SOFTWARE TESTING, VERIFICATION AND RELIABILITY
Softw. Test. Verif. Reliab. 2007; **17**:41–57
Published online 22 June 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/stvr.352

Heuristics for fault diagnosis when testing from finite state machines

Qiang Guo[1,*,†], Robert M. Hierons[2], Mark Harman[3] and Karnig Derderian[2]

[1]*Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello Street, Sheffield S1 4DP, U.K.*
[2]*School of Information Systems, Computing and Mathematics, Brunel University, Uxbridge, Middlesex UB8 3PH, U.K.*
[3]*Department of Computer Science, King's College London, Strand, London WC2R 2LS, U.K.*

SUMMARY

When testing from finite state machines, a failure observed in the implementation under test (IUT) is called a *symptom*. A symptom could have been caused by an earlier state transfer failure. Transitions that may be used to explain the observed symptoms are called *diagnosing candidates*. Finding strategies to generate an optimal set of diagnosing candidates that could effectively identify faults in the IUT is of great value in reducing the cost of system development and testing. This paper investigates fault diagnosis when testing from finite state machines and proposes heuristics for fault isolation and identification. The proposed heuristics attempt to lead to a symptom being observed in some shorter test sequences, which helps to reduce the cost of fault isolation and identification. The complexity of the proposed method is analysed. A case study is presented, which shows how the proposed approach assists in fault diagnosis. Copyright © 2006 John Wiley & Sons, Ltd.

Received 6 September 2005; Accepted 10 May 2006

KEY WORDS: finite state machines; conformance testing; fault diagnosis; symptom; candidates; conflict sets; minimization

1. INTRODUCTION

Testing is an integral part of system development. Conformance testing aims to check whether the implementation under test (IUT) is functionally equivalent to its specification. Many approaches have been proposed for generating an efficient test from a state-based system. These methods are mostly

*Correspondence to: Qiang Guo, Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello Street, Sheffield S1 4DP, U.K.
†E-mail: q.guo@dcs.shef.ac.uk

based upon traditional finite automata theory and, in particular, they model the system as a finite state machine (FSM) [1–5].

1.1. Background

In FSM-based testing, a standard test strategy tests a transition in two parts: input/output (I/O) check and tail state verification. The former part aims to determine whether a transition of an IUT produces the expected output while the latter checks that the IUT arrives at the specified state when the I/O check is finished. Unique input/output (UIO) sequence, distinguishing sequence (DS) and characterizing set (CS) are three commonly used techniques for state verification. Due to their practical characteristics, UIO-based techniques are often used for test sequence generation [6].

Aho *et al.* [1] showed how an efficient test sequence may be produced using UIOs for state verification. Shen *et al.* [4] extended the method by using multiple UIOs for each state and showed that this leads to a shorter test sequence. Yang and Ural [5] and Miller and Paul [7] showed that overlap can be used in conjunction with (multiple) UIOs to further reduce the test sequence length. Hierons [8,9] represented overlap by invertible sequences.

When testing an FSM, I/O differences exhibited between the IUT and its specification suggest the existence of faults in the implementation. The first observed faulty I/O pair in an observed I/O sequence is called a *symptom*. A symptom could have been caused by either an incorrect output (an *output fault*) or an earlier incorrect state transfer (a *state transfer fault*). Applying strategies to determine the location of faults is therefore important.

Ghedamsi and von Bochmann [10] and Ghedamsi *et al.* [11] generated a set of transitions whose failure could explain the behaviour exhibited. The set is called a *conflict set*. Transitions in the set are called *candidates*. They then produced tests (called distinguishing tests) in order to find the faulty transitions within this set. However, in their approach, the cost of generating a conflict set was not considered. Hierons [12] extended the approach to a special case where a state identification process is known to be correct. Test cost is then analysed by applying statistical methods. As the problem of optimizing the cost of testing is NP-hard [12], heuristic optimization techniques such as Tabu Search (TS) and Hill Climbing (HC) are therefore suggested.

1.2. Motivation

This paper investigates fault diagnosis when testing from FSMs. The work was motivated by the following question. Let ts be a test sequence of length L and suppose that the ith input of ts, $1 \leq i \leq L$, executes a faulty transition tr_f in the IUT. The question is whether it is possible to define the maximum number of inputs that is needed to reveal the failure (a symptom is exhibited) after tr_f is executed. In other words, given a symptom exhibited at the jth input of ts, is it possible to define an interval with a maximum range $d_{\max}$, $d_{\max} \geq 0$, such that inputs between $(j - d_{\max})$th and the jth of ts execute a sequence of transitions that must contain tr_f? If such an interval can be defined, the process of fault isolation is then reduced to a shorter test sequence.

Obviously, the smaller $d_{\max}$ is, the fewer transitions will be considered when isolating the faulty transition. It is always preferred that a symptom is observed immediately after a faulty transition is executed. However, it may require more inputs to exhibit the fault.

DOI: 10.1002/stvr

Clearly, the sequence of transitions executed up to the symptom contains the faulty transition that causes the occurrence of the symptom. However, diagnosing within such a set of candidates might result in a high cost of fault isolation. Finding ways to define the maximum number of inputs that is required to exhibit an executed fault is therefore of great value in minimizing the cost of fault isolation and identification.

In this paper, heuristics are proposed for fault diagnosis, which helps to reduce the cost of fault isolation and identification. In the proposed method, a set of transitions with minimum size is constructed to isolate the faulty transition that could explain an observed symptom. The erroneous final state of the isolated faulty transition is further identified by applying the proposed heuristics. The heuristics defined in this paper consider the use of the U-method [1]. One can easily extend the approach to other formal methods such as the W-method [13] and the Wp-method [14].

The rest of this paper is organized as follows. Section 2 presents the basic definitions and notation of finite state machines and fault diagnosis. Section 3 introduces the detection of a single fault and the construction of a conflict set. Section 4 investigates the minimization of a conflict set. Section 5 studies fault isolation and identification, a case study is carried out in Section 6 and the complexity of the proposed approach is analysed in Section 7. Conclusions are finally drawn in Section 8.

2. PRELIMINARIES

In this section, definitions and notation of finite state machines and fault diagnosis are introduced.

2.1. Finite state machines

A deterministic FSM M is defined as a 6-tuple $M = (I, O, S, \delta, \lambda, s_0)$, where I, O, and S are finite and non-empty sets of input symbols, output symbols and states, respectively, s_0 is the initial state of M, $\delta : S \times I \longrightarrow S$ is the state transition function and $\lambda : S \times I \longrightarrow O$ is the output function. If the machine receives an input $a \in I$ when in state $s \in S$, it moves to the state $\delta(s, a)$ and produces output $\lambda(s, a)$. Functions δ and λ can be extended to take input sequences in the usual way [15].

Two states s_i and s_j are said to be *equivalent* if and only if for every input sequence α the machine produces the same output sequence, $\lambda(s_i, \alpha) = \lambda(s_j, \alpha)$. Machines M_1 and M_2 are *equivalent* if and only if for every state in M_1 there is an equivalent state in M_2, and *vice versa*. A machine M is *minimal* (*reduced*) if and only if no FSM with fewer states than M is equivalent to M. It is assumed that any FSM being considered is minimal since any (deterministic) FSM can be converted into an equivalent (deterministic) minimal FSM [15].

A FSM is *completely specified* if and only if for each state s_i and input a there is a specified next state $s_{i+1} = \delta(s_i, a)$ and a specified output $o_i = \lambda(s_i, a)$; otherwise, the machine is *partially specified*. A partially specified FSM can be converted to a completely specified one in two ways [15]. One way is to define an error state. When a machine is in state s and receives an input a such that there is no transition from s with input a, it moves to the error state with a given (error) output. The other way is to add a loop transition. When receiving an undefined input, the state of a machine remains unchanged. At the same time, the machine produces no output. An FSM is *strongly connected* if, given any ordered pair of states (s_i, s_j), there is a sequence of transitions that moves the FSM from s_i to s_j.

It is assumed throughout this paper that an FSM is deterministic, minimal, completely specified and strongly connected.

DOI: 10.1002/stvr

2.2. Conformance testing

Given a specification FSM M, for which the complete transition diagram is available, and an implementation M', for which only its I/O behaviour can be observed ('black box'), a test is required to determine whether the I/O behaviour of M' conforms to that of M. This is called *conformance testing*. A test sequence that solves this problem is called a *checking sequence*. An I/O difference between the specification and implementation can be caused by either an incorrect output (an output fault) or an earlier incorrect state transfer (a state transfer fault). The latter can be detected by adding a final state check after a transition. A standard test strategy is:

1. homing: move M' to an initial state s;
2. output check: apply an input sequence α and compare the output sequences generated by M and M' separately;
3. tail state verification: use state verification techniques to check the final state.

The first step is known as homing a machine to a desired initial state. The second step checks whether M' produces the desired output sequence. The last step checks whether M' is in the expected state $s' = \delta(s, \alpha)$ after the transition. UIO sequence, DS and CS are three commonly used techniques for state verification.

A UIO sequence of a state s_i is an I/O sequence x/y, that may be observed from s_i, such that the output sequence produced by the machine in response to x from any other state is different from y, i.e. $\lambda(s_i, x) = y$ and $\lambda(s_i, x) \neq \lambda(s_j, x)$ for any $i \neq j$. A DS defines a UIO for every state. While not every FSM has a UIO for each state, some FSMs without a DS have a UIO for each state.

A distinguishing sequence is an input sequence that produces a different output for each state. Not every FSM has a DS.

A characterizing set W is a set of input sequences with the property that, for every pair of states (s_i, s_j), $j \neq i$, there is some $w \in W$ such that $\lambda(s_i, w) \neq \lambda(s_j, w)$. Thus, the output sequences produced by executing each $w \in W$ from s_j verify s_j.

UIOs are often used for test generation. However, due to the problem of fault masking in UIOs, a UIO sequence generated from the specification FSM might not be able to uniquely identify the corresponding state in the IUT [2,16]. Guo *et al.* [17] studied the problem and proposed a new type of Unique Input/Output Circuit (UIOC) sequence for state verification, which helps to increase the robustness of UIOs.

UIOCs are particular types of UIOs where the final states are the same as their initial states. Guo *et al.* [17] formalized the fault masking into two basic types and proposed solutions to overcome them. By checking the final state s of a UIO with the UIO for s, type I of fault masking discussed by Guo *et al.* might be avoided; by using overlap or internal state observation schema, the internal states that a UIO sequence traverses are checked, which helps to overcome type II of fault masking discussed by Guo *et al*.

Two advantages can be noted when UIOCs are applied. Firstly, robustness of UIOs is enhanced, which makes the test more robust, and, secondly, the use of UIOs for test generation can simplify the test structure, which reduces the complexity of fault analysis.

It is assumed throughout this paper that any test sequence considered has been generated by using a standard test strategy. This work considers the use of UIOCs for state verification.

DOI: 10.1002/stvr

2.3. Fault diagnosis

In testing, a test sequence *ts* is applied to an IUT M'. The observed I/O sequence is then compared with that produced by the specification FSM M. I/O differences exhibited between M and M' indicate the existence of faults in M'. The first observed faulty I/O pair in an observed I/O sequence is called a *symptom*. A symptom could have been caused by either an incorrect output (an *output fault*) produced from the transition being tested in M' or an earlier incorrect state transfer (a *state transfer fault*) that occurs in a transition that has already been executed. It is, therefore, important to define strategies to guide the construction of test sequences. These test sequences could be used to (effectively) isolate the faulty transitions in M' that explain the symptoms exhibited.

The process of isolating faults from the IUT with regard to the symptoms observed in an I/O sequence is called *fault diagnosis* [6].

3. ISOLATING A SINGLE FAULT

This section introduces an approach for detecting a single fault in the IUT and the construction of a conflict set for fault diagnosis.

3.1. Detecting a single fault

When testing an IUT, a set of tests $TC = \{tc_1, tc_2, \ldots, tc_l\}$ needs to be developed. A test tc_i consists of a sequence of expected transitions $\langle t_{i,1}, t_{i,2}, \ldots, t_{i,n_i}\rangle$, starting at s_0, with input $\langle x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}\rangle$ and the expected output $\langle y_{i,1}, y_{i,2}, \ldots, y_{i,n_i}\rangle$, where y_{i,n_i} is the expected output after input x_{i,n_i}. When executed, tc_i produces the observed output $\langle z_{i,1}, z_{i,2}, \ldots, z_{i,n_i}\rangle$. If differences between $y_i = \langle y_{i,1}, y_{i,2}, \ldots, y_{i,n_i}\rangle$ and $z_i = \langle z_{i,1}, z_{i,2}, \ldots, z_{i,n_i}\rangle$ appear, there must exist at least one faulty transition in the implementation. The first difference exhibited between y_i and z_i is called a *symptom*. Additional tests are necessary in order to isolate the faulty transitions that cause the observed symptom.

3.2. Generating conflict sets

A conflict set is a set of transitions, each of which could be used to explain a symptom exhibited. Here, the work focuses on identifying the faulty transition that is responsible for the first exhibited symptom. The transitions after the symptom are ignored.

Suppose, for a test tc_i, the sequence of expected transitions is $\langle t_{i,1}, t_{i,2}, \ldots, t_{i,n_i}\rangle$ where n_i is the number of transitions. When executed with $\langle x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}\rangle$, a symptom occurs at the input $x_{i,j}$. The conflict set of the maximum size is $\{t_{i,1}, t_{i,2}, \ldots, t_{i,l}\}$ where $1 \leq l \leq n_i$.

4. MINIMIZING THE SIZE OF A CONFLICT SET

If the number of transitions in a conflict set is large, the effort required to isolate the fault could be high. It is therefore useful to reduce the size of a conflict set. Two abstract schema are applied in this paper:

DOI: 10.1002/stvr

1. transition removals using transfer sequences;
2. transition removals using repeated states.

In the first schema, a short transfer sequence is used to remove a segment of inputs from the original test sequence; this is intended to lead to a symptom being observed in a shorter test sequence. In the second schema, a segment of inputs is further removed from the original test sequence. These inputs execute a sequence of transitions where the initial state of the first transition is the final state of the last transition. By such an operation, a symptom might be observed in a shorter test sequence, which helps to reduce the cost of fault isolation. The two removal schema are discussed in the following subsections.

4.1. Estimating a fault location

Once a symptom is observed, the set of transitions executed up to the symptom constitutes a conflict set S_{conflict} with the maximum size. A subset of transitions $S_r \subset S_{\text{conflict}}$ might be removed to reduce the size of S_{conflict} by applying some transfer sequences. Before explaining this in detail, some concepts are defined.

Definition 1. A UIO sequence generated from the specification FSM is a *strong UIO* if it can identify the corresponding state in the IUT; otherwise, it is a *weak UIO*.

Due to the problem of fault masking in UIOs, a UIO sequence generated from the specification FSM might lose its property of uniqueness and fail to identify its corresponding state in the IUT [2,16].

Definition 2. When testing an IUT, if the UIOs used for the generation of a test sequence are all strong UIOs, the test is a *strong test* and the test sequence is a *strong test sequence*; otherwise, the test is a *weak test* and the test sequence is a *weak test sequence*.

Definition 3. In a UIO-based test, if there are k weak UIOs in the test sequence, the test is called a *k-degree weak test* and the test sequence is a *k-degree weak test sequence*.

It can be seen that a strong test is a *0-degree* weak test.

Definition 4. Let $[a, b]$ be the interval of transitions between the ath and the bth inputs from an input sequence α. A transition tr is said to be *within* $[a, b]$ *of* α if the cth input executes tr when α is applied to the FSM for some $a \leq c \leq b$.

In FSM-based testing, a complete test sequence should test all transitions in the FSM M. A transition is tested by checking its I/O behaviour plus the tail state verification. Once a transition test is finished, M arrives at a state s. If s is not the initial state s' of the transition selected for the following test, a transfer sequence is required to move M to s'. This transfer sequence constitutes a linking sequence in the final test sequence.

Definition 5. A *linking sequence* in a test sequence for an FSM M is a transfer sequence that moves M to the initial state of a transition under test after the previous transition test is finished.

Proposition 1. *In a UIO-based test, if the test is a strong test and a symptom is observed at the ath input, then the faulty transition that causes the occurrence of the symptom must be within $[(a - L_{\text{UIO(max)}} - L_{\text{Link(max)}}), a]$ of the inputs where $L_{\text{UIO(max)}}$ is the maximum length of UIOs and $L_{\text{Link(max)}}$ the maximum length of linking sequences.*

DOI: 10.1002/stvr

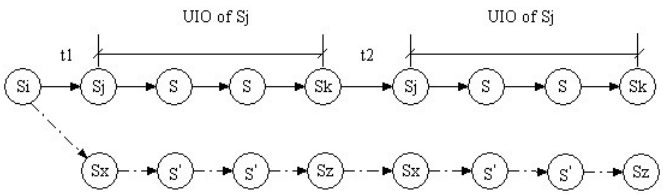

Figure 1. Fault-masked UIO cycling.

Proof. The standard strategy of a transition test in a UIO-based test is formed by a transition I/O test and tail state verification. Since the test is a strong test, no problem of fault masking exists in the test sequence. Suppose a faulty transition is executed at the bth input and the fault is unveiled with an observable symptom at the ath input. If the transition has an I/O error, the fault is detected by the bth input ($a = b$); otherwise, if the transition is one under test, the faulty final state is detected by the following state verification with maximum length $L_{\mathrm{UIO(max)}}$ ($b \leq a \leq b + L_{\mathrm{UIO(max)}}$), or, if the faulty transition is an element of a linking sequence, the following inputs move the machine to the initial state of the next transition under test with the maximum steps of $L_{\mathrm{Link(max)}}$. After executing the transition with one input, the faulty final state can be detected by the forthcoming state verification with the maximum steps of $L_{\mathrm{UIO(max)}}$ ($b \leq a \leq b + L_{\mathrm{Link(max)}} + 1 + L_{\mathrm{UIO(max)}}$). Therefore, if a symptom is observed by a strong test sequence at the ath input, the faulty transition that caused the occurrence of the symptom must be within $[(a - L_{\mathrm{UIO(max)}} - L_{\mathrm{Link(max)}}), a]$ of the inputs. □

Definition 6. In a weak test, if a UIO sequence fails to identify the corresponding state in the IUT more than once, the problem is called *fault-masked UIO cycling*.

An example of fault-masked UIO cycling is illustrated in Figure 1 where a state transfer error occurs in $t_1(s_i \rightarrow s_j)$ first, leading to an erroneous final state s_x. Owing to fault masking, the UIO of s_j fails to find the error, moving the FSM to s_z. Suppose, according to the test order, $t_2(s_k \rightarrow s_j)$ is tested after t_1. When responding to the input, the IUT produces the same output as defined in the specification and arrives at s_x. When applying the UIO of s_j, it again fails to find the fault. The UIO of s_j appears in the test twice, in both cases, failing to exhibit an incorrect final state in the observed I/O sequence.

Proposition 2. *In a k-degree weak test, if the problem of fault-masked UIO cycling does not exist and a symptom is observed at the ath input, then the faulty transition that causes the occurrence of the symptom must be within $[(a + 1 - (k + 1) \times (L_{\mathrm{UIO(max)}} + L_{\mathrm{Link(max)}} + 1)), a]$ of the inputs where $L_{\mathrm{UIO(max)}}$ is the maximum length of UIOs and $L_{\mathrm{Link(max)}}$ the maximum length of linking sequences.*

Proof. In a similar way to Proposition 1, a proof can be obtained by considering the test structure. Since no problem of fault-masked UIO cycling exists in the test, if there are k weak UIOs in the test, the faulty final state of a faulty transition can be detected with the maximum steps of $(k + 1) \times (L_{\mathrm{UIO(max)}} + L_{\mathrm{Link(max)}} + 1)$. □

Testing can be simplified if UIOC sequences are applied for state verification. When UIOCs are used, no linking sequence is required, namely, $L_{\mathrm{Link(max)}} = 0$.

DOI: 10.1002/stvr

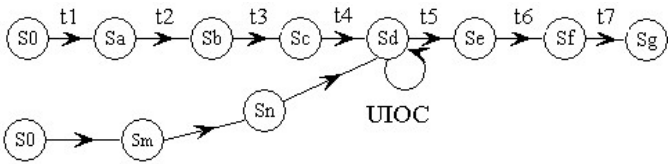

Figure 2. Reducing the size of a conflict set by applying a transfer sequence.

4.2. Reducing the size of a conflict set using transfer sequences

4.2.1. Making a hypothesis

Once a conflict set S_{conflict} is defined, it can be refined. A subset of transitions S_r in S_{conflict} can be removed according to Propositions 1 and 2. Figure 2 demonstrates a paradigm. Let $L_{\text{UIO(max)}} = 2$. Suppose a symptom is observed at the ith input where it executes the transition t_7 ($s_f \rightarrow s_g$). The conflict set with the maximum size is then $S_{\text{conflict}} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. If the test is a strong test, the faulty transition must be within $[(i-2), i]$ of the inputs, namely, it must be in the subset of transitions $S_f = \{t_5, t_6, t_7\}$ where $S_{\text{conflict}} = S_f \cup S_r$ and $S_r = \{t_1, t_2, t_3, t_4\}$.

4.2.2. Verifying the hypothesis

To verify the hypothesis, a new test sequence is constructed by concatenating a shortest transfer sequence with the inputs that execute t_5, t_6 and t_7 from the original test. The transfer sequence moves the FSM from s_0 to s_d, removing S_r from S_{conflict}. In order to increase confidence that the IUT arrives at an expected final state, the final state is verified by its UIOC sequence.

When the new test sequence is applied to the system, two observations need to be made: (1) have any failures been observed from applying the transfer sequence in the new test sequence? (2) if no failure is exhibited by the transfer sequence, the I/O pairs observed afterwards in the new test sequence need to be compared to those observed after s_d in the original test to check if there exist any differences. If a failure is observed by applying the transfer sequence, transitions executed by the transfer sequence constitute a new conflict set S'_{conflict} and additional tests need to be developed to isolate the fault. Since the transfer sequence traverses the shortest path from s_0 to s_d, $|S'_{\text{conflict}}| \leq |S_{\text{conflict}}|$.

Let tr'_f be the faulty transition that is identified in S'_{conflict}. If $tr'_f \in S_{\text{conflict}}$, tr'_f is defined as the principal faulty transition that causes the occurrence of the observed symptom in the original test. The process of isolating the faulty transition for the observed symptom is then complete. More faults might exist in S_{conflict}, and these faults can be isolated by constructing some new test sequences where tr'_f is not executed or is executed as late as possible; otherwise, if $tr'_f \notin S_{\text{conflict}}$, one more fault is detected. tr'_f needs to be further processed as described in Section 5.2. Meanwhile, a new transfer sequence needs to be constructed until no failure is exhibited by this sequence.

Suppose, after applying the transfer sequence, no I/O change is found when the sequence of transitions in S_f is executed; this provides evidence that both the transfer sequence and the input

DOI: 10.1002/stvr

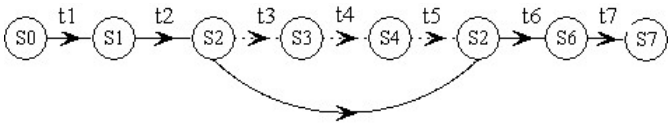

Figure 3. Reducing the size of a conflict set by considering the repeated states.

sequence that executes S_r in the original test make the FSM arrive at the same state s_{com}. Since the final state of the transfer sequence is verified by its UIOC sequence, evidence that $s_{\text{com}} = s_d$ is provided as well. This further suggests that S_f contains the faulty transition that causes the symptom. Additional tests need to be developed to identify the faulty transition.

Having tested all transitions in S_{conflict}, if no faulty transition is defined, this implies that at least one UIO fails to identify the corresponding state. The test is a weak test. Proposition 2 can be applied to estimate the input interval in which the faulty transition might fall. The process starts by considering $[(i + 1 - (k + 1) \times (L_{\text{UIO(max)}} + L_{\text{Link(max)}} + 1)), i]|_{k=1}$ first. By removing a set of transitions, if the faulty transition is still not isolated, k is increased by 1. The process is repeated until the faulty transition is isolated.

The above considers the situation where no problem of fault-masked UIO cycling exists in a test sequence. The existence of such a problem in a test makes the estimation of fault location harder. Fault maskings can be caused either by two different faulty UIOs or a cycled faulty UIO as shown in Figure 1. To simplify the estimation, here, a cycled faulty UIO is treated as two or more independent faulty UIOs depending on the number of times this UIO reoccurs. For example, in Figure 1, two faulty UIOs are counted for the computation (UIO of s_j appears twice). By such an operation, a *k-degree* weak test becomes a $(k + c)$-*degree* weak test where c is the sum of times that the cycled faulty UIOs reoccur.

4.3. Reducing the size of a conflict set using repeated states

In a conflict set S_{conflict}, a state that is the initial state of a transition $tr_a \in S_{\text{conflict}}$ may also be the final state of another transition $tr_b \in S_{\text{conflict}}$ where tr_b is executed after tr_a. This leads to the repetition of a state when the sequence of transitions in S_{conflict} is executed successively. Transitions between repeated states can be removed to check whether they are responsible for the symptom. Figure 3 illustrates the removal schema.

In the figure, $S_{\text{conflict}} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. It can be noted that s_2 appears twice. A subset of transitions between the repeated state is defined as $S_{\text{cycle}} = \{t_3, t_4, t_5\}$. S_{conflict} is then split into two subsets S_{cycle} and $S_{\text{remain}} = \{t_1, t_2, t_6, t_7\}$ so that $S_{\text{conflict}} = S_{\text{cycle}} \cup S_{\text{remain}}$. Based on the original test sequence, a new test sequence is constructed removing the inputs that execute S_{cycle}.

Applying the new test sequence to the system, if, when compared to the original test, the rest of the I/O behaviour remains unchanged, the symptom is then observed in a shorter sequence. The conflict set is consequently reduced to S_{remain}. Additional tests can then be devised to verify the hypothesis.

DOI: 10.1002/stvr

If, compared to the corresponding I/O segment in the original test sequence, the new test sequence behaves differently, no conclusion can be drawn and, in this situation, the removal schema of using repeated states does not reduce the size of S_{conflict}.

5. IDENTIFYING THE FAULTY TRANSITION

Having reduced the size of a conflict set, further tests need to be devised to identify the fault. Here, the process intends not only to locate the faulty transition, but also to determine its faulty final state.

5.1. Isolating the faulty transition

After a conflict set S_{conflict} has been minimized, in order to locate the faulty transition, transitions in S_{conflict} need to be tested individually. Each transition $tr_i \in S_{\text{conflict}}$ is tested by moving the FSM to the head state of tr_i, executing tr_i and then verifying tr_i's tail state. In order to increase the reliability, this process should avoid using other untested candidates in S_{conflict}.

If, when testing a transition $tr_i \in S_{\text{conflict}}$, the use of another untested candidate $tr_j \in S_{\text{conflict}}$ is inevitable, one might verify tr_j's tail state as well when it is executed and then apply a transfer sequence to move the IUT back to the tail state of tr_j. If there exists a UIOC sequence for the tail state of tr_j, the UIOC sequence can be applied. Through such an operation, two transitions are tested simultaneously.

The test process described above assumes that the UIOs or UIOCs used for state verification are strong UIOs. This, however, might not be true. In order to increase test confidence, one might use a set of test sequences to test a transition $tr_i \in S_{\text{conflict}}$, each of which uses a different UIO sequence to verify the final state of tr_i. This, however, requires more test effort.

5.2. Identifying the faulty final state

Once a faulty transition has been located, the faulty final state needs to be identified. This helps to reduce the fault correction effort. A set of estimated erroneous final states S_{EndState} is then constructed.

Let n be the number of states in the FSM. Suppose transition $tr_f : s_i \rightarrow s_j$ is identified as being faulty. It can be noted that, in terms of the alternative final states for tr_f, there are $n-1$ possible mutants. Therefore, S_{EndState} with the maximum size is $S_{\text{EndState}} = \{s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_n\}$ and $|S_{\text{EndState}}| = n-1$.

The size of S_{EndState} might be reduced by comparing the I/O behaviour exhibited after the faulty transition in the IUT to that defined in the specification.

Definition 7. Let $M_S = (I, O, S, \delta, \lambda, s_0)$ be a specification FSM and $M_I = (I, O, S, \delta', \lambda', s_0)$ be an implementation FSM of M_S. Let $x_v \in I$ be an input that executes a transition tr from s_a to s_b in M_I and z_v be the observed output, $z_v = \lambda'(s_a, x_v)$. $S_{\text{VDIS}} \subseteq S$ is called the *valid defined initial state (VDIS) set of* x_v/z_v if $z_v \in O$, for all $s_v \in S_{\text{VDIS}}$, $\lambda(s_v, x_v) = z_v$ and for all $s_v \notin S_{\text{VDIS}}$, $\lambda(s_v, x_v) \neq z_v$.

Let x_v be the input that executes the transition following the isolated faulty transition tr_f in the IUT and z_v be the observed output. The set S_{VDIS} of x_v/z_v is constructed by applying x_v to each state in the specification FSM and comparing the corresponding output with z_v. Let $y_{v(i)}$ be the response from the specification FSM when the machine is in $s_i \in S$ and receives x_v. By comparing $y_{v(i)}$ to z_v, S can

DOI: 10.1002/stvr

be divided into two subsets S_{VDIS} and $\bar{S}_{\text{VDIS}}$ where for all $s_j \in S_{\text{VDIS}}$, $\lambda(s_j, x_v) = y_{v_{(j)}} = z_v$ and for all $s_k \in \bar{S}_{\text{VDIS}}$, $\lambda(s_k, x_v) = y_{v_{(k)}} \neq z_v$. If $\bar{S}_{\text{VDIS}} \neq \emptyset$, this indicates that there exists a non-empty set of states $\bar{S}_{\text{VDIS}}$ such that for all $s_k \in \bar{S}_{\text{VDIS}}$, $\lambda(s_k, x_v) \neq z_v$, which suggests that the erroneous final state of tr_f is less likely to be in $\bar{S}_{\text{VDIS}}$. S_{EndState} is then reduced to S_{VDIS}.

The size of the estimated faulty final state set might be further reduced by using a set of faulty final state identification test sequences.

Definition 8. Let $I = \{a_1, \ldots, a_k\}$ be the input set of a specification FSM M_S and M_I be an IUT of M_S. Let an isolated faulty transition tr_f of M_I be executed by a test sequence $tv = x_1, \ldots, x_v$ at the vth input x_v. $TS = \{ts_1, \ldots, ts_k\}$ is called the *set of faulty final state identification test sequences (FFSITSs) of* tr_f where[‡] $ts_l = tv \cdot a_l$, $a_l \in I$.

For each $ts_j \in TS$, a set S_{VDIS} can be constructed when a_j is applied to the IUT, denoted by S^j_{VDIS}. The final estimated faulty final state set S_{EndState} can then be reduced to $S_{\text{EndState}} = S^1_{\text{VDIS}} \cap \cdots \cap S^k_{\text{VDIS}}$.

The complexity of faulty final state identification is determined by the number of states in S_{EndState}. This is discussed in Section 7.2. If the size of S_{EndState} is reduced, the effort involved in identifying the faulty final state is thus reduced.

Once the size of S_{EndState} is reduced, each state $s_i \in S_{\text{EndState}}$ needs to be tested to identify the faulty final state. State s_i is checked by moving the IUT from s_0 to s_i with a transfer sequence $Seq_{(\text{transfer})}$, and then applying UIO_{s_i} for s_i. In order to increase test confidence, a set of test sequences, $TV = \{tv_1, tv_2, \ldots, tv_r\}$, can be applied where $tv_i \in TV$ is constructed by concatenating $Seq_{(\text{transfer})}$ with a different UIO sequence for s_i.

6. A CASE STUDY

A case study is designed to evaluate the effectiveness of the proposed method. A reduced, completely specified and strongly connected specification FSM M is defined in Table I where the machine has five states. The input set is $I = \{a, b, c, d\}$ and the output set is $O = \{x, y\}$. In order to simplify the analysis, a set of UIOCs (shown in Table II) is used for state verification. For each state, the first UIOC sequence is used for the generation of the test sequence. The rest of the UIOCs are used to verify hypotheses when diagnosing faults. The maximum length[§] of UIOCs, $L_{\text{UIO(max)}}$, is 4. In the implementation M', two faults are injected. They are listed in Table III.

Based upon the rural Chinese postman algorithm and UIOCs for state verification, a test sequence ts is generated from M; ts is then applied to M' for fault detection.

After ts is applied to M', a symptom is observed at the 17th input where, according to M, $t_8 : s_1 \xrightarrow{b/x} s_4$ should have been executed (see Figure 4). The sequence of transitions, $\langle t_1, t_8, t_{19}, t_{10}, t_6, t_9, t_1, t_6, t_{11}, t_{14}, t_8, t_{20}, t_{16}, t_{17}, t_2, t_{10}, t_8\rangle$, executed by the first 17 inputs constitutes the conflict set of the maximum size, this being $S_{\text{conflict}} = \{t_1, t_8, t_{19}, t_{10}, t_6, t_9, t_{11}, t_{14}, t_{20}, t_{16}, t_{17}, t_2\}$.

[‡]The notation '·' implies the concatenation of two sequences.

[§]Here, $L_{\text{UIO(max)}}$ refers to the maximum length of UIOCs that are used for the test generation.

DOI: 10.1002/stvr

Table I. The specification finite state machine used in the case study.

Number	Transition	Number	Transition
t_1	$s_0 \xrightarrow{a/x} s_1$	t_{11}	$s_2 \xrightarrow{c/x} s_3$
t_2	$s_0 \xrightarrow{d/y} s_2$	t_{12}	$s_2 \xrightarrow{b/y} s_4$
t_3	$s_0 \xrightarrow{c/x} s_3$	t_{13}	$s_3 \xrightarrow{a/x} s_0$
t_4	$s_0 \xrightarrow{b/y} s_4$	t_{14}	$s_3 \xrightarrow{b/y} s_1$
t_5	$s_1 \xrightarrow{d/y} s_0$	t_{15}	$s_3 \xrightarrow{c/x} s_2$
t_6	$s_1 \xrightarrow{a/y} s_2$	t_{16}	$s_3 \xrightarrow{d/y} s_4$
t_7	$s_1 \xrightarrow{c/x} s_3$	t_{17}	$s_4 \xrightarrow{b/x} s_0$
t_8	$s_1 \xrightarrow{b/x} s_4$	t_{18}	$s_4 \xrightarrow{d/y} s_1$
t_9	$s_2 \xrightarrow{d/x} s_0$	t_{19}	$s_4 \xrightarrow{c/x} s_2$
t_{10}	$s_2 \xrightarrow{a/y} s_1$	t_{20}	$s_4 \xrightarrow{a/y} s_3$

Table II. Unique I/O circuit sequences for each state of the finite state machine shown in Table I.

State	UIOC sequence
s_0	dd/yx
	$daad/yyyx$
s_1	bca/xxy
	$baca/xyxy$
s_2	daa/xxy
	$dadd/xxyy$
s_3	bba/yxy
	$dada/yyyy$
s_4	$bdab/xyyx$
	$aaab/yxxx$

The size of S_{conflict} is then reduced by applying the proposed heuristics. At first it is assumed that ts is a strong test sequence. The removal schema is then determined by Proposition 1. As $L_{\text{Link(max)}} = 0$ and $L_{\text{UIO(max)}} = 4$, according to Proposition 1, the faulty transition that causes this symptom must be within the set of transitions that are executed by those inputs between the 13th and the 17th. This hypothesis reduces S_{conflict} to $\{t_{16}, t_{17}, t_2, t_{10}, t_8\}$.

To verify the hypothesis, a shortest transfer sequence, c/x, is applied to move M' from s_0 to s_3, removing the inputs in the original test sequence that successively execute $\langle t_1, t_8, t_{19}, t_{10}, t_6,$

DOI: 10.1002/stvr

Table III. Injected faults.

Number	Transition	Mutant
t_3	$s_0 \xrightarrow{c/x} s_3$	$s_0 \xrightarrow{c/x} s_0$
t_{17}	$s_4 \xrightarrow{b/x} s_0$	$s_4 \xrightarrow{b/x} s_4$

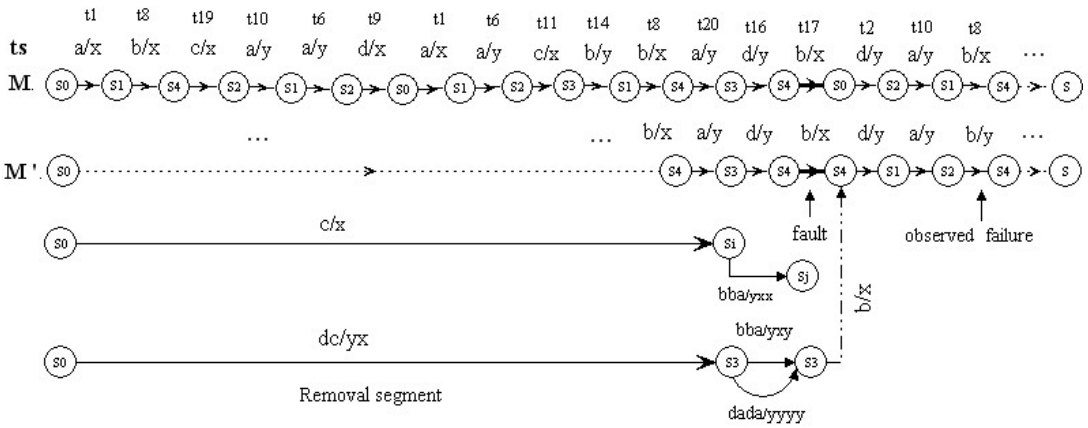

Figure 4. Fault detection and identification in M'.

$t_9, t_1, t_6, t_{11}, t_{14}, t_8, t_{20}\rangle$. The final state of the transfer sequence s_3 is then verified by its UIOC sequence. In order to increase test confidence, two UIOC sequences $dada/yyyy$ and bba/yxy for s_3 are applied. After applying $cbba$ and $cdada$ to M', $xyxy$ and $xyyyx$ ($xyyyx \neq xyyyy$) are received respectively. These results imply that (1) t_3 is faulty—it is detected by $dada/yyyy$ but masked by bba/yxy; or (2) t_3 is correctly implemented but $dada/yyyy$ traverses a faulty transition, leading to a failure being observed.

To further check the hypothesis, two additional tests $tv_1 = (c/x) \cdot (aaba/xxxy)$ and $tv_2 = (c/x) \cdot (abdba/xyyxy)$ are devised where $aaba/xxxy$ and $abdba/xyyxy$ are two different UIOC sequences for s_3. After applying $caaba$ and $cabdba$ to M', $xxyyy$ and $xxxyxy$ are received; $xxyyy \neq xxxxy$ and $xxxyxy \neq xxyyxy$, which suggests t_3 is faulty and bba/yxy is a weak UIO for s_3.

The erroneous final state of t_3 is further identified as described in Section 5. A set of estimated faulty final states for t_3 is constructed by applying a set of faulty final state identification test sequences to M', each test sequence in the set being used to construct the corresponding S_{VDIS}.

Let $S_{\text{VDIS}}^{g/h}$ be the S_{VDIS} of g/h, where g/h indicates that, when applying g to M', h is observed. After all elements in the input set have been applied, a set of S_{VDIS} can then be obtained.

DOI: 10.1002/stvr

Table IV. Tests devised to check *VT*.

Transition	Test set
t_{16}	$vt_{t_{16}} = \{(ac/xx) \cdot (d/y) \cdot (bdab/xyyx), (ac/xx) \cdot (d/y) \cdot (aaab/yxxx)\}$
t_{17}	$vt_{t_{17}} = \{(b/y) \cdot (b/x) \cdot (dd/yx), (b/y) \cdot (b/x) \cdot (daad/yyyx)\}$
t_2	$vt_{t_2} = \{(a/x) \cdot (c/x) \cdot (daa/xxy), (a/x) \cdot (c/x) \cdot (dadd/xxyy)\}$
t_{10}	$vt_{t_{10}} = \{(d/y) \cdot (a/x) \cdot (bca/xxy), (d/y) \cdot (a/x) \cdot (baca/xyxy)\}$
t_8	$vt_{t_8} = \{(a/x) \cdot (b/x) \cdot (bdab/xyyx), (a/x) \cdot (b/x) \cdot (aaab/yxxx)\}$

The elements in the set are $S_{\text{VDIS}}^{a/x} = \{s_0, s_3\}$, $S_{\text{VDIS}}^{b/y} = \{s_0, s_2, s_3\}$, $S_{\text{VDIS}}^{c/x} = \{s_0, s_1, s_2, s_3, s_4\}$ and $S_{\text{VDIS}}^{d/y} = \{s_0, s_1, s_3, s_4\}$. The final estimated faulty final state set is $S_{\text{EndState}} = S_{\text{VDIS}}^{a/x} \cap S_{\text{VDIS}}^{b/y} \cap S_{\text{VDIS}}^{c/x} \cap S_{\text{VDIS}}^{d/y} = \{s_0, s_3\}$. Additional tests can now be added to verify the hypothesis.

In order to increase test confidence, two test sets $tv_{s_0} = \{(c/x) \cdot (dd/yx), (c/x) \cdot (daad/yyyx)\}$ and $tv_{s_3} = \{(c/x) \cdot (aaba/xxxy), (c/x) \cdot (dada/yyyy)\}$ are devised where s_0 and s_3 are tested, respectively. In both tests, two different UIOC sequences are used to verify the corresponding final state. The test results suggest that the erroneous final state of t_3 is s_0.

Since $t_3 \notin S_{\text{conflict}}$, a new transfer sequence needs to be constructed to isolate the fault that causes the failure observed in the original test. Still, the S_{conflict} is assumed to be $\{t_{16}, t_{17}, t_7, t_{10}, t_8\}$. The transfer sequence dc/yx is applied, moving M' from s_0 to s_3. In order to increase test confidence, two UIOC sequences, $aaba/xxxy$ and $dada/yyyy$, are used to verify s_3.

After $dcaaba$ and $dcdada$ are applied to M', $yxxxxy$ and $yxyyyy$ are received, respectively. This provides evidence that the current state is s_3. One continues applying those inputs in the original test sequence after the 17th input. By comparing the behaviour to the original test, it is found that the outputs remain unchanged. This increases confidence that the conflict set $S_{\text{conflict}} = \{t_{16}, t_{17}, t_7, t_{10}, t_8\}$ contains the faulty transition that caused the observed symptom. Additional tests are required to check each transition in S_{conflict}.

When constructing a test sequence, the traversing of t_3 needs to be avoided since it has been found to be faulty.

A set of tests, $VT = \{vt_{t_{16}}, vt_{t_{17}}, vt_{t_2}, vt_{t_{10}}, vt_{t_8}\}$, is devised where $vt_{t_{16}}$, $vt_{t_{17}}$, vt_{t_2}, $vt_{t_{10}}$ and vt_{t_8}, check t_{16}, t_{17}, t_2, t_{10} and t_8, respectively. In order to increase test confidence, each test in *VT* is comprised of two test sequences where two different UIOCs are used to verify the corresponding tail state. These tests are as given in Table IV.

When applying vt_{t_2} and $vt_{t_{10}}$ to M' no failure is observed, which suggests t_2 and t_{10} are correctly implemented. When applying $vt_{t_{17}}$ to M' both test sequences exhibit a failure which suggests t_{17} is faulty.

When applying $vt_{t_{16}}$ and vt_{t_8} to M', in both tests, one test sequence exhibits a failure while the other shows no error. The test results are $\{(ac/xx) \cdot (d/y) \cdot (bdab/xyyy), (ac/xx) \cdot (d/y) \cdot (aaab/yxxx)\}$ and $\{(a/x) \cdot (b/x) \cdot (bdab/xyyy), (a/x) \cdot (b/x) \cdot (aaab/yxxx)\}$. Through these observations, two hypotheses can be made: (1) t_8 and t_{16} are faulty, and $aaab/yxxx$ is a weak UIO sequence for s_4—a fault is exhibited by $bdab/xyyx$ but masked by $aaab/yxxx$; (2) t_8 and t_{16} are correctly implemented, but $bdab/xyyx$ traverses at least one faulty transition, leading to a failure being observed.

DOI: 10.1002/stvr

By examining the structure of $bdab/xyyx$, it is found that $bdab/xyyx$ traverses t_{17} which is found to be faulty. It is likely that the second hypothesis is true. To verify the hypothesis, $vt_{t_{16}}$ and vt_{t_8} are replaced with $\{(ac/xx) \cdot (d/y) \cdot (abdb/yyyy), (ac/xx) \cdot (d/y) \cdot (acab/yxyy)\}$ and $\{(a/x) \cdot (b/x) \cdot (abdb/yyyy), (a/x) \cdot (b/x) \cdot (acab/yxyy)\}$. In the tests, $(abdb/yyyy)$ and $(acab/yxyy)$ are two UIOC sequences for s_4 where, according to M, t_{17} is not traversed. After applying *acdabdb*, *acdacab*, *ababdb* and *abacab* to M', *xxyyyyy*, *xxyyxyy*, *xxyyyy* and *xxyxyy* are received, respectively. These results suggest that t_8 and t_{16} have been correctly implemented.

The faulty final state of t_{17} is then identified. After all elements in the input set are applied, a set of S_{VDIS} is obtained, this being $S^{a/y}_{\text{VDIS}} = \{s_1, s_2, s_4\}$, $S^{b/x}_{\text{VDIS}} = \{s_1, s_4\}$, $S^{c/x}_{\text{VDIS}} = \{s_0, s_1, s_2, s_3, s_4\}$ and $S^{d/y}_{\text{VDIS}} = \{s_0, s_1, s_3, s_4\}$. The final estimated faulty final state set is $S_{\text{EndState}} = S^0_{\text{VDIS}} \cap S^1_{\text{VDIS}} \cap S^2_{\text{VDIS}} \cap S^3_{\text{VDIS}} = \{s_1, s_4\}$. Additional tests are then devised to verify the hypothesis.

Two test sequences $ts_1 = (a/x) \cdot (baca/xyxy)$ and $ts_2 = (b/y) \cdot (aaab/yxxx)$ are devised where ts_1 tests s_1 while ts_2 checks s_4. It is concluded that the faulty final state of t_{17} is s_4.

7. COMPLEXITY

In this section, the complexity of the proposed approach is analysed. The analysis is comprised of two parts: the complexity of fault isolation and the complexity of fault identification. It is shown that the proposed approach can isolate and identify a single fault in low-order polynomial time.

7.1. Complexity of fault isolation

The complexity of fault isolation is determined by the strength of the UIOs used for the generation of test sequences. The strength of a UIO is its capability to resist fault maskings when required for state verification in the IUT [16]. If a symptom is exhibited by a strong test sequence, the conflict set S_{conflict} is of the maximum number $|S_{\text{conflict}}|_{\max} = L_{\text{UIO(max)}} + L_{\text{Link(max)}} + 1$; otherwise, if the test is a *k-degree* weak test, $|S_{\text{conflict}}|_{\max} = (k+1) \times (L_{\text{UIO(max)}} + L_{\text{Link(max)}} + 1), k \geq 0$[¶]. If there exists the problem of fault-masked UIO cycling, the test is treated as a $(k+c)$-*degree* weak test as discussed in the previous sections.

After the conflict set S_{conflict} is constructed, in order to isolate the faulty transitions, each transition $tr_i \in S_{\text{conflict}}$ needs to be tested. Let TrS be a set of transfer sequences where $trs_i \in TrS$ is used to move the IUT from s_0 to the initial state of $tr_i \in S_{\text{conflict}}$. Let $L_{TrS(\max)}$ be the maximum length of the transfer sequences in TrS. The maximum number of steps required for isolating a faulty transition is of $O(|S_{\text{conflict}}| \times (L_{TrS(\max)} + L_{\text{UIO(max)}} + 1))$.

The process of fault isolation from S_{conflict} considers the use of one UIO sequence for state verification when testing a transition $tr_i \in S_{\text{conflict}}$ and assumes this UIO sequence is a strong UIO. However, this might not be true. In order to increase test confidence, a set of test sequences TS_i might be used for the test of a transition tr_i in S_{conflict}, each of which uses a different UIO sequence to verify the final state of tr_i.

[¶] $k = 0$ is equivalent to the case where the test is a strong test.

DOI: 10.1002/stvr

Let $|TS_i|_{\max} = m$, $m \geq 1$. The maximum number of steps required to isolate a faulty transition is then of $O(|S_{\text{conflict}}| \times m \times (L_{TrS(\max)} + L_{\text{UIO}(\max)} + 1))$. Therefore, the maximum number of steps required to isolate a single fault is of $O((k + c + 1) \times (L_{\text{UIO}(\max)} + L_{\text{Link}(\max)} + 1) \times m \times (L_{TrS(\max)} + L_{\text{UIO}(\max)} + 1))$ where k is the number of faulty UIOs in the test sequence and c is the sum of times that the cycled faulty UIOs reoccur.

7.2. Complexity of fault identification

7.2.1. Construction of S_{EndState}

When identifying the faulty final state of an isolated faulty transition (if the transition holds a state transfer error), a set of estimated faulty final states S_{EndState} needs to be constructed by applying a set of faulty final state identification test sequences. Suppose, in the original test, the faulty transition and the sequence of transitions before this transition are executed by a test segment of length L_{sg}. The number of steps required to construct S_{EndState} is of $O((|I| - 1) \times (L_{sg} + 1))$, where I is the input set of the FSM. In order for the faulty final state to be identified, each state in S_{EndState} needs to be tested.

7.2.2. Determining the faulty final state

Let trs_{shortest} with length L_{trs} be the shortest transfer sequence that moves the IUT from s_0 to the initial state of tr_f. Let $|S_{\text{EndState}}| = q$, $1 \leq q \leq |S| - 1$. $s_i \in S_{\text{EndState}}$ is checked by applying trs_{shortest}, executing tr_f with the corresponding input and applying UIO_{s_i}. The length of UIO_{s_i} is less than or equal to $L_{\text{UIO}(\max)}$. The maximum number of steps required to test $s_i \in S_{\text{EndState}}$ is of $O(L_{trs} + L_{\text{UIO}(\max)} + 1)$. All states in S_{EndState} need to be tested. Therefore, the maximum number of steps required to identify the faulty final state in S_{EndState} is of $O(q \times (L_{trs} + L_{\text{UIO}(\max)} + 1))$.

Again, the process of faulty final state estimation considers the use of one UIO sequence to verify the corresponding state and assumes this UIO sequence is a strong UIO. In order to increase test confidence, a set of distinct UIOs, $MUIO_i$, may be used to verify state s_i in S_{EndState}. Let $|MUIO_i|_{\max} = p$, $p \geq 1$. The maximum number of steps required to identify the faulty final state in S_{EndState} is then of $O(q \times p \times (L_{trs} + L_{\text{UIO}(\max)} + 1))$.

By considering the process of the construction of S_{EndState} together, the maximum number of steps required to identify the faulty final state is of $O((|I| - 1) \times (L_{sg} + 1) + q \times p \times (L_{trs} + L_{\text{UIO}(\max)} + 1))$. In the worst case where $q = |S| - 1$, the maximum number of steps is of $O((|I| - 1) \times (L_{sg} + 1) + p \times (|S| - 1) \times (L_{trs} + L_{\text{UIO}(\max)} + 1))$, while in the best case where $q = 1$ the maximum number of steps is of $O((|I| - 1) \times (L_{sg} + 1) + p \times (L_{trs} + L_{\text{UIO}(\max)} + 1))$.

8. CONCLUSIONS AND FUTURE WORK

This paper has investigated fault diagnosis when testing from finite state machines and proposed heuristics to optimize the process of fault isolation and identification. In the proposed approach, a test sequence is first constructed for fault detection. Once a symptom is observed, additional tests are designed to identify the faults that are responsible for the occurrence of the observed symptom.

DOI: 10.1002/stvr

Based upon the original test, the proposed heuristics are applied in order to lead to the detected symptom being observed in some shorter test sequences. These shorter test sequences are then used for the construction of a set of diagnosing candidates that is of minimal size. The minimal set of candidates helps to reduce the cost of fault isolation and identification.

The complexity of the proposed approach has been described. A case study was used to demonstrate the application of the approach. In the case study, two state transfer faults were injected into the implementation. These faults were isolated and identified after applying the proposed heuristics.

The case study used in this paper considered the use of a comparatively simple example for fault isolation and identification. The paper shows how more complicated testing problems such as *k-degree* weak test and fault-masked UIO cycling can be catered for. However, more work is required to evaluate these approaches experimentally. This remains a topic for future work.

REFERENCES

1. Aho AV, Dahbura AT, Lee D, Uyar MU. An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours. *IEEE Transactions on Communications* 1991; **39**(11):1604–1615.
2. Chan WYL, Vuong ST, Ito MR. An improved protocol test generation procedure based on UIOs. *ACM SIGCOMM Computer Communication Review* 1989; **19**(4):283–294.
3. Hennie FC. Fault detecting experiments for sequential circuits. *Proceedings of the 5th Annual Switching Theory and Logical Design Symposium*, Princeton, NJ, 1964. IEEE Computer Society Press: Los Alamitos, CA, 1964; 95–110.
4. Shen YN, Lombardi F, Dahbura AT. Protocol conformance testing using multiple UIO sequences. *IEEE Transactions on Communications* 1992; **40**(8):1282–1287.
5. Yang B, Ural H. Protocol conformance test generation using multiple UIO sequences with overlapping. *Proceedings of the ACM Symposium on Communications, Architectures, and Protocols (ACM SIGCOMM 1990)*, Philadelphia, PA, September 1990. ACM Press: New York, 1990; 118–125.
6. Lee D, Yannakakis M. Testing finite state machines: State identification and verification. *IEEE Transactions on Computers* 1994; **43**(3):306–320.
7. Miller RE, Paul S. On the generation of minimal-length conformance tests for communication protocols. *IEEE/ACM Transactions on Networking* 1993; **1**(1):116–129.
8. Hierons RM. Extending test sequence overlap by invertibility. *The Computer Journal* 1996; **39**(4):325–330.
9. Hierons RM. Testing from a finite state machine: Extending invertibility to sequences. *The Computer Journal* 1997; **40**(4):220–230.
10. Ghedamsi A, von Bochmann G. Test result analysis and diagnosis for finite state machines. *Proceedings of the 12th International Conference on Distributed Computing Systems (ICDCS 1992)*, Yokohama, Japan, June 1992. IEEE Computer Society Press: Los Alamitos, CA, 1992; 244–251.
11. Ghedamsi A, von Bochmann G, Dssouli R. Multiple fault diagnosis for finite state machines. *Proceedings of the 12th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 1993)*, San Francisco, CA, March/April 1993. IEEE Computer Society Press: Los Alamitos, CA, 1993; 782–791.
12. Hierons RM. Minimizing the cost of fault location when testing from a finite state machine. *Computer Communications* 1998; **22**(2):120–127.
13. Chow TS. Testing software design modeled by finite state machines. *IEEE Transactions on Software Engineering* 1978; **4**(3):178–187.
14. Fujiwara S, von Bochmann G, Khendek F, Amalou M, Ghedamsi A. Test selection based on finite state models. *IEEE Transactions on Software Engineering* 1991; **17**(6):591–603.
15. Lee D, Yannakakis M. Principles and methods of testing finite state machines—a survey. *Proceedings of the IEEE* 1996; **84**(8):1090–1123.
16. Naik K. Fault-tolerant UIO sequences in finite state machines. *Proceedings of the IFIP WG6.1 TC6 8th International Workshop on Protocol Test Systems*, Evry, France, September 1995. North-Holland: Amsterdam, 1995; 201–214.
17. Guo Q, Hierons RM, Harman M, Derderian K. Improving test quality using robust unique input/output circuit sequences (UIOCs). *Information and Software Technology* 2006; **48**(8):696–707.

DOI: 10.1002/stvr