Reducing Concretization Effort in FSM-Based Testing of Software Product Lines

Vanderson Hafemann Fragal[1], Adenilso Simão[1], André Takeshi Endo[2], and Mohammad Reza Mousavi[3]

[1] Institute of Math. and Computer Sciences (ICMC), University of São Paulo, Brazil
[2] Federal Technological University of Paraná (UTFPR-CP), Brazil
[3] Centre for Research on Embedded Systems (CERES), Halmstad University, Sweden

1 Background

In this section, we present basic definitions regarding finite state machines, test properties, and software product line testing. Section 2 presents our detailed selection algorithm.

1.1 Finite State Machine

A finite-state machine (FSM) in our context, is a deterministic Mealy machine, which can be defined as follows.

Definition 1. *An FSM M is a 7-tuple $(S, s_0, I, O, D, \delta, \lambda)$, where S is a finite set of states with the initial state s_0, I is a finite set of inputs, O is a finite set of outputs, $D \subseteq S \times I$ is a specification domain, $\delta : D \to S$ is a transition function, and $\lambda : D \to O$ is an output function.*

If $D = S \times I$, then M is a `complete` FSM; otherwise, it is a `partial` FSM. As M is deterministic, a tuple $(s, x) \in D$ determines uniquely a `defined transition` of M. A transition from state s to s' with input x and output o is represented by quadruple $(s, x, o, s') \in D \to O \times S$, or alternatively by $s \xrightarrow[o]{x} s'$.

A sequence $\alpha = x_1, ..., x_k, \alpha \in I^*$ is a `defined input sequence` at state $s \in S$, if there exist states $s_1, ..., s_{k+1} \in S$, where $s = s_1$ such that $(s_i, x_i) \in D$ and $\delta(s_i, x_i) = s_{i+1}$, for all $1 \leq i \leq k$. Notation $\Omega(s)$ is used to denote all defined input sequences for state $s \in S$ and Ω_M denotes $\Omega(s_0)$. We extend the transition and output functions from input symbols to defined input sequences, including the `empty sequence` ε, as usual, assuming $\delta(s, \varepsilon) = s$ and $\lambda(s, \varepsilon) = \varepsilon$ for $s \in S$.

Given sequences $\alpha, \beta, \gamma \in I^*$, a sequence α is `prefix` of a sequence β, denoted by $\alpha \leq \beta$, if $\beta = \alpha\gamma$, for some sequence γ, and γ is a `suffix` of β. A sequence α is `proper prefix` of $\beta, \alpha < \beta$, if $\beta = \alpha\omega$ for some $\omega \neq \varepsilon$. We denote by $pref(\beta)$ the set of prefixes of β, i.e. $pref(\beta) = \{\alpha | \alpha \leq \beta\}$. For a set of sequences A, $pref(A)$ is the union of $pref(\beta)$ for all $\beta \in A$. If $A = pref(A)$, then we say that A is `prefix-closed`. Moreover, we say that a sequence $\alpha \in A$ is `maximal` in A if there is no sequence $\beta \in A$ such that α is a proper prefix of β. Given a sequence α and $k \geq 0$, we define α^k recursively as follows: $\alpha^0 = \varepsilon$; $\alpha^k = \alpha\alpha^{k-1}$,

if $k > 0$. The `common extensions` of two sequences are the sequences obtained by appending a common sequence to them.

An FSM M is said to be `initially connected`, if for each state $s \in S$, there exists an input sequence $\alpha \in \Omega_M$, such that $\delta(s_0, \alpha) = s$, called a `transfer sequence` for state s. Given a set $C \subseteq \Omega(s) \cap \Omega(s')$, states s and s' are `C-equivalent` if $\lambda(s, \gamma) = \lambda(s', \gamma)$ for all $\gamma \in C$. Otherwise, if there exists a $\gamma \in C$ such that $\lambda(s, \gamma) \neq \lambda(s', \gamma)$, then s and s' are `C-distinguishable`. An FSM M is `minimal` (or `reduced`), if every pair of states $s, s' \in S$ are C-distinguishable. In this paper, only minimal and initially connected machines are considered, since it is a pre-requisite for the P test case generation method[4] used to execute our strategy.

A set $C \subseteq \Omega_M$ is a `state cover` for an FSM M if, for each state $s \in S$, there exists $\alpha \in C$ such that $\delta(s_0, \alpha) = s$. The set $C \subseteq \Omega_M$ `covers` a transition (s, x) if there exists $\alpha \in C$ such that $\delta(s_0, \alpha) = s$ and $\alpha x \in C$. The set C is a `transition cover` (for M) if it covers every defined transition of M. A set of sequences is `initialized` if it contains the empty sequence.

1.2 Test Properties

In this paper, we use the full fault coverage criteria for FSMs from the P method[4]. We use the notion of test suite completeness with respect to a given fault domain and sufficiency conditions based on convergence and divergence properties introduced in [4].

Throughout this paper, we assume that $M = (S, s_0, I, O, D, \delta, \lambda)$ and $N = (Q, q_0, I, O', D', \Delta, \Lambda)$ are a specification FSM and an implementation FSM, respectively. Moreover, n is the number of states of M. We denote by $\Im$ the set of all deterministic FSMs with the same input alphabet as M for which all sequences in Ω_M are defined, i.e. for each $N \in \Im$ it holds that $\Omega_M \subseteq \Omega_N$. The set $\Im$ is called a `fault domain` for M and $\Im_n$ is the set of FSMs with n states. Faults can be detected by tests, which are input sequences defined in the specification FSM M.

Definition 2 ([4]). *A defined input sequence of FSM M is called a `test case` (or simply a `test`) of M. A `test suite` of M is a finite prefix-closed set of tests of M.*

The size of a test $\alpha \in I^*$ denoted by $|\alpha|$ is calculated by the number of inputs that it contains, i.e., $|\alpha| = k, \alpha = (x_1 ... x_k)$. Similarly, $|T|$ is the size of a test suite T calculated by the sum of all tests plus the reset operation for each maximal test, i.e., $|T| = \sum(|\alpha| + 1), \alpha \in T, \nexists_{\beta \in T} \bullet \beta = \alpha\gamma \wedge \gamma \neq \varepsilon$.

The distinguishability of FSMs is defined as the corresponding relation of their initial states, thus, tests are assumed to be applied in the initial state. Given a test suite T, FSMs are `T-equivalent` if their initial states are T-equivalent. Similarly, FSMs are `T-distinguishable` if their initial states are T-distinguishable.

Given two tests $\alpha, \beta \in \Omega_M$ they `converge` if when applied to the initial state they take the FSM into the same state, and they `diverge` if they take the FSM from the initial state to different states. Given a non-empty set of FSMs

$\Sigma \subseteq \Im$ and two tests $\alpha, \beta \in \Omega_M$, we say that α and β are Σ-`convergent` if they converge in each FSM of the set Σ. Similarly, we say that α and β are Σ-`divergent` if they diverge in each FSM of Σ.

Two tests α and β in a given test suite T are T-`separated` if there exist `common extensions` $\alpha\gamma, \beta\gamma \in T$, such that $\lambda(\delta(s_0, \alpha), \gamma) \neq \lambda(\delta(s_0, \beta), \gamma)$. T-separated tests are divergent in all FSMs that are T-equivalent to M. Given a test suite T, let $\Im(T)$ be the set of all $N \in \Im$, such that N and M are T-equivalent.

Lemma 1 ([4]). *Given a test suite T of an FSM M, T-separated tests are $\Im(T)$-divergent.*

We refer to [4] for detailed proofs of the lemmas and theorems presented in this section.

Divergence of two tests can be identified by different outputs produced by the tests, while the convergence of two tests cannot be directly ascertained. However, it can be shown that if a maximal number of states of FSMs in the fault domain is known, and the two tests are $\Im(T)$-divergent with tests reaching all but one state of the FSM M, these two tests must also converge in the same state in any FSM in the fault domain that is T-equivalent to M. Given a test suite T, let $\Im_n(T) = \Im_n \cap \Im(T)$, i.e. the set of FSMs in which are T-equivalent to M and have at most n states.

Lemma 2 ([4]). *Given a test suite T and $\alpha \in T$, let K be an $\Im_n(T)$-divergent set with n tests and $\beta \in K$ be a test M-convergent with α. If α is $\Im_n(T)$-divergent with each test in $K \backslash \{\beta\}$, then α and β are $\Im_n(T)$-convergent.*

The condition for n-completeness of a test suite T uses the notion of convergence-preserving set, for which the M-convergence implies the $\Im_n(T)$-convergence.

Definition 3 ([4]). *Given a test suite T of an FSM M, a set of tests is $\Im_n(T)$-`convergence-preserving` (or, simply, `convergence-preserving`) if all its M-convergent tests are $\Im_n(T)$-convergent.*

Any M-divergent set is by definition convergence-preserving, and $M \in \Im_n$, then $\Im_n(T)$ is by definition not empty. Next, we define the n-complete condition for the full fault coverage criteria.

Theorem 1 ([4]). *Let T be a test suite for an FSM M with n states. We have that T is an n-complete test suite for M if T contains an $\Im_n(T)$-convergence-preserving initialized transition cover set for M.*

When an n-complete test suite T has `guaranteed fault coverage` (or `full fault coverage`) for an FSM M, then, by executing T we are capable of detecting any fault in all FSM implementations $N \in \Im_n(T)$.

There exist several methods to generate n-complete test suites [1, 3, 4]. For example, the P method [4] uses two input parameters: a deterministic, initially connected, and minimal FSM M; and an initial test suite T. The initial set

T can be empty, and new tests are added/incremented (if necessary) until an n-complete test suite for M is produced. Therefore, the P method checks if all implementations $N \in \Im_n$ can be distinguished from M using T, and decides if more sequences need to be added to T. Experimental evaluation indicates that the P method often results in smaller n-complete test suites compared with other methods [2].

2 Selection Algorithm

In this section, we present an overview of the test sets used in the reuse strategy and our selection algorithm with extra details about its steps.

2.1 Overview

Given an FSM M, an n-complete test suite T and n states, the algorithm select tests that satisfies the condition of Definition 1 resulting an n-complete subset $S \subseteq T$. Tests of T are analyzed and only a specific subset of tests is selected to cover each transition for the convergence-preserving property, and thus removing redundancy. Figure 1 (b) shows the abstract selection algorithm steps, and more details are described in the next section.

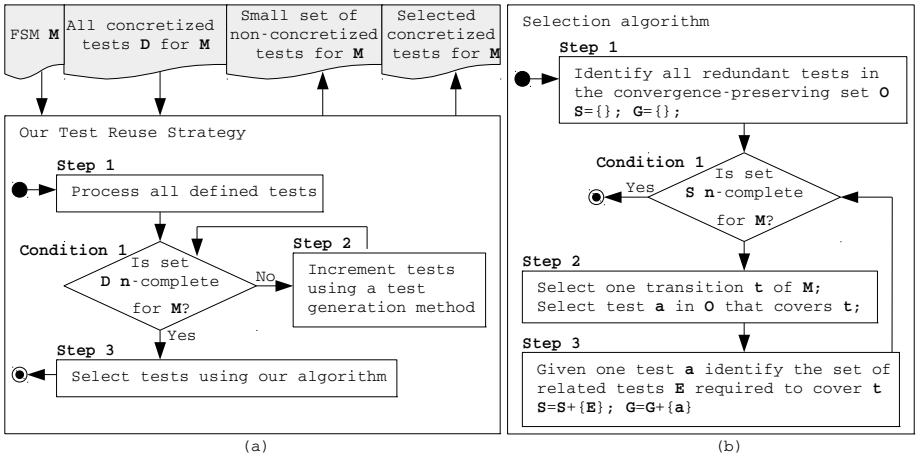

Fig. 1. (a) IRT-SPL test reuse strategy, and (b) selection algorithm.

The relations C and D represent subsets of $\Im_n(T)$-convergent and $\Im_n(T)$-divergent, respectively. The relation D is initially the set of all pairs of T-separated tests according to Definition 1. Next, a M-divergent state cover set K with n tests is identified. The relation C is the set of all pairs of $\Im_n(T)$-convergent tests according to Definition 2 (including the identity set where $(\alpha, \alpha) \in C$). We used

10 rules that were introduced in [4] to identify extra pairs of convergent and divergent test pairs in C and D, respectively. The ten rules are [4]:

1. If (α, β) is added to C, for each $(\alpha, \chi) \in C$, add (β, χ) to C.
2. If (α, β) is added to C, then, for all their common extensions $\alpha\varphi, \beta\varphi \in T$, add $(\alpha\varphi, \beta\varphi)$ to C.
3. If (α, β) is added to D, and they are common extensions of tests α' and β', then add (α', β') to D.
4. If (α, β) is added to C, then, for each $\chi \in T$ if $(\alpha, \chi) \in D$, add (β, χ) to D; if $(\beta, \chi) \in D$, add (α, χ) to D.
5. If (α, β) is added to D, then, for each $\chi \in T$ if $(\alpha, \chi) \in C$, add (β, χ) to D; if $(\beta, \chi) \in C$, add (α, χ) to D.
6. If (α, β), with $\alpha \leq \beta$, is added to D and there exists sequence φ and $k > 1$, such that $\beta = \alpha\varphi^k$, then add $(\alpha, \alpha\varphi)$ to D.
7. If $(\alpha, \alpha\beta\gamma)$ is added to C, and $(\alpha, \alpha\gamma) \in D$, then add $(\alpha, \alpha\beta)$ to D.
8. If $(\alpha, \alpha\gamma)$ is added to D, then, for each sequence β such that $(\alpha, \alpha\beta\gamma) \in C$, add $(\alpha, \alpha\beta)$ to D.
9. If $(\alpha, \alpha\gamma)$ is added to C, then, for each sequence β such that $(\beta, \beta\gamma) \in D$, add (α, β) to D.
10. If $(\beta, \beta\gamma)$ is added to D, then, for each sequence α such that $(\alpha, \alpha\gamma) \in C$, add (α, β) to D.

The relation $C_{\cup}(K) = \{\beta | (\alpha, \beta) \in C, \alpha \in K\}$ is an $\Im_n(T)$-convergence-preserving set for M according to Definition 3. Moreover, we define sets $V \subseteq C \cup D$ for verified pairs of tests of C and D, $G \subseteq C_{\cup}(K)$ for goal coverage, and $S \subseteq T$ to store the selected tests of T. To identify convergent and divergent pairs that were added to C and D by rules 1-10, we use designed inverse rules to trace tests back to T-separated pairs where they originally came from.

11. If $(\beta, \chi) \in C$ was added by rule 1, then check $(\alpha, \beta), (\alpha, \chi) \in C$
12. If $(\alpha\varphi, \beta\varphi) \in C$ was added by rule 2, then check $(\alpha, \beta) \in C$
13. If $(\alpha, \beta) \in D$ was added by rule 3, then check $(\alpha\gamma, \beta\gamma) \in D$.
14. If $(\alpha, \chi) \in D$ was added by rule 4, then check $(\beta, \chi) \in D, (\alpha, \beta) \in C$.
15. If $(\alpha, \chi) \in D$ was added by rule 5, then check $(\beta, \chi) \in C, (\alpha, \beta) \in D$.
16. If $(\alpha, \alpha\varphi) \in D$ was added by rule 6, then check $(\alpha, \beta) \in D$, with $\alpha \leq \beta, \beta = \alpha\varphi^k, k > 1$.
17. If $(\alpha, \alpha\beta) \in D$ was added by rule 7, then check $(\alpha, \alpha\gamma) \in D, (\alpha, \alpha\beta\gamma) \in C$.
18. If $(\alpha, \alpha\beta) \in D$ was added by rule 8, then check $(\alpha, \alpha\gamma) \in D, (\alpha, \alpha\beta\gamma) \in C$.
19. If $(\alpha, \beta) \in D$ was added by rule 9, then check $(\beta, \beta\gamma) \in D, (\alpha, \alpha\gamma) \in C$.
20. If $(\alpha, \beta) \in D$ was added by rule 10, then check $(\beta, \beta\gamma) \in D, (\alpha, \alpha\gamma) \in C$.

2.2 Detailed Selection Algorithm

The main detailed steps of the selection algorithm are presented in Figure 2.

Given an FSM M and n-complete test suite T for M, initially the algorithm processes T and identify sets D, K, C, and $C_{\cup}(K)$. Some tests may be added in those sets (except K) by rules 1-10. The verification set V start empty, coverage set G initialized, and the resulting selected set S empty.

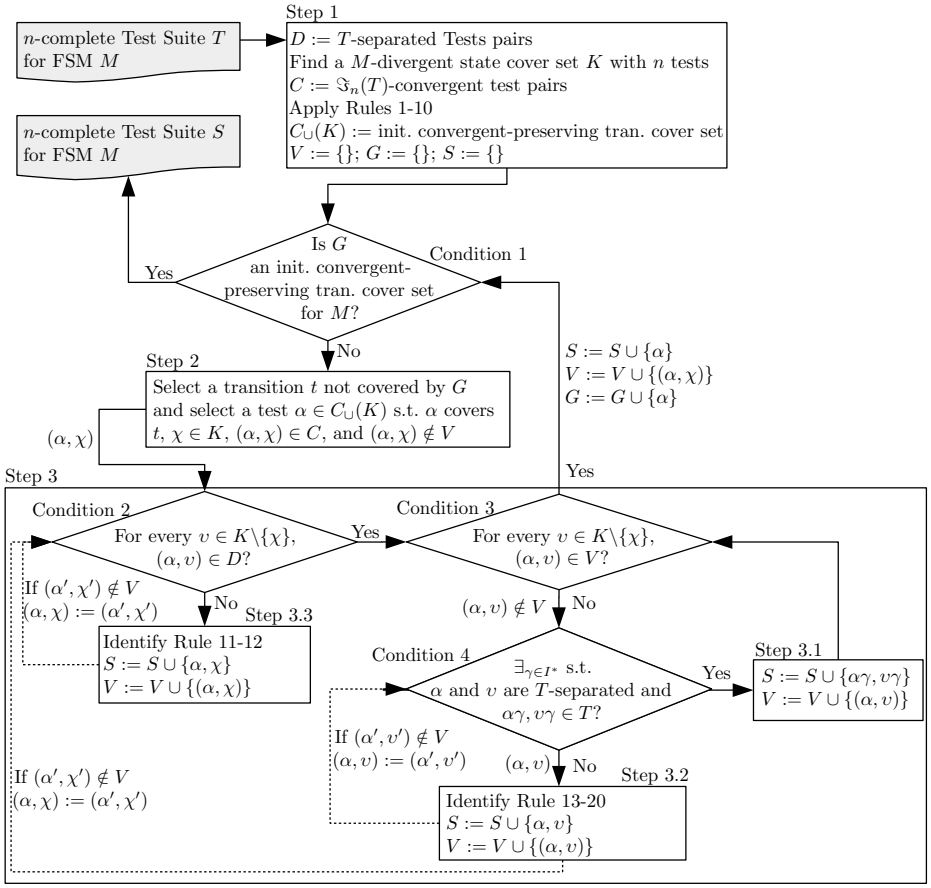

Fig. 2. Algorithm for reducing an n-complete test suite.

Condition 1 checks whether G meets the condition of a n-complete test suite, and since it is initially empty Step 2 is executed. To populate G first select a transition t not covered by G and select a test $\alpha \in C_{\cup}(K)$ such that α covers t, $\chi \in K$, $(\alpha, \chi) \in C$, and $(\alpha, \chi) \notin V$. Then, we check whether the pair (α, χ) was added to C by some rule or by Lemma 2. Let Condition 2 be true for (α, χ), then on Condition 3 no pair (α, υ) is true as V is empty at this point. Moving to Condition 4, let it be true, then tests $\alpha\gamma, \upsilon\gamma$ are added to S and (α, υ) added to V marking this pair as visited and checked. Back to Condition 3, assuming that every other pair (α, υ) make Condition 4 true, then after verifying them Condition 3 turns true, (α, χ) is added to V, α is added to S and G finishing the basic cycle for t.

Let Step 2 select a test pair not verified $(\beta, \chi) \in C$ that was added to C by rule 1. If Condition 2 turn false, then Step 5 identify pairs e.g., $(\alpha, \beta), (\alpha, \chi) \in C$ by rule 11 that were used to add (β, χ) in C, then, add β and χ to S and put (β, χ) in S to mark as verified. First, if $(\alpha, \beta) \notin V$ then the execution continues on Condition 2 for (α, β) instead of (β, χ) and (α, χ) enters in a waiting state. If $(\alpha, \beta) \in V$, then (α, β) is ignored, and if $(\alpha, \chi) \notin V$ then the execution continues on Condition 2 for (α, χ) instead of (β, χ). However, if $(\alpha, \beta), (\alpha, \chi) \in V$ then the execution returns to the last waiting state. If there is no waiting point to return, then the execution stops with a failure, and T is not n-complete.

On Condition 3 for every pair $(\alpha, \upsilon) \notin V$ select one and use as input for Condition 4 and put the rest in a waiting state. If Condition 4 is false, then identify which pairs of tests were responsible on the addition of (α, υ) to D, then add α and υ to S, and (α, υ) to V to mark as verified. For example, if (α, υ) was triggered by Rule 14, then $(\beta, \upsilon) \in D, (\alpha, \beta) \in C$. If $(\beta, \upsilon) \notin V$ then the execution continues on Condition 4 for (β, υ) and (α, β) enters in a waiting state. If $(\beta, \upsilon) \in V$, then (β, υ) is ignored, and if $(\alpha, \beta) \notin V$ then the execution continues on Condition 2 for (α, β). However, if $(\beta, \upsilon), (\alpha, \beta) \in V$ then the execution returns to the last waiting state. If there is no waiting point to return, then the execution stops with a failure, and T is not n-complete.

2.3 Analysis

In the remainder of this section, we prove that the algorithm terminates and the obtained test suite S is indeed n-complete.

Theorem 2. *Given an n-complete test suite T for an FSM M, the algorithm terminates with an n-complete test suite $S \subseteq T$.*

Proof. The algorithm contains four cycles. We show that each cycle can be executed a finite number of times and, thus, the algorithm indeed terminates. Then, we prove that the resulting test suite is n-complete.

Cycle 1 corresponds to the executions where Condition 1 does not hold, but Conditions 2-3 do, i.e. select tests to populate G. As the selected pair (α, χ) cannot be in the set of verified pairs V and the number of pairs in C is finite, at the end of the cycle (α, χ) is put in V and cannot be selected again, thus, the number of executions of Cycle 1 is bounded.

Cycle 2 corresponds to the executions where Condition 3 does not hold, but Condition 4 does, i.e. the pair (α, υ) is T-separated. On Step 3 (α, υ) is put in V to mark as verified. As the number of pairs in D is finite, the Cycle is bounded to D size.

Cycle 3 corresponds to the executions where Condition 2 does not hold, i.e. the pair (α, χ) was added to C by rules. On Step 5 (α, χ) is put in V to mark as verified. The cycle continues only if one of the responsible pairs for (α, χ) is not in V, otherwise, the execution returns to the last waiting state. As the number of pairs in C is finite, the Cycle is bounded to C size.

Cycle 4 corresponds to the executions where Condition 4 does not hold, i.e. the pair (α, υ) was added to D by rules. On Step 4 (α, υ) is put in V to mark as verified. The cycle continues only if one of the responsible pairs for (α, υ) is not in V, otherwise, the execution returns to the last waiting state. As the number of pairs in D is finite, the Cycle is bounded to D size.

Given an FSM M, we now show that the obtained test suite S is n-complete. When the algorithm terminates, Condition 1 holds and G is an initialized $\Im_n(S)$-convergent-preserving transition cover set for M, by Definition 1. For every test α added to G, a subset of tests that made $\alpha \in C_{\cup}(K)$ were added to S resulting in a n-complete test suite for M. □

The time required to execute the algorithm is similar to the P method, since several steps have the same complexity executed in inverse order. However there are two exceptions: the state cover set K that was identified by simple breadth-search instead of a clique; and we only apply the first 10 rules on the first step instead of each step of P method. We refer to [4] for more details about the complexity on each step.

References

1. Chow, T.S.: Testing Software Design Modeled by Finite-State Machines. IEEE Transactions on Software Engineering SE-4(3), 178–187 (1978)
2. Endo, A.T., Simao, A.: Evaluating test suite characteristics, cost, and effectiveness of FSM-based testing methods. Information and Software Technology 55(6), 1045–1062 (2013)
3. Luo, G., Petrenko, A., Petrenko, R., Bochmann, G.V.: Selecting Test Sequences For Partially-Specified Nondeterministic Finite State Machines. In: Proc. of IFIP 1994. pp. 91–106 (1994)
4. Simao, A., Petrenko, A.: Fault Coverage-Driven Incremental Test Generation. The Computer Journal 53(9), 1508–1522 (2010)