analyse how the series is *evolving*. Actually, the input sequence length used will depend on the amount of testing to be carried out since this will determine the lengths of the input sequences that a component is likely to receive. Note

[3] Recall that a partial FSM can be mapped to an input-enabled FSM.

also that there is potential to use Squeeziness values, for different input sequence lengths, to inform the choice of test cases. In other words, we use these values with the aim of using test cases that minimise the likelihood of FEP occurring, that is, this approach provides a way to know, for a given length, if the probability of having FEP, once we have tested all the possible inputs with the given length, will be greater than 0 or not. Despite concentrating on the second approach, we believe that the first approach is also interesting. We consider the development of the first approach, and a comparison with the second approach, to be an interesting line of future work.

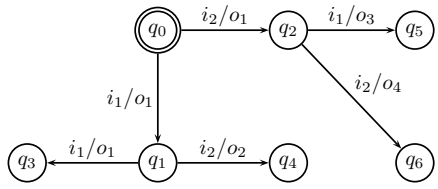
We have that $\mathtt{dom}_{M,k}$ represents the possible inputs of length equal to k that M can perform (therefore, other elements of I^k have probability equal to zero) and $\mathtt{image}_{M,k}$ represents the possible outputs of length equal to k that M can produce after receiving an element of $\mathtt{dom}_{M,k}$. Therefore, defining the random variables that range over each set as $\xi_{\mathtt{dom}_{M,k}}$ and $\xi_{\mathtt{image}_{M,k}}$, we have that the entropy will be the amount of information of each set, and the difference of entropy (that is, $\mathcal{H}(\xi_{\mathtt{dom}_{M,k}}) - \mathcal{H}(\xi_{\mathtt{image}_{M,k}})$) represents the amount of information destroyed by M. This is the notion of Squeeziness that we use in this paper.

Definition 5 *Let $M = (Q, q_{in}, I, O, T)$ be an FSM and $k > 0$. Let us consider two random variables $\xi_{\mathtt{dom}_{M,k}}$ and $\xi_{\mathtt{image}_{M,k}}$ ranging, respectively, over the domain and image of $f_{M,k}$. The Squeeziness of M at length k is defined as*

$$\mathtt{Sq}_k(M) = \mathcal{H}(\xi_{\mathtt{dom}_{M,k}}) - \mathcal{H}(\xi_{\mathtt{image}_{M,k}})$$

Squeeziness for FSMs is an interesting notion that has some unexpected properties. For example, it is not monotonic with respect to k. That is, there exist FSMs where using longer sequences can solve a loss of information produced by shorter sequences.

Example 1 *Consider M, depicted below, where q_0 is the initial state.*



We have that the value of Squeeziness for $k = 1$, assuming a uniform distribution of probabilities, is computed by the following expression

$$2 \cdot \left(-\frac{1}{2} \cdot \log_2\left(\frac{1}{2}\right)\right) - (-1 \cdot \log_2(1)) = \log_2(2) = 1$$

because we have $|\mathtt{dom}_{M,1}| = 2$ and each input of $\mathtt{dom}_{M,k}$ has probability $\frac{1}{2}$ while $|\mathtt{image}_{M,1}| = 1$ and this output has probability 1. Meanwhile, for $k = 2$ we have

$$4 \cdot \left(-\frac{1}{4} \cdot \log_2\left(\frac{1}{4}\right)\right) - 4 \cdot \left(-\frac{1}{4} \cdot \log_2\left(\frac{1}{4}\right)\right) = 0$$

because we have $|\mathtt{dom}_{M,2}| = 4$ and $|\mathtt{image}_{M,2}| = 4$ and each input or output has probability $\frac{1}{4}$ due to the uniform distribution assumption.

Note that obtaining a value of Squeeziness equal to zero for a certain value of k does not imply that Squeeziness will be equal to zero for greater values of k. For example, if we add to q_3, q_4, q_5 and q_6 two outgoing transitions labelled, respectively, by i_1/o_1 and i_2/o_1 and reaching a new state q_7, then we obtain a value of Squeeziness greater than 0 for $k = 3$.

An important remark concerning random variables associated with inputs and outputs is that given an FSM M, $k > 0$ and a random variable $\xi_{\mathtt{dom}_{M,k}}$ we have that the probability distribution of the random variable $\xi_{\mathtt{image}_{M,k}}$ is completely determined. This is because for each element $\beta \in \mathtt{image}_{M,k}$ we have that

$$\sigma_{\xi_{\mathtt{image}_{M,k}}}(\beta) = \sum_{\alpha \in f_M^{-1}(\beta)} \sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha)$$

The following result is immediate from the definition of entropy and the previous explanation concerning how the random variable associated with outputs is determined by the one corresponding to inputs.

Lemma 1 *Let $M = (Q, q_{in}, I, O, T)$ be an FSM and $k > 0$. If $f_{M,k}$ is bijective then $\mathtt{Sq}_k(M) = 0$.*

Next, we present an alternative formulation of Squeeziness. The proof of the following result, given in the appendix, follows from the partition property of entropy [14] and the definition of $\sigma_{\xi_{\mathtt{image}_{M,k}}}$ in terms of $\sigma_{\xi_{\mathtt{dom}_{M,k}}}$. First, we give an auxiliary result concerning conditional distributions of random variables (the proof is also in the appendix). In the following, $\xi_1|\xi_2$ denotes the conditional random variable ξ_1 given ξ_2.

Lemma 2 *Let $M = (Q, q_{in}, I, O, T)$ be an FSM and $k > 0$. Let us consider two random variables $\xi_{\mathtt{dom}_{M,k}}$ and $\xi_{\mathtt{image}_{M,k}}$ ranging, respectively, over the domain and image of $f_{M,k}$. We have that $\mathcal{H}(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}}) = 0$.*

Proposition 1 *Let $M = (Q, q_{in}, I, O, T)$ be an FSM and $k > 0$. Let us consider two random variables $\xi_{\mathtt{dom}_{M,k}}$ and $\xi_{\mathtt{image}_{M,k}}$ ranging, respectively, over the domain and image of $f_{M,k}$. We have that*

$$\mathcal{H}(\xi_{\mathtt{dom}_{M,k}}) = \mathcal{H}(\xi_{\mathtt{image}_{M,k}}) - \mathcal{P}(M, \xi_{\mathtt{image}_{M,k}})$$

where the term $\mathcal{P}(M, \xi_{\mathtt{image}_{M,k}})$ is equal to

$$\sum_{\beta \in \mathtt{image}_{M,k}} \sigma_{\xi_{\mathtt{image}_{M,k}}}(\beta) \cdot \left(\sum_{\alpha \in f_M^{-1}(\beta)} \sigma_{\xi_{f_M^{-1}(\beta)}}(\alpha) \cdot \log_2(\sigma_{\xi_{f_M^{-1}(\beta)}}(\alpha)) \right)$$

A trivial corollary of the previous result provides an alternative definition of Squeeziness where the value is computed in terms of the inverse images partition of the input space taking into account, as previously explained, that we have

$$\sigma_{\xi_{\mathtt{image}_{M,k}}}(\beta) = \sum_{\alpha \in f_M^{-1}(\beta)} \sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha)$$

Therefore, we only use the probability distribution on inputs given by $\xi_{\mathtt{dom}_{M,k}}$.

Corollary 1 *Let $M = (Q, q_{in}, I, O, T)$ be an FSM and $k > 0$. Let us consider a random variable $\xi_{\mathtt{dom}_{M,k}}$ ranging over the domain of $f_{M,k}$. We have that*

$$\mathtt{Sq}_k(M) = -\sum_{\beta \in \mathtt{image}_{M,k}} \left(\sum_{\alpha \in f_M^{-1}(\beta)} \sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha) \right) \cdot \mathcal{R}_M(\beta)$$

where the term $\mathcal{R}_M(\beta)$ is equal to

$$\left(\sum_{\alpha \in f_M^{-1}(\beta)} \frac{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha)}{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(f_M^{-1}(\beta))} \cdot \log_2 \left(\frac{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha)}{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(f_M^{-1}(\beta))} \right) \right)$$

The above notion of Squeeziness is parameterised by the distribution over inputs to the function (and so input sequences). If we know the actual distribution then we can use this. If we do not know the distribution then there is a need to choose one and we now discuss two approaches to this.

3.1. Maximum entropy principle

In general, it is not possible to know the probability distribution that ranges over the inputs. Therefore, if we want to have an estimation of the different values of Squeeziness for a given FSM, then we need to make an assumption about this distribution. There are different possibilities. For example, we can assume *maximum entropy*, that is, we choose a probability distribution that maximises the entropy. If there are no further restrictions, then maximum entropy is obtained with a uniform distribution [14]. In this case, the weight of a single element of $\sigma_{\xi_{\mathtt{dom}_{M,k}}}$ is $\frac{1}{|\mathtt{dom}_{M,k}|}$. Thus, the weight of the inverse image of an output $\beta \in \mathtt{image}_{M,k}$ is equal to $\frac{|f_M^{-1}(\beta)|}{|\mathtt{dom}_{M,k}|}$. Finally, Squeeziness under the assumption of having a uniform distribution over inputs is equal to

$$\begin{aligned}
\mathtt{Sq}_k(M) &= -\sum_{\beta \in \mathtt{image}_{M,k}} \left(\sum_{\alpha \in f_M^{-1}(\beta)} \frac{1}{|\mathtt{dom}_{M,k}|} \right) \\
&\quad \cdot \left(\sum_{\alpha \in f_M^{-1}(\beta)} \frac{\frac{1}{|\mathtt{dom}_{M,k}|}}{\frac{|f_M^{-1}(\beta)|}{|\mathtt{dom}_{M,k}|}} \cdot \log_2 \left(\frac{\frac{1}{|\mathtt{dom}_{M,k}|}}{\frac{|f_M^{-1}(\beta)|}{|\mathtt{dom}_{M,k}|}} \right) \right) \\
&= -\sum_{\beta \in \mathtt{image}_{M,k}} \frac{|f_M^{-1}(\beta)|}{|\mathtt{dom}_{M,k}|} \\
&\quad \cdot \left(\frac{|f_M^{-1}(\beta)|}{|f_M^{-1}(\beta)|} \cdot \log_2 \left(\frac{1}{|f_M^{-1}(\beta)|} \right) \right) \\
&= -\sum_{\beta \in \mathtt{image}_{M,k}} \frac{|f_M^{-1}(\beta)|}{|\mathtt{dom}_{M,k}|} \cdot \log_2 \left(\frac{1}{|f_M^{-1}(\beta)|} \right) \\
&= \frac{1}{|\mathtt{dom}_{M,k}|} \cdot \sum_{\beta \in \mathtt{image}_{M,k}} |f_M^{-1}(\beta)| \cdot \log_2(|f_M^{-1}(\beta)|)
\end{aligned}$$

3.2. Maximum loss of information

Another strategy considers the worst case scenario, that is, we may suppose that the chosen probability distribution induces the maximum loss of information. In other words, we look for a probability distribution that maximises Squeeziness. This distribution is uniformly distributed in the largest inverse image of an element of the outputs and zero elsewhere [13]. Formally, consider $\beta' \in \mathtt{image}_{M,k}$ such that for all $\beta \in \mathtt{image}_{M,k}$ we have that $|f_M^{-1}(\beta')| \geq |f_M^{-1}(\beta)|$. Then,

$$\sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha) = \begin{cases} \frac{1}{|f_M^{-1}(\beta')|} & \text{if } \alpha \in f_M^{-1}(\beta') \\ 0 & \text{otherwise} \end{cases}$$

Using this probability distribution, Squeeziness is defined as follows:

$$\begin{aligned}
\mathtt{Sq}_k(M) &= -\left(\sum_{\alpha \in f_M^{-1}(\beta')} \frac{1}{|f_M^{-1}(\beta')|} \right) \\
&\quad \cdot \left(\sum_{\alpha \in f_M^{-1}(\beta')} \frac{1}{|f_M^{-1}(\beta')|} \cdot \log_2 \left(\frac{1}{|f_M^{-1}(\beta')|} \right) \right) \\
&= -\frac{|f_M^{-1}(\beta')|}{|f_M^{-1}(\beta')|} \cdot \left(\frac{|f_M^{-1}(\beta')|}{|f_M^{-1}(\beta')|} \cdot \log_2 \left(\frac{1}{|f_M^{-1}(\beta')|} \right) \right) \\
&= -\log_2 \left(\frac{1}{|f_M^{-1}(\beta')|} \right) \\
&= \log_2(|f_M^{-1}(\beta')|)
\end{aligned}$$

Let us remark that this probability distribution maximises Squeeziness because for any other possible distribution $\xi_{\mathtt{dom}_{M,k}}$ we have $\mathtt{Sq}_k(M) \leq \log_2(|f_M^{-1}(\beta')|)$. This result is an immediate consequence of the following result [13].

Lemma 3 *Let us consider $2 \cdot n$ non-negative real numbers $a_1, \ldots, a_n, p_1, \ldots, p_n \in \mathbb{R}^+$. If for all $1 \leq i \leq n$ we have that $a_1 \geq a_i$ and $\sum_i p_i \leq 1$, then $\sum_i (p_i \cdot a_i) \leq a_1$.*

An important consequence of this result is that it allows us to define a normalisation of the value of Squeeziness, if needed, so that we can have a concept of *normalised Squeeziness*. Later we will see that in the experiments we explored this normalised Squeeziness, which was obtained by dividing Squeeziness by the size of the maximum inverse domain of any output.

3.3. Domain to Range Ratio vs. Squeeziness

It is difficult to compare Squeeziness with other notions to compute fault masking because the literature is very scarce. One of the few notions in this line is the Domain to Range Ratio (DRR) [46]. In this section we explore how Squeeziness and DRR relate. In the next section we report on results of experiments that compared DRR with our notion of Squeeziness. First, we give the original definition.

Definition 6 *Let $f : I \longrightarrow O$ be a total and surjective function. We define the Domain to Range Ratio of f, denoted by $\texttt{DRR}(f)$, as $\frac{|I|}{|O|}$.*

Next, we adapt this notion to our framework. Note that our *functions* are total and surjective because we restrict ourselves to their domains and ranges.

Definition 7 *Let $M = (Q, q_{in}, I, O, T)$ be an FSM and $k > 0$. Let us consider $f_{M,k} : \texttt{dom}_{M,k} \longrightarrow \texttt{image}_{M,k}$. We define the Domain to Range Ratio for M and k, denoted by $\texttt{DRR}(f_{M,k})$, as $\frac{|\texttt{dom}_{M,k}|}{|\texttt{image}_{M,k}|}$.*

The next result, whose proof is in the appendix, shows that this measure is inconsistent with Squeeziness.

Lemma 4 *There exist FSMs M_1 and M_2 and $k > 0$ such that $\texttt{DRR}(f_{M_1,k}) = \texttt{DRR}(f_{M_2,k})$ but $\texttt{Sq}_k(M_1) \neq \texttt{Sq}_k(M_2)$.*

There exist FSMs M_1 and M_2 and $k > 0$ such that $\texttt{DRR}(f_{M_1,k}) < \texttt{DRR}(f_{M_2,k})$ but $\texttt{Sq}_k(M_1) > \texttt{Sq}_k(M_2)$.

4. Empirical Evaluation

In this section we outline the different experiments carried out to evaluate the proposed measure. First, we describe the experiments that used simulations, that is, instead of FSMs we consider sequences of input/output actions. Next we explain the experiments that used FSMs. We conclude the section with an evaluation of the threats to validity, and how they were addressed, and a discussion about the obtained results and some of their implications.

4.1. Evaluation via simulations

We outline an evaluation in which we simulated an FSM by randomly generating the sizes of the inverse images of the output sequences. We designed the simulation in this way since it represents the notion of FEP that we consider in this paper: one in which one component C masks a fault in another component C_P by mapping two potential output sequences α and α' for C_P (and so input sequences for C) to the same output sequence β. This scenario corresponds to FEP if, for example, α is an expected (correct) output sequence for C_P and α' is a possible faulty output sequence (for C_P). Observe that this type of FEP occurs if and only if α and α' are both in the inverse image of the same output sequence β. As a result, in order to reason about the probability of FEP it is sufficient to retain only the information about the sizes of the inverse images of output sequences and so we simulate these values (the sizes of the inverse images of output sequences).

In this section we first introduce a collision measure, which is the probability of FEP occurring. Although this collision measure could potentially be used to reason about FEP, as usual for this type of measure, it is computationally expensive to compute it. Therefore, it is important to study alternative measures, based on Information Theory, that are either less computationally expensive or that can be efficiently estimated. Having defined the collision measure, we then use experiments with randomly generated scenarios in order to compare this with our information theoretic measure and the Domain to Range Ratio.

Research Question 1 *Is there a correlation between the measures defined in this paper and the probability of a component introducing FEP through masking incorrect output produced by earlier components?*

4.1.1. Collisions and FEP

In our context, fault masking (FEP) happens when the expected and faulty input sequences, received from another component, produce the same sequence β of output actions. If given an FSM M and $k > 0$ we have that there exist $\beta \in \texttt{image}_{M,k}$ such that $\alpha, \alpha' \in f^{-1}_{M,k}(\beta)$, with $\alpha \neq \alpha'$, then there is a collision and this might hide a fault. Next we provide a notion to compute the probability of having a collision.

Definition 8 *Let M be an FSM and $k > 0$. Let $\texttt{image}_{M,k} = \{\beta_1, ..., \beta_n\}$ and for all $1 \leq i \leq n$ let $I_i = f^{-1}_{M,k}(\beta_i)$ and $m_i = |f^{-1}_{M,k}(\beta_i)|$. We have that $d = \sum_{i=1}^{n} m_i$ is the size of the input space.*

Given a uniform distribution over the inputs, the probability of α and α' both being in the set I_i is equal to $p_i = \frac{m_i \cdot (m_i - 1)}{d \cdot (d-1)}$. We have that the probability of having a collision in M for sequences of length k, denoted by $\texttt{PColl}_k(M)$, is given by

$$\texttt{PColl}_k(M) = \sum_{i=1}^{n} \frac{m_i \cdot (m_i - 1)}{d \cdot (d-1)}$$

Observe that there is potential to use this measure, $\texttt{PColl}_k(M)$, instead of Squeeziness. The problem with using $\texttt{PColl}_k(M)$ is that it is computationally hard to compute. While this also applies to Squeeziness, it has the advantage of being an information theoretic measure. As a result, there is potential to draw on Information Theory research that has devised techniques that either estimate or bound measures [8, 11]. Note that estimates and bounds will suffice as long as they are useful - they do not need to be precise. This is in contrast to some applications of Information Theory, such as security, in which we require guarantees. We therefore expect that much smaller samples should suffice.

Previous work [13] states that PColl can be seen as a probability of collisions when the probability distribution over the inputs is uniform. However, it is worth to mention that the relationship between $\texttt{PColl}_k(M)$ and $\texttt{Sq}_k(M)$ is not, in general, monotonic. The proof of the following result is given in the appendix.

Lemma 5 *There exist FSMs M_1 and M_2 and $k > 0$ such that $\texttt{Sq}_k(M_1) < \texttt{Sq}_k(M_2)$ but $\texttt{PColl}_k(M_1) > \texttt{PColl}_k(M_2)$.*

4.1.2. Experimental Results

We now report on simulations that compared PColl, Sq and DRR. The three measures are defined (assuming uniform distributions over the inputs) in terms of the sizes of the subdomains ($f^{-1}_{M,k}(\beta)$). Our methodology to perform simulations followed the approach used in the original work on Squeeziness [13] but used a much wider range of scenarios.

First, we fixed the size of the input space (denoted by d) and a maximum subdomain size (denoted by m). Next, we generated random integers between 1 and m until the values summed to d; if the sum of the values exceeded d then the last value was suitably reduced. Once we had these partitions, we computed the three measures. This way, d represents the number of different inputs of a fixed length k that the *simulated* FSM has and each partition (each random number) represents one output, whose value is the number of inputs that are in the inverse image of this output (i.e. the number of inputs that generate this output). This process was repeated 200 times for each pair (d, m) and we computed the Pearson correlation coefficient between these 200 values of PColl and the other two measures. We used 120 pairs with d ranging between 10^4 and $2 \cdot 10^9$ and m ranging between 10^2 and 10^4. We also computed the Spearman Rank correlation coefficient, but the results were almost identical, so we will not discuss these correlation coefficients. For each pair (d, m) we performed the entire process twice.

The main result is that there is a strong correlation between PColl and Sq, with all of the values being greater than 0.96.

We obtained a not so strong correlation between PColl and DRR, with all correlations being between 0.86 and 0.67.

Input set size	Maximum size	Correlation of Sq	Correlation of DRR
10,000	100	0.968366	0.763623
10,000	100	0.973918	0.783759
10,000	200	0.973016	0.823959
10,000	200	0.967349	0.77492
10,000	10,000	0.967281	0.71496
10,000	10,000	0.966184	0.670497
100,000	500	0.980028	0.836659
100,000	500	0.972878	0.769055
500,000	5,000	0.95885	0.743651
500,000	5,000	0.969437	0.765643
2,000,000	5,000	0.978818	0.810967
2,000,000	5,000	0.964455	0.698505
200,000,000	2,000	0.974498	0.799512
200,000,000	2,000	0.980097	0.843219
1,000,000,000	200	0.978771	0.859822
1,000,000,000	200	0.968844	0.759952
2,000,000,000	5,000	0.970575	0.807333
2,000,000,000	5,000	0.965495	0.781112
2,000,000,000	10,000	0.969172	0.79843
2,000,000,000	10,000	0.972477	0.783512

Table 1: Representative results from the simulation

Interestingly, the correlation between PColl and DRR, appears not to change as we increase the size of the input domain. This is in contrast to the previous white-box work, which found that increases in input-domain size led to a reduction in the effectiveness of all measures used [13]. This is promising since it suggests that effectiveness may be more robust in the context considered in this paper and so the measures may be effective in a wider range of scenarios.

A number of the most representative results can be found in Table 1 while the full set of results can be found in the appendix of the paper. Specifically, we have given the cases that obtain the highest Sq correlation, the highest DRR correlation, the lowest Sq correlation and the lowest DRR correlation. Also, we give the cases corresponding to the smallest scenario (that is, input set size of 10,000 and maximum subdomain size of 100) and the largest scenario (that is, input set size of 2,000,000,000 and maximum subdomain size of 10,000). Finally, we have given some cases in which the Sq and/or DRR values are around the mean of the values of each measure (Sq and DRR). It is important to note that when we show a case, we display the result of both runs, although the result of interest need not appear on both runs.

As a side note, we performed an additional experiment but the results were worse than expected. Specifically, we computed the results also for the normalised version of Squeeziness that we mentioned in Section 3.2, which is obtained by dividing Squeeziness by the size of the maximum inverse domain of any output. However, some of the correlations obtained were relatively small (see Table 2). Interestingly, we found very poor correlations for some of the small input sets, while the corresponding correlations for Squeeziness were good. Therefore, we decided to no longer consider this form of *normalised Squeeziness* during the rest of our experiments.

Input set lenght	Maximum size	Correlation of NormalizedSqueeziness
10,000	100	0.958346
10,000	100	0.961652
10,000	200	0.950883
10,000	200	0.926334
10,000	5,000	0.469301
10,000	5,000	0.471505
10,000	10,000	0.427412
10,000	10,000	0.470961
20,000	10,000	0.415143
20,000	10,000	0.515837
100,000	500	0.972534
100,000	500	0.96534
500,000	5,000	0.926782
500,000	5,000	0.949692
2,000,000	5,000	0.971917
2,000,000	5,000	0.958095
200,000,000	2,000	0.974498
200,000,000	2,000	0.980097
1,000,000,000	200	0.978771
1,000,000,000	200	0.968844
2,000,000,000	5,000	0.970575
2,000,000,000	5,000	0.965495
2,000,000,000	10,000	0.969172
2,000,000,000	10,000	0.972477

Table 2: Representative results from the simulation with normalized Squeeziness

4.2. Empirical evaluation using FSMs

In the previous section we reported on the results of simulations that showed that Squeeziness is related to the probability of FEP. The simulations represented general functions, with finite input domains, by giving the sizes of the inverse images of outputs. However, it is unclear whether these simulations correspond to functions that can be described using FSMs and so in this section we report on the results of experiments that used FSM models. The experiments were driven by one research question that assessed whether the measure can be used as intended when we have FSMs.

Research Question 2 *When using FSMs, is there a correlation between the measures defined in this paper and the probability of a component introducing FEP through masking incorrect output produced by earlier components?*

Next we report on the results of an experiment that assessed this research question. First, we briefly explain how we generated the (FSMs) used in our experiments.

4.2.1. FSM Generator

In order to perform our experiments we need to generate FSMs. We developed an FSM generator that randomly generates FSMs given some parameters.[4] The first issue we solved was to fix the internal representation of FSMs. Since our work is not the first one dealing with FSMs we decided to review the literature and found the OpenFST library [1]. This library is intended to work with Finite State Transducers (as its name indicates). These are a kind of FSMs with an input/output pair in each transition and a weight. Therefore, we simply ignored the weight. This library also provides shell commands that we can use, in particular, to generate the associated binary files and to generate the topological representation of each FSM as an image.

Once we had a proper representation for our FSMs, we developed the tool for generating those FSMs. The main reason for developing this tool was to generate a wide range of different FSMs that have some specific properties. In order to have a general tool that can be used in a range of experiments, we included some basic parameters:

- *#Rep*: the number of FSMs we want to generate.
- *Max_States*: the maximum number of states an FSM can have.
- *Min_States*: the minimum number of states an FSM must have.
- *Max_Transitions*: the maximum number of transitions each state of an FSM can have.
- *Min_Transitions*: the minimum number of transitions each state of an FSM must have.
- *#Inputs*: the number of inputs.
- *#Outputs*: the number of outputs.

After setting these basic parameters, the program can be executed. The execution flow for generating an FSM using *#Rep* is given in Algorithm 1. Note that, by construction, the tool returns connected (all states are reachable from the initial state) and deterministic FSMs. Also note that the algorithm allows the construction of FSMs that have loops.

In order to create input-enabled FSMs with our tool, as used in our experiment, we simply set *Min_Transitions* = *Max_Transitions* = *#Inputs*.

4.2.2. Experimental results

This section describes the results of experiments that addressed the research question. Similar to Section 4.1, we compared our measure with the probability of collision, but this time for the specific FSMs being considered. Recall that the probability of collision is given by the following expression:

$$\texttt{PColl}_k(M) = \sum_{j=1}^{n} \frac{m_j \cdot (m_j - 1)}{d \cdot (d-1)}$$

where m_j is the cardinality of the inverse image of the j-th output (i.e. the number of inputs that lead to this output) and d is the cardinality of the inputs (i.e. the total

[4] All the tools developed to perform the experiments of this paper are freely available at https://github.com/Colosu/FSTGenerator .

Result: *#Rep* FSMs.
machine = 0;
while *machine* < *#Rep* **do**
  Create a folder to save the FSM files;
  Set a random number S of states between *Min_States* and *Max_States* for the FSM;
  Choose the state 0 as initial state;
  for *each state* $0 \leq i < S-1$ *of the machine* **do**
    Set a random number T of transitions between *Min_Transitions* and *Max_Transitions* for the state;
    for *each transition* $0 \leq j < T$ *of the state* **do**
      if $j == 0$ **then**
        Set the state $i+1$ as the end of the transition;
      else
        Set a random state as the end of the transition;
      end
      Set a random input label for the transition not previously used for another transition of the state (so FSMs are deterministic);
      Set a random output label for the transition;
      Save this transition to the FSM file;
    end
  end
  Create the binary file that the OpenFST library uses to interpret FSMs using the FSM file we created;
  Create a pdf image with the FSM topology;
  machine + +;
end

Algorithm 1: FSM generation algorithm

Run Number	Pearson Sq	Pearson DRR	Spearman Sq	Spearman DRR
1	0.878449	0.789925	0.915152	0.835599
2	0.769163	0.577342	0.709091	0.527583
3	0.926841	0.836864	0.939394	0.69347
4	0.919335	0.843178	0.890909	0.811444
5	0.888474	0.85478	0.733333	0.71462
6	0.779444	0.56354	0.842424	0.67769
7	0.899583	0.87125	0.927273	0.885083
8	0.895344	0.792316	0.854545	0.877186
9	0.683355	0.46901	0.842424	0.610832
10	0.906801	0.909021	0.830303	0.887425
11	0.56845	0.483205	0.721212	0.592422
12	0.838746	0.834886	0.672727	0.544839
13	0.630317	0.531773	0.793939	0.551174
14	0.410659	0.504509	0.272727	0.355335
15	0.640715	0.56302	0.151515	$8.36862e-18$
16	0.73553	0.601444	0.757576	0.549532
17	0.272227	0.160886	0.333333	0.113904
18	0.679269	0.577716	0.50303	0.449199
19	0.505532	0.276551	0.684848	0.334363
20	0.866044	0.856532	0.854545	0.877186
21	0.899159	0.832556	0.890909	0.830399
22	0.273041	0.0300172	0.224242	0.012975
23	0.635755	0.635614	0.830303	0.740844
24	0.907813	0.853587	0.854545	0.889898
25	0.804562	0.694005	0.660606	0.563845
26	0.438958	0.299874	0.6	0.375029
27	0.75262	0.577602	0.563636	0.394771
28	0.900993	0.911372	0.939394	0.885657
29	0.909105	0.863749	0.842424	0.805143
30	0.88053	0.818995	0.527273	0.644304
31	0.864043	0.782816	0.709091	0.664867
32	0.782251	0.763869	0.69697	0.846658
33	0.891232	0.815343	0.709091	0.660696
34	0.707623	0.522515	0.709091	0.486655
35	0.608514	0.549941	0.648485	0.589186
36	0.89894	0.824825	0.963636	0.806406
37	0.680069	0.520374	0.648485	0.555997
38	0.718919	0.528805	0.866667	0.761549
39	0.803944	0.857559	0.575758	0.71462
40	0.749198	0.46362	0.818182	0.635946
41	0.841066	0.416991	0.830303	0.341882
42	0.871523	0.699391	0.709091	0.661358
43	0.841827	0.606074	0.939394	0.905111
44	0.783823	0.706252	0.684848	0.552679
45	0.156226	−0.0418695	0.0545455	−0.12975
46	0.911163	0.819401	0.806061	0.793018
47	0.883863	0.780593	0.878788	0.774176
48	0.711095	0.516978	0.866667	0.603382
49	0.76034	0.434219	0.806061	0.568535
50	0.710906	0.75102	0.672727	0.742155

Table 3: Results from the experiment with 500 FSMs with 25 states.

number of inputs). Squeeziness was designed to compare models with the same input domains. In order to facilitate this task, we used input-enabled FSMs but the results are essentially the same if we use non input-enabled FSMs (as long as we consider the same number of input sequences in all the FSMs). We generated 500 machines with 25 states and 5 outgoing transitions from each state. We considered sets of 5 inputs and 5 outputs.

Having generated the FSMs, we computed Squeeziness and PColl for each FSM. The next step was to randomly partition the set of FSMs into groups of 10 and compute the (Pearson and Spearman) correlations (between Squeeziness and PColl) for each group. We used multiple groups in order to obtain insights into the consistency of the results. Therefore, we obtained 50 values for each correlation coefficient. Note that the number of input sequences that we have to consider grows exponentially with the input sequence length. As a result of this exponential growth, and memory limits, we computed the measures for input sequences of length 10.

Similar to the simulations, we obtained positive experimental results. The results of this experiment can be found in Table 3. In most cases, the results show a high correlation between Squeeziness and PColl, with a mean of 0.745468 for Pearson and 0.715152 for Spearman. This fact supports the results from the simulations. It is interesting to see that there were a few relatively small values but it seems likely that these were simply the result of the randomness in the experiments (we also observe some higher correlations, up to 0.96). Also, we can see that in most of the cases the correlation values of Squeeziness are higher than the ones of DRR, as we saw in the simulations. Specifically, the mean for DRR is equal to 0.634677 for Pearson and equal to 0.611338 for Spearman. These values are noticeably lower than the ones corresponding to the correlations between Squeeziness and PColl.

In order to check the flexibility of our results, we decide

Run Number	Pearson Sq	Pearson DRR	Spearman Sq	Spearman DRR
1	0.856654	0.759691	0.839822	0.718795
2	0.799091	0.691343	0.803782	0.710352
3	0.83536	0.720232	0.901224	0.809638
4	0.812958	0.809717	0.733037	0.667405
5	0.637571	0.565636	0.474972	0.439789
6	0.628766	0.550766	0.573304	0.468414
7	0.78118	0.662821	0.829143	0.722609
8	0.648335	0.563561	0.689433	0.607159
9	0.651849	0.52149	0.599555	0.452153
10	0.890877	0.84885	0.866518	0.83724
11	0.822498	0.745911	0.78109	0.776184
12	0.756156	0.648846	0.788654	0.674843
13	0.705051	0.628704	0.751724	0.733294
14	0.812525	0.539454	0.85673	0.580215
15	0.750044	0.522172	0.699221	0.548734
16	0.794857	0.677268	0.866963	0.742223
17	0.761287	0.696498	0.739711	0.695613
18	0.711764	0.700859	0.676085	0.626331
19	0.874628	0.85356	0.777976	0.775392
20	0.916858	0.897105	0.874527	0.792335
21	0.871061	0.902461	0.811791	0.814555
22	0.896291	0.867035	0.822469	0.733963
23	0.815214	0.79039	0.599555	0.478928
24	0.731722	0.642186	0.721913	0.593555
25	0.764644	0.677635	0.826029	0.77864
26	0.624603	0.626964	0.618687	0.585576
27	0.731404	0.558051	0.879422	0.838548
28	0.727717	0.659671	0.725918	0.601497
29	0.828107	0.716817	0.823359	0.673092
30	0.544259	0.319727	0.630256	0.430179

Table 4: Results from the experiment with 900 FSMs with 25 states.

to repeat the experiment with a different configuration. We generated 900 machines with the same characteristics: 25 states, 5 outgoing transitions from each state and sets of inputs and outputs with 5 elements. We grouped the machines in 30 groups of 30 machines per group and repeated the experiment. Previously we used groups of 10 FSMs and using larger groups of FSMs allows us to check our intuition that there should be greater consistency in the results. The results are shown in Table 4. These results are slightly better than the previous ones, with a higher similarity between Pearson and Spearman correlations. In this case, the means of the correlations between Squeeziness and PColl are equal to 0.766111 for Pearson and equal to 0.752762 for Spearman; the ones corresponding to DRR are equal to 0.678847 for Pearson and equal to 0.663575 for Spearman. Therefore, the conclusions of the experiment are similar to the ones obtained in the previous experiment.

4.3. Threats to Validity

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

First, we explore threats to internal validity, which consider uncontrolled factors that might be responsible for the obtained results. In our work, the main threat to internal validity is associated with the possible faults in the developed tools, which could lead to misleading results. In order to reduce the impact of this threat we tested our code with carefully constructed examples for which we could manually check the results. In addition, we repeated each experiment that used FSMs in order to check that the results were consistent and there was no randomisation involved.

Second, we consider threats to external validity, which concern conditions that allow us to generalise our findings to other situations. In our work, the main external threat is the different possible representations of a black-box component as an FSM. Such a threat cannot be entirely addressed since the population of such 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 both a large number of simulations and of randomly generated FSMs. Note also that the simulations provided significant diversity in terms of experimental subjects, with the role of the FSM-based experiments primarily being to check that the results extend to the class of functions that can be represented by FSMs.

Last, we consider threats to construct validity. This is related to the *reality* of our experiments, that is, whether our experiments reflect real-world situations. In our work, the main construct threat is whether the FSMs used in the experiments correspond to possible system components. In order to reduce the impact of this threat, we restricted our range of FSM samples to connected deterministic machines. In future work we intend to test with real-world cases and/or non-deterministic FSMs.

4.4. Discussion

The two sets of results presented in this section were encouraging. The simulations showed that there is a strong positive correlation between our notion of Squeeziness and a measure of the probability of collisions if we simulate the function computed by a component. As expected, this correlation was higher than the one that we obtained with DRR, with this being consistently seen across the 24,000 simulation experiments.

The simulations addressed the suitability of our measures for a general framework, in which we have a function that represents the (input/output) behaviour of the component of interest. Since we developed the details of the framework for FSM models we also had experiments that explored whether similar results hold for functions that can be represented by FSMs. The results of these experiments were similar to those previously observed, supporting the results that used simulations.

Interestingly, the correlations returned were slightly lower when using FSMs. There are at least three possible explanations for the differences. First, the experiments that used FSMs considered a smaller set of scenarios; it may be that we would observe results similar to those found in the simulations if we ran many more experiments with a wider range of FSMs. Second, the simulations may have used functions that are rather different from those found when using FSMs. If this is the case then the results might be better if we use more general types of models as specifications of components (rather than FSMs). Third, the differences may result from the simulations using larger sample sizes (sample size 200) than the experiments with FSMs (sample size 10). Note that the smaller sample size

Run Number	Pearson Sq	Spearman Sq
1	0.279452	0.333333
2	0.55035	0.309091
3	0.0716533	−0.151515
4	0.77656	0.890909
5	0.622	0.890909
6	0.66114	0.660606
7	0.655833	0.757576
8	0.317683	0.321212
9	0.87951	0.818182
10	0.798106	0.878788
11	0.834614	0.781818
12	0.732538	0.733333
13	0.27344	0.406061
14	0.583361	0.478788
15	0.580003	0.769697
16	0.892117	0.939394
17	0.166601	−0.0909091
18	0.455388	0.50303
19	0.843653	0.660606
20	0.551302	0.672727
21	0.901526	0.927273
22	0.89004	0.842424
23	0.442546	0.393939
24	0.683401	0.50303
25	0.931557	0.757576
26	0.52448	0.321212
27	0.600394	0.539394
28	0.323964	0.284848
29	0.698155	0.684848
30	0.61123	0.672727
31	0.559464	0.745455
32	0.765045	0.454545
33	0.570081	0.430303
34	0.859868	0.878788
35	0.917171	0.878788
36	0.837555	0.721212
37	0.539043	0.490909
38	0.704665	0.660606
39	0.740488	0.878788
40	0.552796	0.527273
41	0.717961	0.757576
42	0.634331	0.6
43	0.563094	0.672727
44	0.749326	0.587879
45	0.877225	0.866667
46	0.584465	0.587879
47	0.716488	0.10303
48	0.392777	0.563636
49	0.866184	0.890909
50	0.597951	0.672727

Table 5: Results from the experiment with FSMs between 10 and 25 states.

Run Number	Pearson Sq	Spearman Sq
1	0.962624	0.951515
2	0.770315	0.624242
3	0.861972	0.769697
4	0.806692	0.660606
5	0.782762	0.745455
6	0.582544	0.842424
7	0.777772	0.781818
8	0.848743	0.793939
9	0.282844	0.309091
10	0.749289	0.660606
11	0.815693	0.587879
12	0.465784	0.527273
13	0.854386	0.866667
14	0.514125	0.478788
15	0.94217	0.90303
16	0.956262	0.90303
17	0.556821	0.539394
18	0.658927	0.478788
19	0.423597	0.478788
20	0.927198	0.866667
21	0.225621	0.515152
22	0.800149	0.769697
23	0.732404	0.769697
24	0.964573	0.90303
25	0.693287	0.563636
26	0.904747	0.612121
27	0.797333	0.684848
28	0.950163	0.842424
29	0.874851	0.793939
30	0.547064	0.709091
31	0.81926	0.866667
32	0.783165	0.878788
33	0.872504	0.866667
34	0.576504	0.50303
35	0.827418	0.915152
36	0.894726	0.781818
37	0.814328	0.406061
38	0.76672	0.890909
39	0.829572	0.648485
40	0.92247	0.951515
41	0.913127	0.806061
42	0.804393	0.781818
43	0.786996	0.866667
44	0.643162	0.612121
45	0.72758	0.721212
46	0.781083	0.757576
47	0.823419	0.915152
48	0.807975	0.818182
49	0.728113	0.793939
50	0.731225	0.769697

Table 6: Results from the experiment with FSMs between 25 and 50 states.

used in the FSM experiments (for practical reasons) could also explain the greater variability observed in the results.

In order to explore the two first possibilities, we repeated the FSM experiment with three new sets of FSMs. This allowed us to consider more scenarios (exactly, $1,500$ additional ones) and to test if increasing or reducing the generality of the functions represented by the FSMs has any effect on the correlations.

The first set of samples included FSMs that had between 10 and 25 states, so we place even stronger limits on the generality of the functions represented. As expected, we got slightly worse correlations, with only 77% of samples having correlations greater than 0.5 instead of the 93% that we got with the initial experiment. That leads to a mean of 0.6376 for the Pearson correlation and of 0.6092 for the Spearman one. The full results can be found in Table 5.

The second set of samples considered FSMs that had between 25 and 50 states, slightly increasing the generality of the functions represented. The results were reasonably similar to the initial FSM results, in that 91% of samples had correlations greater than 0.5. However, the results were arguably a little better since the lowest value was greater than 0.2 (instead of being negative as in the initial experiment). That leads to a mean of 0.7577 for the Pearson correlation and of 0.7297 for the Spearman one. The full results can be found in Table 6.

In order to increase the confidence on the validity of our results when the number of states increase, we performed an additional experiment where all the FSMs have the same number of states (75). The values show that 98% of the results have correlations greater than 0.5, and the lowest correlation is greater than 0.4. That leads to a mean of 0.844027 for the Pearson correlation and of 0.810182 for the Spearman one. The full results are provided in Table 7.

The results suggest that correlations will be better if

Run Number	Pearson Sq	Spearman Sq
1	0.872409	0.915152
2	0.744849	0.527273
3	0.910456	0.951515
4	0.943024	0.854545
5	0.828649	0.927273
6	0.837167	0.757576
7	0.797805	0.866667
8	0.89309	0.745455
9	0.46309	0.478788
10	0.94864	0.951515
11	0.951563	0.745455
12	0.973901	0.878788
13	0.973626	0.951515
14	0.991222	0.915152
15	0.862179	0.721212
16	0.92241	0.915152
17	0.796633	0.587879
18	0.991884	0.721212
19	0.649892	0.684848
20	0.897328	0.90303
21	0.783819	0.721212
22	0.653267	0.648485
23	0.865528	0.878788
24	0.819458	0.769697
25	0.829544	0.745455
26	0.800867	0.612121
27	0.764609	0.890909
28	0.896245	0.769697
29	0.738119	0.672727
30	0.893021	0.90303
31	0.91068	0.818182
32	0.937707	0.963636
33	0.834563	0.818182
34	0.750709	0.793939
35	0.564428	0.490909
36	0.937426	0.975758
37	0.936632	0.90303
38	0.89141	0.975758
39	0.745184	0.818182
40	0.835685	0.854545
41	0.847526	0.830303
42	0.961247	0.951515
43	0.890352	0.951515
44	0.951611	0.927273
45	0.944342	0.951515
46	0.870518	0.890909
47	0.913673	0.927273
48	0.545525	0.551515
49	0.741032	0.587879
50	0.896819	0.915152

Table 7: Results from the experiment with FSMs with 75 states.

we increase the generality of the functions represented by the FSMs, that is, as we consider bigger FSMs. This allows us to hypothesise that the results corresponding to FSMs will *tend towards* the results from the simulations. Also, as we considered many more scenarios, we can observe that even after taking into account the bad results of the experiment with FSMs between 10 and 25 states, 88% of the cases showed correlations greater than 0.5.

We also performed a small experiment to test what happens if we increase the sample size, that is, to explore the third possibility. We increased the sample size from 10 to 20 samples, using FSMs with 25 states from the initial (FSM) experiments. As expected, we got better correlations, with all the correlations being greater than 0.5. The full results can be found in Table 8.

These results suggest that the correlations will improve if we use larger sample sizes and, in particular, this will reduce the variability of the results. So, again, we can

Run Number	Pearson Sq	Spearman Sq
1	0.827858	0.798496
2	0.895008	0.792481
3	0.58499	0.810526
4	0.707372	0.78797
5	0.837989	0.929323
6	0.885698	0.857143
7	0.76128	0.711278
8	0.84805	0.700752
9	0.767311	0.694737
10	0.880905	0.899248
11	0.722467	0.658647
12	0.834103	0.778947
13	0.880195	0.891729
14	0.874195	0.696241
15	0.57377	0.607519
16	0.736536	0.669173
17	0.733254	0.702256
18	0.846274	0.861654
19	0.765341	0.783459
20	0.784864	0.843609
21	0.678806	0.700752
22	0.914889	0.809023
23	0.813506	0.735338
24	0.742288	0.769925
25	0.668486	0.696241

Table 8: Results from the experiment with sample size of 20 FSMs of 25 states.

expect that the results will *tend towards* the results from the simulations. Overall, our experiments with FSMs indicate that the measures perform well with FSMs and not just with simulated functions.

Finally, we did an experiment to show what happens if we do not assume a uniform distribution over the inputs of the FSM. In order to do so, we considered the same setup of our main experiment with 900 FSMs, but this time we used the Maximum Loss of Information approach explained in section 3.2 for computing Squeeziness.

The results of this experiment showed lower correlations and can be found in Table 9. The results are unsurprising since PColl assumes a uniform distribution over the inputs of the FSM, that is, these two measures used different distributions. However, they are still relatively good, with a mean of 0.646355 for Pearson and of 0.629577 for Spearman. Note that the lower correlations suggest that techniques that use Squeeziness may be most effective when we know the true distribution of values.

To conclude, the results suggest that Squeeziness can be used to estimate the probability of the FEP introduced by a component. As a result, there is potential to use it to direct testing in order to avoid components that have a high probability of FEP. It might also be used as a measure of testability, with the tester potentially choosing to use more test cases in situations in which FEP is particularly likely. It would be interesting to explore this further through additional experiments.

5. Conclusions and future work

It is known that failed error propagation (FEP) can have a significant effect on testing. Recent work has shown that an information theoretic measure called Squeeziness

Run Number	Pearson Sq	Spearman Sq
1	0.710801	0.791365
2	0.749008	0.701602
3	0.624626	0.713539
4	0.670041	0.646195
5	0.642364	0.496048
6	0.452779	0.525309
7	0.463446	0.646934
8	0.525565	0.613348
9	0.703414	0.589675
10	0.878989	0.792566
11	0.830456	0.75203
12	0.651943	0.772191
13	0.776044	0.582555
14	0.714373	0.733645
15	0.865138	0.738564
16	0.648421	0.828883
17	0.684968	0.647903
18	0.695542	0.57537
19	0.684215	0.679355
20	0.531623	0.502392
21	0.485815	0.505174
22	0.707509	0.637446
23	0.613067	0.495327
24	0.534643	0.478087
25	0.703643	0.75153
26	0.35483	0.373817
27	0.78047	0.66548
28	0.617858	0.523026
29	0.517789	0.567646
30	0.571259	0.560303

Table 9: Results from the experiment with Maximum Loss of Information approach.

strongly correlates with the likelihood of FEP [13]. However, this work only considered the white-box scenario in which the SUT simply receives input and returns output; there is no persistent state. In this paper we adapted the Squeeziness measure to work with situations in which we are interested in fault masking. Specifically, we adapted Squeeziness to the scenario in which we are interested in the FEP that a component C introduces when it receives its input from another component C_P. We are interested in this since such FEP makes it more difficult to find faults in C_P when testing. The work also considered the black-box scenario, in which we base the computations on models. This has the advantage that the approach is applicable at an earlier stage (for example, as a notion of testability that can help inform test planning) and also that the approach can be used in situations in which the source code is not available (for example, when development has been outsourced).

It was not possible to directly reuse the previous notion of Squeeziness [13] since we considered a different scenario and also a different source of FEP. In addition, we argued that in our scenario it makes sense to base the analysis on models of components rather that the source code: this should aid scalability and also address the issue that we might not have access to the source code of a component. As a result, we addressed a different type of FEP and also used a different source of information (an `FSM` specification rather than the source code).

Having devised a new notion of Squeeziness, for black-box component-based systems, we carried out experiments in order to evaluate this measure. These experiments focused on the capability of the second component to hide faulty inputs from the first component by giving the expected outputs. In the experiments, we compared our measure with a measure of the probability of this hiding/FEP happening (`PColl`). We used two types of experiments: simulations and experiments with `FSM`s. In both cases, we observed a strong correlation between the likelihood of FEP and our measure (Squeeziness). Interestingly, in the experiments with `FSM`s we observed a slight improvement when we increased the number of states. This supports our original hypothesis: our new notion of Squeeziness can be used as a measure that estimates the probability of FEP being introduced by a component.

The results in this paper have two potential uses. First, the measure defined might be used as a measure of testability, allowing one to assess how easy it is to test a system or part of a system. This might be used as part of the process of deciding how much testing is required. In addition, there is potential to use Squeeziness to direct testing. For example, we might want to execute a part of the system with a test case where the probability of FEP (introduced by another component) is relatively low.

We have several lines for future work. First, we will explore the previously mentioned potential uses, develop tools, and evaluate these on case studies. We plan to explore approximations, most likely based on sampling, and the trade-off between the cost of sampling (sample size) and the effectiveness of the estimates. We also intend to generalise the framework and measures to introduce data into the models. Finally, we would like to adapt Squeeziness to systems with other features. It is natural to consider how Squeeziness works in systems where decisions are probabilistically quantified and we will take as initial step our previous work on formally testing this kind of systems [28, 29]. Similarly, we would like to consider distributed systems and how Squeeziness *predicts* FEP induced by different distributed components. Again, we will take as initial step our work on the distributed test architecture [26, 27].

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.

This work has been supported by the Spanish MINECO-FEDER (grant number DArDOS, TIN2015-65845-C3-1-R); the Region of Madrid (grant number FORTE-CM, S2018/TCS-4314); and the UK EPSRC (grant number InfoTestSS, EP/P006116/2).

References

[1] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. OpenFst: A general and efficient weighted finite-state transducer library. In *9th Int. Conf. on Implementation and Appli-*

cation of Automata, CIAA'07, LNCS 4783, volume 4783, pages 11–23. Springer, 2007.

[2] N. Alshahwan and 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*, pages 181–192. ACM Press, 2014.

[3] P. Ammann and J. Offutt. *Introduction to Software Testing*. Cambridge University Press, 2nd edition, 2017.

[4] K. Androutsopoulos, D. Clark, H. Dan, R.M. Hierons, and 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*, pages 573–583. ACM Press, 2014.

[5] R. Anido, A. R. Cavalli, L. A. Paula Lima Jr., and N. Yevtushenko. Test suite minimization for testing in context. *Software Testing, Verification and Reliability*, 13(3):141–155, 2003.

[6] T. Apiwattanapong, R. A. Santelices, P. K. Chittimalli, A. Orso, and M. J. Harrold. MATRIX: Maintenance-oriented testing requirements identifier and examiner. In *1st Testing: Academia and Industry Conference - Practice And Research Techniques, TAIC PART'06*, pages 137–146. IEEE Computer Society, 2006.

[7] R. V. Binder, B. Legeard, and A. Kramer. Model-based testing: where does it stand? *Communications of the ACM*, 58(2):52–56, 2015.

[8] M. Boreale and M. Paolini. On formally bounding information leakage by statistical estimation. In *17th Int. Conf. on Information Security, ISC'14, LNCS 8783*, pages 216–236. Springer, 2014.

[9] C. Braunstein, A. E. Haxthausen, W.-L. Huang, F. Hübner, J. Peleska, U. Schulze, and L. V. Hong. Complete model-based equivalence class testing for the ETCS ceiling speed monitor. In *16th Int. Conf. on Formal Engineering Methods, ICFEM'14, LNCS 8829*, pages 380–395. Springer, 2014.

[10] A. R. Cavalli, T. Higashino, and M. Núñez. A survey on formal active and passive testing with applications to the cloud. *Annales of Telecommunications*, 70(3-4):85–93, 2015.

[11] T. Chothia, Y. Kawamoto, and C. Novakovic. Leakwatch: Estimating information leakage from java programs. In *19th European Symposium on Research in Computer Security, ESORICS'14, LNCS 8713*, pages 219–236. Springer, 2014.

[12] T. S. Chow. Testing software design modeled by finite state machines. *IEEE Transactions on Software Engineering*, 4:178–187, 1978.

[13] D. Clark and R. M. Hierons. Squeeziness: An information theoretic measure for avoiding fault masking. *Information Processing Letters*, 112(8-9):335–340, 2012.

[14] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Interscience, 1991.

[15] K. El-Fakih, A. Petrenko, and N. Yevtushenko. FSM test translation through context. In *18th Int. Conf. on Testing Communicating Systems, TestCom'06, LNCS 3964*, pages 245–258. Springer, 2006.

[16] R. Feldt, S. M. Poulding, D. Clark, and 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*, pages 223–233. IEEE Computer Society, 2016.

[17] R. Feldt, R. Torkar, T. Gorschek, and W. Afzal. Searching for cognitively diverse tests: Towards universal test diversity metrics. In *1st IEEE Int. Conf. on Software Testing Verification and Validation Workshops*, pages 178–186. IEEE Computer Society, 2008.

[18] M.-C. Gaudel. Testing can be formal, too! In *6th Int. Joint Conf. CAAP/FASE, Theory and Practice of Software Development, TAPSOFT'95, LNCS 915*, pages 82–96. Springer, 1995.

[19] W. Grieskamp, Y. Gurevich, W. Schulte, and M. Veanes. Generating finite state machines from abstract state machines. In *ACM SIGSOFT Symposium on Software Testing and Analysis, ISSTA'02*, pages 112–122. ACM Press, 2002.

[20] W. Grieskamp, N. Kicillof, K. Stobie, and V. Braberman. Model-based quality assurance of protocol documentation: tools and methodology. *Software Testing, Verification and Reliability*, 21(1):55–71, 2011.

[21] Q. Guo, R. M. Hierons, M. Harman, and K. Derderian. Improving test quality using robust unique input/output circuit sequences (UIOCs). *Information & Software Technology*, 48(8):696–707, 2006.

[22] C. Henard, M. Papadakis, M. Harman, Y. Jia, and Y. Le Traon. Comparing white-box and black-box test prioritization. In *38th Int. Conf. on Software Engineering, ICSE'14*, pages 523–534. ACM Press, 2016.

[23] F. C. Hennie. Fault-detecting experiments for sequential circuits. In *5th Annual Symposium on Switching Circuit Theory and Logical Design*, pages 95–110. IEEE Computer Society, 1964.

[24] R. M. Hierons. Testing from partial finite state machines without harmonised traces. *IEEE Transactions on Software Engineering*, 43(11):1033–1043, 2017.

[25] R. M. Hierons, K. Bogdanov, J.P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Luettgen, A.J.H Simons, S. Vilkomir, M.R. Woodward, and H. Zedan. Using formal specifications to support testing. *ACM Computing Surveys*, 41(2):9:1–9:76, 2009.

[26] R. M. Hierons, M. G. Merayo, and M. Núñez. Implementation relations and test generation for systems with distributed interfaces. *Distributed Computing*, 25(1):35–62, 2012.

[27] R. M. Hierons, M. G. Merayo, and M. Núñez. Bounded reordering in the distributed test architecture. *IEEE Transactions on Reliability*, 67(2):522–537, 2018.

[28] R. M. Hierons and M. Núñez. Using schedulers to test probabilistic distributed systems. *Formal Aspects of Computing*, 24(4-6):679–699, 2012.

[29] R. M. Hierons and M. Núñez. Implementation relations and probabilistic schedulers in the distributed test architecture. *Journal of Systems and Software*, 132:319–335, 2017.

[30] I. Hwang and A. R. Cavalli. Testing a probabilistic FSM using interval estimation. *Computer Networks*, 54(7):1108–1125, 2010.

[31] Z. Kohavi. *Switching and Finite State Automata Theory*. McGraw-Hill, 1978.

[32] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines: A survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.

[33] W. Masri, R. Abou-Assi, M. El-Ghali, and N. Al-Fatairi. An empirical study of the factors that reduce the effectiveness of coverage-based fault localization. In *2nd Int. Workshop on Defects in Large Software Systems, DEFECTS'09*, pages 1–5. ACM Press, 2009.

[34] E. P. Moore. Gedanken experiments on sequential machines. In C. Shannon and J. McCarthy, editors, *Automata Studies*. Princeton University Press, 1956.

[35] G. J. Myers, C. Sandler, and T. Badgett. *The Art of Software Testing*. John Wiley & Sons, 3rd edition, 2011.

[36] J. Peleska. Model-based avionic systems testing for the airbus family. In *23rd IEEE European Test Symposium, ETS'18*, pages 1–10. IEEE Computer Society, 2018.

[37] A. Petrenko. Fault model-driven test derivation from finite state models: Annotated bibliography. In *4th Summer School on Modeling and Verification of Parallel Processes, MOVEP'00, LNCS 2067*, pages 196–205. Springer, 2001.

[38] A. Petrenko, S. Boroday, and R. Groz. Confirming configurations in EFSM testing. *IEEE Transactions on Software Engineering*, 30(1):29–42, 2004.

[39] A. Petrenko and N. Yevtushenko. Testing from partial deterministic FSM specifications. *IEEE Transactions on Computers*, 54(9):1154–1165, 2005.

[40] A. Petrenko, N. Yevtushenko, and G. von Bochmann. Testing deterministic implementations from their nondeterministic FSM specifications. In *9th IFIP Workshop on Testing of Communicating Systems, IWTCS'96*, pages 125–140. Chapman & Hall, 1996.

[41] A. Petrenko, N. Yevtushenko, G. von Bochmann, and

R. Dssouli. Testing in context: Framework and test derivation. *Computer Communications*, 19:1236–1249, 1996.
[42] M. Shafique and Y. Labiche. A systematic review of state-based test tools. *International Journal on Software Tools for Technology Transfer*, 17(1):59–76, 2015.
[43] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.
[44] X. Wang, S.-C. Cheung, W. K. Chan, and Z. Zhang. Taming coincidental correctness: Coverage refinement with context patterns to improve fault localization. In *31st Int. Conf. on Software Engineering, ICSE'09*, pages 45–55. IEEE Computer Society, 2009.
[45] Y. Wang, M. Ü. Uyar, S. S. Batth, and M. A. Fecko. Fault masking by multiple timing faults in timed EFSM models. *Computer Networks*, 53(5):596–612, 2009.
[46] M. R. Woodward and Z. A. Al-Khanjari. Testability, fault size and the domain-to-range ratio: An eternal triangle. In *12th Int. Symposium on Software Testing and Analysis, ISSTA'00*, pages 168–172. ACM Press, 2000.

Appendix A. Proofs of the results

Lemma 2 Let $M = (Q, q_{in}, I, O, T)$ be an FSM and $k > 0$. Let us consider two random variables $\xi_{\mathtt{dom}_{M,k}}$ and $\xi_{\mathtt{image}_{M,k}}$ ranging, respectively, over the domain and image of $f_{M,k}$. We have that $\mathcal{H}(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}}) = 0$.

Proof: Consider the entropy of the conditional random variable $\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}}$. We have that $\mathcal{H}(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}})$ is equal to

$$\sum_{\alpha\in\mathtt{dom}_{M,k}} \sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha)\cdot\mathcal{H}(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}}=\alpha)$$

If we unfold the second term of the sum we have that the previous expression is equal to

$$\sum_{\alpha\in\mathtt{dom}_{M,k}} \sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha)\cdot\left(\sum_{\beta\in\mathtt{image}_{M,k}}\gamma(\beta|\alpha)\cdot\log_2(\gamma(\beta|\alpha))\right)$$

where $\gamma(\beta|\alpha) = \sigma_{(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}})}(\beta|\alpha)$. We will prove that all the summands of the previous expression are equal to zero. Taking into account that M is deterministic we have that $\sigma_{(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}})}$ can be either 0 or 1. Using this fact in the previous expression, we have two cases:

- If $\sigma_{(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}})}(\beta|\alpha) = 0$ then the result obviously holds.
- Otherwise, that is, $\sigma_{(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}})}(\beta|\alpha) = 1$, we have that $\log_2(\sigma_{(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}})}(\beta|\alpha)) = 0$ and, again, the result holds.

We finally conclude that $\mathcal{H}(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}}) = 0$. □

Proposition 1 Let $M = (Q, q_{in}, I, O, T)$ be an FSM and $k > 0$. Let us consider two random variables $\xi_{\mathtt{dom}_{M,k}}$ and $\xi_{\mathtt{image}_{M,k}}$ ranging, respectively, over the domain and image of $f_{M,k}$. We have that

$$\mathcal{H}(\xi_{\mathtt{dom}_{M,k}}) = \mathcal{H}(\xi_{\mathtt{image}_{M,k}}) - \mathcal{P}(M, \xi_{\mathtt{image}_{M,k}})$$

where the term $\mathcal{P}(M, \xi_{\mathtt{image}_{M,k}})$ is equal to

$$\sum_{\beta\in\mathtt{image}_{M,k}}\sigma_{\xi_{\mathtt{image}_{M,k}}}(\beta)\cdot\left(\sum_{\alpha\in f_M^{-1}(\beta)}\sigma_{\xi_{f_M^{-1}(\beta)}}(\alpha)\cdot\log_2(\sigma_{\xi_{f_M^{-1}(\beta)}}(\alpha))\right)$$

Proof: By the definition of conditional entropy [14] we have that $\mathcal{H}(\xi_{\mathtt{dom}_{M,k}}|\xi_{\mathtt{image}_{M,k}})$ is equal to

$$\sum_{\beta\in\mathtt{image}_{M,k}}\sigma_{\xi_{\mathtt{image}_{M,k}}}(\beta)\cdot\mathcal{H}(\xi_{\mathtt{dom}_{M,k}}|\xi_{\mathtt{image}_{M,k}}=\beta)$$

Next, we apply the notion of conditional probability and take into account that $\xi_{\mathtt{dom}_{M,k}}$ restricted to $\xi_{\mathtt{image}_{M,k}} = \beta$ is the random variable $\xi_{f_M^{-1}(\beta)}$ ranging over $f_M^{-1}(\beta)$ and whose probabilities are equal to

$$\frac{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(\beta)}{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(f_M^{-1}(\beta))}$$

Therefore, we we have that

$$\begin{aligned}\mathcal{H}(\xi_{\mathtt{dom}_{M,k}}|\xi_{\mathtt{image}_{M,k}}=\beta) &= \\ &= \mathcal{H}(\xi_{f_M^{-1}(\beta)}) \\ &= -\sum_{\alpha\in f_M^{-1}(\beta)}\sigma_{\xi_{f_M^{-1}(\beta)}}(\alpha)\cdot\log_2(\sigma_{\xi_{f_M^{-1}(\beta)}}(\alpha)) \\ &= -\sum_{\alpha\in f_M^{-1}(\beta)}\frac{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha)}{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(f_M^{-1}(\beta))}\cdot\log_2\left(\frac{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(\alpha)}{\sigma_{\xi_{\mathtt{dom}_{M,k}}}(f_M^{-1}(\beta))}\right)\end{aligned}$$

Therefore, the term $\mathcal{H}(\xi_{\mathtt{dom}_{M,k}}|\xi_{\mathtt{image}_{M,k}})$ is equal to

$$-\sum_{\beta\in\mathtt{image}_{M,k}}\sigma_{\xi_{\mathtt{image}_{M,k}}}(\beta)\cdot\left(\sum_{\alpha\in f_M^{-1}(\beta)}\theta(\alpha)\cdot\log_2(\theta(\alpha))\right) \quad \text{(A.1)}$$

where $\theta(\alpha) = \sigma_{\xi_{f_M^{-1}(\beta)}}(\alpha)$. If we apply the *Chain rule* then we have

$$\mathcal{H}(\xi_{\mathtt{image}_{M,k}}, \xi_{\mathtt{dom}_{M,k}}) = \mathcal{H}(\xi_{\mathtt{image}_{M,k}}) + \mathcal{H}(\xi_{\mathtt{dom}_{M,k}}|\xi_{\mathtt{image}_{M,k}})$$

where $\mathcal{H}(\xi_{\mathtt{image}_{M,k}}, \xi_{\mathtt{dom}_{M,k}})$ is the joint probability of the two random variables. Taking into account that, applying again the *Chain rule*, we also have

$$\mathcal{H}(\xi_{\mathtt{image}_{M,k}}, \xi_{\mathtt{dom}_{M,k}}) = \mathcal{H}(\xi_{\mathtt{dom}_{M,k}}) + \mathcal{H}(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}})$$

Combining the previous equalities we obtain

$$\begin{gathered}\mathcal{H}(\xi_{\mathtt{image}_{M,k}}) + \mathcal{H}(\xi_{\mathtt{dom}_{M,k}}|\xi_{\mathtt{image}_{M,k}}) \\ \| \\ \mathcal{H}(\xi_{\mathtt{dom}_{M,k}}) + \mathcal{H}(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}})\end{gathered}$$

Finally, by Lemma 2, we have $\mathcal{H}(\xi_{\mathtt{image}_{M,k}}|\xi_{\mathtt{dom}_{M,k}}) = 0$ and taking into account the value of $\mathcal{H}(\xi_{\mathtt{dom}_{M,k}}|\xi_{\mathtt{image}_{M,k}})$,

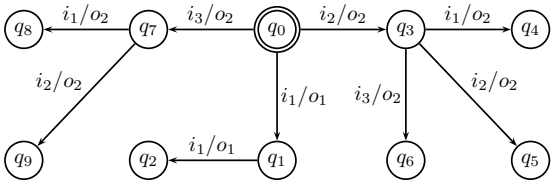
given in equation (A.1), we obtain the desired reformulation of $\mathcal{H}(\xi_{\texttt{dom}_{M,k}})$. □

Lemma 4 There exist FSMs M_1 and M_2 and $k > 0$ such that $\texttt{DRR}(f_{M_1,k}) = \texttt{DRR}(f_{M_2,k})$ but $\texttt{Sq}_k(M_1) \neq \texttt{Sq}_k(M_2)$.

There exist FSMs M_1 and M_2 and $k > 0$ such that $\texttt{DRR}(f_{M_1,k}) < \texttt{DRR}(f_{M_2,k})$ but $\texttt{Sq}_k(M_1) > \texttt{Sq}_k(M_2)$.

Proof: First, let us note that in this proof we assume uniform distributions over inputs (and outputs) of the FSMs. However, the result holds for any probability distribution: we would only need to slightly modify the definition of the given machines.

In order to prove the first part of the result, we define two machines M_1 and M_2, both with initial state q_0, fulfilling the conditions. Let M_1 be the following FSM:

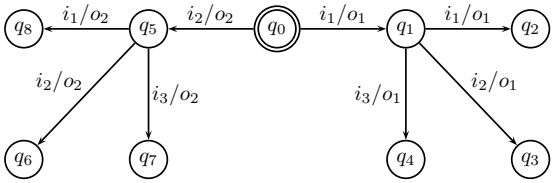


We have that $\texttt{dom}_{M_1,2}$ is equal to

$$\{(i_1,i_1),(i_2,i_1),(i_2,i_2),(i_2,i_3),(i_3,i_1),(i_3,i_2)\}$$

and $\texttt{image}_{M_1,2}$ is equal to $\{(o_1,o_1),(o_2,o_2)\}$. On the one hand we have $\texttt{DRR}(f_{M_1,2}) = 6/2 = 3$ while, on the other hand, we have $\texttt{Sq}_2(M_1) = \frac{5 \cdot log_2(5) + 1 \cdot log_2(1)}{6} \approx 1.9349$.

Now, let M_2 be the following FSM:

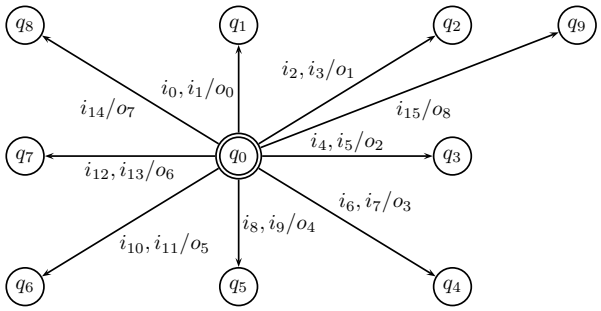


We have that $\texttt{dom}_{M_2,2}$ is equal to

$$\{(i_1,i_1),(i_1,i_2),(i_1,i_3),(i_2,i_1),(i_2,i_2),(i_2,i_3)\}$$

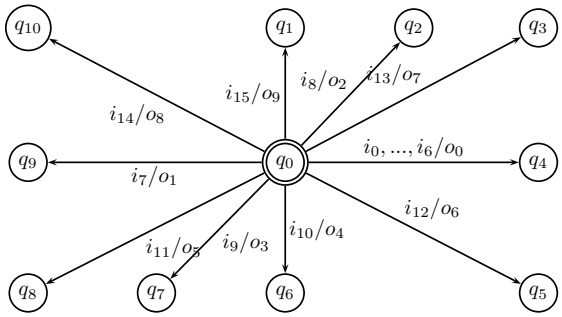
and $\texttt{image}_{M_2,2} = \{(o_1,o_1),(o_2,o_2)\}$. We have, on the one hand, that $\texttt{DRR}(f_{M_2,2}) = 6/2 = 3$ while, on the other hand, $\texttt{Sq}_2(M_2) = \frac{2 \cdot 3 \cdot log_2(3)}{6} \approx 1.5849$.

In order to prove the second part of the result, let us consider again two machines M_1 and M_2, with initial state q_0, and we will show that they fulfil the required conditions. In these machines, we consider that $x_1, \ldots, x_n/y$ is a shorthand for n different transitions labelled, respectively, by $x_1/y, x_2/y, \ldots, x_n/y$. Let M_1 be:



We have that $\texttt{dom}_{M_1,1} = \{i_0, ..., i_{15}\}$ and $\texttt{image}_{M_1,1} = \{o_0, ..., o_8\}$. Therefore, $\texttt{DRR}(f_{M_1,1}) = 16/9 \approx 1.778$ while $\texttt{Sq}_1(M_1) = \frac{7 \cdot 2 \cdot log_2(2) + 2 \cdot 1 \cdot log_2(1)}{16} = 0.875$.

Finally, let M_2 be the FSM:

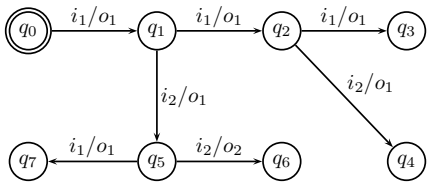


We have that $\texttt{dom}_{M_2,1} = \{i_0, ..., i_{15}\}$ and $\texttt{image}_{M_2,1} = \{o_0, ..., o_9\}$. Therefore, $\texttt{DRR}(f_{M_2,1}) = 16/10 = 1.6$ while $\texttt{Sq}_1(M_2) = \frac{1 \cdot 7 \cdot log_2(7) + 9 \cdot 1 \cdot log_2(1)}{16} \approx 1.2282$. □

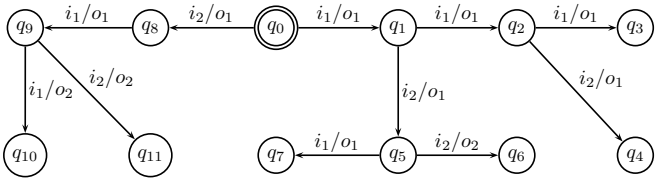
Lemma 5 There exist FSMs M_1 and M_2 and $k > 0$ such that $\texttt{Sq}_k(M_1) < \texttt{Sq}_k(M_2)$ but $\texttt{PColl}_k(M_1) > \texttt{PColl}_k(M_2)$.

Proof: First, let us note again that, similar to the proof of Lemma 4, in this proof we assume uniform distributions over inputs (and outputs) of the FSMs. Again, if we have a different probability distribution then we only need to adapt the definition of the machines so that the result still holds.

First, we consider M_1 with initial state q_0:



Second, let M_2, again with initial state q_0, be:



On the one hand $\texttt{PColl}_3(M_1) = 0.5$ and $\texttt{PColl}_3(M_2) = 0.4$ while, on the other hand, we have $\texttt{Sq}_3(M_1) = 1.1887$ and $\texttt{Sq}_3(M_2) = 1.5849$. □

Appendix B. Simulation Results

Here we show the results for the simulation performed on Section 4.1.

Input set size	Maximum size	Correlation of Sq	Correlation of DRR
10,000	100	0.968366	0.763623
10,000	100	0.973918	0.783759
10,000	200	0.973016	0.823959
10,000	200	0.967349	0.77492
10,000	500	0.973849	0.828911
10,000	500	0.973267	0.803445
10,000	1,000	0.963235	0.744021
10,000	1,000	0.973658	0.804282
10,000	2,000	0.969764	0.787121
10,000	2,000	0.966409	0.753929
10,000	5,000	0.968639	0.778538
10,000	5,000	0.968937	0.768205
10,000	10,000	0.967281	0.71496
10,000	10,000	0.966184	0.670497
20,000	100	0.969669	0.780648
20,000	100	0.975364	0.824959
20,000	200	0.969587	0.778449
20,000	200	0.971707	0.771526
20,000	500	0.971942	0.798243
20,000	500	0.974174	0.792043
20,000	1,000	0.971248	0.786153
20,000	1,000	0.967574	0.769014
20,000	2,000	0.967758	0.770978
20,000	2,000	0.975119	0.819613
20,000	5,000	0.972733	0.823052
20,000	5,000	0.970411	0.780216
20,000	10,000	0.960576	0.728561
20,000	10,000	0.961688	0.724022
50,000	100	0.963278	0.74568
50,000	100	0.975731	0.817716
50,000	200	0.974623	0.795574
50,000	200	0.969418	0.746002
50,000	500	0.966153	0.777624
50,000	500	0.975947	0.84295
50,000	1,000	0.967855	0.76079
50,000	1,000	0.967894	0.789061
50,000	2,000	0.96735	0.764992
50,000	2,000	0.969433	0.804356
50,000	5,000	0.97278	0.797072
50,000	5,000	0.971647	0.792316
50,000	10,000	0.970928	0.779042
50,000	10,000	0.963673	0.723346
100,000	100	0.97475	0.797906
100,000	100	0.972203	0.799384
100,000	200	0.972457	0.788938
100,000	200	0.969988	0.78341
100,000	500	0.980028	0.836659
100,000	500	0.972878	0.769055
100,000	1,000	0.976104	0.817482
100,000	1,000	0.974571	0.820023
100,000	2,000	0.971424	0.779667
100,000	2,000	0.975182	0.787567
100,000	5,000	0.96594	0.762143
100,000	5,000	0.96303	0.73042
100,000	10,000	0.970134	0.757703
100,000	10.000	0.96836	0.778925
200,000	100	0.970841	0.801076
200,000	100	0.974049	0.798232
200,000	200	0.971829	0.776558
200,000	200	0.973847	0.79645
200,000	500	0.978293	0.822944
200,000	500	0.96523	0.748004
200,000	1,000	0.968757	0.768184
200,000	1,000	0.972733	0.808345
200,000	2,000	0.971834	0.798966
200,000	2,000	0.969003	0.749107
200,000	5,000	0.970825	0.760313
200,000	5,000	0.969484	0.76873
200,000	10,000	0.970044	0.792676
200,000	10,000	0.972554	0.788373

Table B.10: First part of the results from the simulation.

Input set size	Maximum size	Correlation of Sq	Correlation of DRR
500,000	100	0.97668	0.836037
500,000	100	0.977493	0.809851
500,000	200	0.963671	0.743951
500,000	200	0.974121	0.807426
500,000	500	0.971647	0.774395
500,000	500	0.973467	0.800447
500,000	1,000	0.976121	0.820915
500,000	1,000	0.97081	0.769445
500,000	2,000	0.976695	0.803875
500,000	2,000	0.973124	0.787502
500,000	5,000	0.95885	0.743651
500,000	5,000	0.969437	0.765643
500,000	10,000	0.971292	0.786862
500,000	10,000	0.975993	0.819747
1,000,000	100	0.976936	0.811867
1,000,000	100	0.971048	0.775681
1,000,000	200	0.970973	0.782711
1,000,000	200	0.977552	0.839242
1,000,000	500	0.972066	0.783899
1,000,000	500	0.974367	0.770392
1,000,000	1,000	0.973926	0.79526
1,000,000	1,000	0.974027	0.830407
1,000,000	2,000	0.969736	0.780849
1,000,000	2,000	0.97408	0.805192
1,000,000	5,000	0.970854	0.809975
1,000,000	5,000	0.970388	0.787131
1,000,000	10,000	0.967924	0.778203
1,000,000	10,000	0.970411	0.769844
2,000,000	100	0.975097	0.814434
2,000,000	100	0.968371	0.768775
2,000,000	200	0.974395	0.809679
2,000,000	200	0.97463	0.800698
2,000,000	500	0.97177	0.790358
2,000,000	500	0.970945	0.809109
2,000,000	1,000	0.978102	0.826712
2,000,000	1,000	0.971722	0.810432
2,000,000	2,000	0.969418	0.755382
2,000,000	2,000	0.970523	0.779241
2,000,000	5,000	0.978818	0.810967
2,000,000	5,000	0.964455	0.698505
2,000,000	10,000	0.96991	0.776906
2,000,000	10,000	0.963563	0.781282
5,000,000	100	0.971105	0.801428
5,000,000	100	0.975811	0.806359
5,000,000	200	0.965705	0.734183
5,000,000	200	0.975194	0.787636
5,000,000	500	0.965762	0.78538
5,000,000	500	0.977868	0.816896
5,000,000	1,000	0.970797	0.782857
5,000,000	1,000	0.974245	0.807752
5,000,000	2,000	0.973636	0.783586
5,000,000	2,000	0.972639	0.782383
5,000,000	5,000	0.977712	0.793327
5,000,000	5,000	0.963994	0.708333
5,000,000	10,000	0.972559	0.773815
5,000,000	10,000	0.975021	0.788634
10,000,000	100	0.972085	0.801643
10,000,000	100	0.96267	0.74051
10,000,000	200	0.973476	0.814127
10,000,000	200	0.978724	0.817254
10,000,000	500	0.968369	0.755809
10,000,000	500	0.976646	0.784194
10,000,000	1,000	0.97411	0.792697
10,000,000	1,000	0.970658	0.782375
10,000,000	2,000	0.973856	0.793005
10,000,000	2,000	0.974945	0.782697
10,000,000	5,000	0.975649	0.814614
10,000,000	5,000	0.9663	0.780145
10,000,000	10,000	0.974921	0.808942
10,000,000	10,000	0.974783	0.821714

Table B.11: Second part of the results from the simulation.

Input set size	Maximum size	Correlation of Sq	Correlation of DRR
20,000,000	100	0.976361	0.816832
20,000,000	100	0.969996	0.785402
20,000,000	200	0.966911	0.773231
20,000,000	200	0.975891	0.830111
20,000,000	500	0.975834	0.80509
20,000,000	500	0.971753	0.761665
20,000,000	1,000	0.970692	0.800126
20,000,000	1,000	0.972765	0.780929
20,000,000	2,000	0.975548	0.79739
20,000,000	2,000	0.97661	0.790627
20,000,000	5,000	0.975512	0.81321
20,000,000	5,000	0.969801	0.778989
20,000,000	10,000	0.97061	0.79285
20,000,000	10,000	0.974807	0.823849
50,000,000	100	0.972157	0.775908
50,000,000	100	0.97394	0.744055
50,000,000	200	0.977712	0.825954
50,000,000	200	0.964124	0.754767
50,000,000	500	0.976058	0.824369
50,000,000	500	0.971696	0.792425
50,000,000	1,000	0.968602	0.773925
50,000,000	1,000	0.975643	0.813831
50,000,000	2,000	0.972101	0.80533
50,000,000	2,000	0.96896	0.763188
50,000,000	5,000	0.967312	0.733459
50,000,000	5,000	0.970914	0.792814
50,000,000	10,000	0.974186	0.831489
50,000,000	10,000	0.97075	0.794533
100,000,000	100	0.967785	0.791843
100,000,000	100	0.973939	0.79906
100,000,000	200	0.970936	0.797435
100,000,000	200	0.971179	0.792618
100,000,000	500	0.965457	0.764338
100,000,000	500	0.967388	0.749111
100,000,000	1,000	0.967278	0.762974
100,000,000	1,000	0.975128	0.816993
100,000,000	2,000	0.976852	0.809661
100,000,000	2,000	0.973916	0.811798
100,000,000	5,000	0.964856	0.752126
100,000,000	5,000	0.975177	0.804654
100,000,000	10,000	0.97333	0.797859
100,000,000	10,000	0.979012	0.839706
200,000,000	100	0.974298	0.793441
200,000,000	100	0.974201	0.817327
200,000,000	200	0.973198	0.79773
200,000,000	200	0.969628	0.752662
200,000,000	500	0.979169	0.843415
200,000,000	500	0.975039	0.830218
200,000,000	1,000	0.975452	0.842656
200,000,000	1,000	0.973656	0.81612
200,000,000	2,000	0.974498	0.799512
200,000,000	2,000	0.980097	0.843219
200,000,000	5,000	0.97596	0.81765
200,000,000	5,000	0.973072	0.794025
200,000,000	10,000	0.972525	0.790124
200,000,000	10,000	0.975228	0.812101
500,000,000	100	0.97099	0.788888
500,000,000	100	0.971083	0.798639
500,000,000	200	0.967438	0.779869
500,000,000	200	0.977179	0.832308
500,000,000	500	0.965965	0.778361
500,000,000	500	0.968144	0.764191
500,000,000	1,000	0.974112	0.800833
500,000,000	1,000	0.973997	0.779971
500,000,000	2,000	0.971501	0.782711
500,000,000	2,000	0.970228	0.743784
500,000,000	5,000	0.976165	0.825479
500,000,000	5,000	0.973031	0.779755
500,000,000	10,000	0.969547	0.772517
500,000,000	10,000	0.966348	0.773234

Table B.12: Third part of the results from the simulation.

Input set size	Maximum size	Correlation of Sq	Correlation of DRR
1,000,000,000	100	0.96974	0.779927
1,000,000,000	100	0.974667	0.824957
1,000,000,000	200	0.978771	0.859822
1,000,000,000	200	0.968844	0.759952
1,000,000,000	500	0.975528	0.799788
1,000,000,000	500	0.972865	0.806221
1,000,000,000	1,000	0.966998	0.742382
1,000,000,000	1,000	0.970395	0.795114
1,000,000,000	2,000	0.96474	0.784384
1,000,000,000	2,000	0.966843	0.768588
1,000,000,000	5,000	0.966975	0.753142
1,000,000,000	5,000	0.969392	0.777797
1,000,000,000	10,000	0.970387	0.78255
1,000,000,000	10,000	0.966483	0.741448
2,000,000,000	100	0.968286	0.797514
2,000,000,000	100	0.974423	0.78976
2,000,000,000	200	0.97463	0.779878
2,000,000,000	200	0.969308	0.776731
2,000,000,000	500	0.97068	0.77233
2,000,000,000	500	0.964814	0.741365
2,000,000,000	1,000	0.977148	0.802956
2,000,000,000	1,000	0.972999	0.824011
2,000,000,000	2,000	0.966897	0.756296
2,000,000,000	2,000	0.967144	0.731439
2,000,000,000	5,000	0.970575	0.807333
2,000,000,000	5,000	0.965495	0.781112
2,000,000,000	10,000	0.969172	0.79843
2,000,000,000	10,000	0.972477	0.783512

Table B.13: Last part of the results from the simulation.