Using a Minimal Number of Resets when Testing from a Finite State Machine

R. M. Hierons [a]

[a] *Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, United Kingdom*

Key words: finite state machine, test generation algorithm, reset, minimization [1].

1 Introduction

Finite State Machines (FSMs) are used to model a number of classes of systems, including communications protocols and control systems. There has thus been much interest in automating the generation of tests from FSMs [1,5,7,8]. A reset is an operation that takes the system from each state to the initial state. The use of a reset may increase the cost of testing and reduce its effectiveness [2,3,9]. Thus, it is often desirable to minimize the number of resets used in testing. This paper investigates problems of the form: produce a test sequence p, that contains each element of some non-empty set T of sequences, such that there does not exist a test sequence p' that contains each element of T and has fewer resets than p. The proposed (polynomial time) algorithm is *guaranteed* to produce a test sequence that has the minimum number of resets when considering test sequences that connect the sequences from T but do not utilize overlap.

2 Preliminaries

2.1 Finite State Machines

A (completely specified and deterministic) FSM M is defined by a tuple $(S, s_1, \delta, \lambda, X, Y)$ in which $S = \{s_1, \ldots, s_n\}$ is a finite set of states, $s_1 \in S$ is the

[1] This is a preliminary version of the following paper: R.M. Hierons, 2004, Using a minimal number of resets when testing from a finite state machine, *Information Processing Letters*, **90** 6, pp. 287-292.

initial state, X and Y are the finite input and output alphabets, δ is the next state function, and λ is the output function. If M receives input x when in state s, a transition t is executed, producing output $y = \lambda(s, x)$ and moving M to $s' = \delta(s, x)$. t is defined by the tuple $(s, s', x/y)$. The functions δ and λ can be extended to be applied to sequences of inputs, giving δ^* and λ^* respectively.

Given FSM M, $(s^1, s^2, x_1/y_1), \ldots, (s^{m-1}, s^m, x_{m-1}/y_{m-1})$ is a *sequence of transitions* if each $(s^i, s^{i+1}, x_i/y_i)$ is a transition of M. A *test sequence* for M is a sequence of transitions [2] that starts at s_1. M is *strongly connected* if for every $(s, s') \in S \times S$ there is some $\bar{x} \in X^*$ that moves M from s to s'. M is *initially connected* if each state can be reached from s_1. An FSM may be converted into an initially connected FSM by removing unreachable states. States s and s' are *equivalent* if for every $\bar{x} \in X^*$, $\lambda^*(s, \bar{x}) = \lambda^*(s', \bar{x})$. Two FSMs are *equivalent* if their initial states are equivalent. FSM M is *minimal* if no equivalent FSM has fewer states. An FSM may be converted into an equivalent minimal FSM [6]. Only minimal initially connected FSMs are considered.

When testing from an FSM it is normal to insist that, for each transition t, the test sequence contains a subsequence that tests t [1,8]. The notion of what it means to tests a transition varies but often either involves just executing t or following t by some preset sequence, that checks its final state, such as a *unique input/output sequence (UIO)* or a *distinguishing sequence (DS)* (see, for example, [8]). Test generation then involves connecting the elements of some set T of sequences of transitions to form one test sequence, although in some cases overlap between elements of T is utilized (here, we assume it is not). Each transition t starts some element of T which is usually either t or t followed by a UIO or DS. For an overview of testing from an FSM see [5,8].

We assume that the IUT has a (reliable) *reset*: some input r that leads to null output (denoted $-$) and moves the IUT to its initial state irrespective of its previous state. In order to simplify the exposition it will be assumed that the reset is not mentioned in the specification. When M is not strongly connected, it may be necessary to use the reset in testing. An FSM, that will be called M_0, is shown in Figure 1. Each reset is in the form of $(s_i, s_1, r/-)$ for a state s_i. Any test sequence that contains each transition of M_0 must use the reset.

For some systems it is difficult to realize a reset [3,9]. Each instance of the reset in the test sequence may involve human intervention. Thus, the inclusion of resets may *increase the cost* of executing a test sequence. It may be necessary to use long test sequences to detect faults that involve extra states in the implementation [2,3]. The inclusion of a reset splits a test sequence into a set of separate (shorter) sequences and so may reduce the chance of finding such faults: it may *reduce the effectiveness* of a test sequence.

[2] The term test sequence can also be used to denote an input sequence.

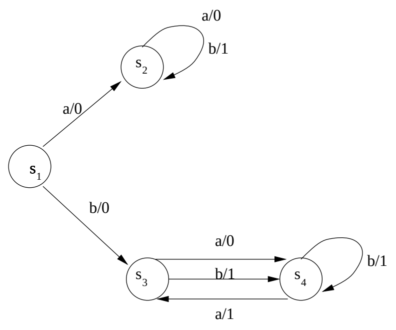

Fig. 1. The Finite State Machine M_0

2.2 Directed Graphs

A *directed graph (digraph)* G is defined by a pair (V, E) where V is a set of vertices and E is a set of edges. An edge e is defined by its initial vertex v, its final vertex v', and possibly a label l. e is represented by (v, v', l). FSM M can be represented by a digraph $G = (V, E)$ in which a state s is represented by a vertex v_s and a transition $t = (s, s', x/y)$ is represented by edge $(v_s, v_{s'}, x/y)$.

A sequence $\sigma = (v^1, v^2, l^1), (v^2, v^3, l^2), \ldots, (v^{j-1}, v^j, l^{j-1})$ of edges from digraph G forms a *path* from v^1 to v^j. σ represents a *tour* if $v^1 = v^j$. Note that for all $1 \leq k < j$, $(v^k, v^{k+1}, l^k), \ldots, (v^{j-1}, v^j, l^{j-1}), (v^1, v^2, l^1), \ldots, (v^{k-1}, v^k, l^{k-1})$ represents the same tour as σ and *starts* at v^k. σ represents a *circuit* if it represents a tour and $v^1, \ldots, v^{j-1}$ are distinct vertices. $R = \{(v, v_1, r/-) | v \in V\}$ will denote a set of edges that represents the reset. Digraph $G = (V, E)$ is *strongly connected* if for all $(v, v') \in V \times V$ there is a path from v to v' in G. G is *weakly connected* if the underlying undirected graph is connected: for all $(v, v') \in V \times V$ there is a sequence $\sigma = (v^1, v^2, l^1), (v^2, v^3, l^2), \ldots, (v^{j-1}, v^j, l^{j-1})$ such that for all $1 \leq k < j$, at least one of (v^{k-1}, v^k, l^{k-1}) and (v^k, v^{k-1}, l^{k-1}) is an edge of G. Since the digraph G representing FSM M does not contain reset edges, G is weakly connected but need not be strongly connected.

For vertex v of $G = (V, E)$, $indegree_E(v)$ is the number of edges from E entering v and $outdegree_E(v)$ is the number of edges from E leaving v. G is *symmetric* if $\forall v \in V. indegree_E(v) = outdegree_E(v)$. An *Euler Tour* is a tour that passes through each edge exactly once. G has an Euler Tour if and only if it is symmetric and strongly connected (see, for example, [4]). Given $C \subseteq E$ there is a corresponding sub-digraph $G[C] = (V^C, C)$; $V^C = \{v \in V | indegree_C(v) \neq 0 \vee outdegree_C(v) \neq 0\}$. G_1 is a *component* of G if it is a maximal strongly connected sub-digraph of G. If G is a weakly connected symmetric digraph then G is strongly connected (see, for example, [4]).

Given a set E_T of edges from G, a tour of G is a *rural postman tour* if it contains every edge in E_T. The *Rural Chinese Postman Problem (RCPP)* is: find a minimum length rural postman tour. While the RCPP is NP-complete, heuristics have been developed for this problem. One such polynomial time heuristic [1] will now be given. In the first phase a network algorithm produces a minimal symmetric augmentation of E_T: a minimum cost multi-set E' of edges from $E \cup E_T \cup R$ such that $E_T \subseteq E'$ and $G' = (V, E')$ is symmetric. If G' is connected an Euler Tour $\mathcal{T}$ is produced. A test sequence is found by starting $\mathcal{T}$ at v_1. If G' is not connected, edges are added to form some G'' that is connected and symmetric and an Euler Tour of G'' is produced.

3 Test generation

We will assume that a set T of sequences of transitions has been given and each transition starts some element of T. The problem is to produce a test sequence that connects the elements of T while introducing as few resets as possible. This section will adapt the heuristic, described in Section 2, to produce a test generation algorithm.

Suppose FSM M is represented by $G = (V, E)$ and the set T is represented by set E_T of edges; there is a one-to-one correspondence between the sequences in T and the edges in E_T. The problem is to produce a path p that connects the element of E_T while introducing as few resets as possible. Let Υ_T denote the set of tours of the *augmented digraph* $G_T = (V, E \cup E_T \cup R)$ that include each edge of E_T. In Section 5, Theorem 2 shows that the problem can be represented in terms of producing a tour $\mathcal{T}$ from Υ_T that, amongst the tours of Υ_T, has fewest edges from R. Where $\mathcal{T}$ contains no reset, p is produced by starting $\mathcal{T}$ with the initial vertex. Where $\mathcal{T}$ contains one or more resets, p is produced by starting $\mathcal{T}$ after some reset, having removed this reset.

Each edge in E_T has a cost: the number of transitions in the corresponding sequence. An edge in E has cost 1. Suppose U is an upper bound on the cost of the edges from E_T. There are $|E_T|$ sequences to be connected and each of these has length at most U. Further, between two edges e_1 and e_2 from E_T there is a connecting path from the final vertex of e_1 to the initial vertex of e_2. There are $|E_T|$ such paths and, assuming minimal length paths are chosen, each has length at most $n - 1$ where n denotes the number of states of M. An upper bound on the test length is provided by $c_M^T = |E_T|(U + n - 1) + 1$. Each (reset) edge in R will be given cost c_M^T.

An approach similar to that used by [1] is applied. In the first phase a network algorithm produces a minimal symmetric augmentation of E_T: a minimum cost multi-set E' of edges from $E \cup E_T \cup R$ with the property that $E_T \subseteq E'$ and $G' = (V, E')$ is symmetric. If G' is strongly connected an Euler Tour $\mathcal{T}$ is produced. The test sequence is produced from $\mathcal{T}$ as explained above.

Suppose G' is not strongly connected. It is sufficient to add edges from E to G' in order to form G'' that is symmetric and strongly connected. In Section 5, Theorem 4 shows that for each component G_i of G', there is a circuit of G that contains v_1 and a vertex of G_i. G'' is formed by, for each G_i, adding such a circuit. By the definition of G, this circuit contains no reset edges. A test sequence is found by producing an Euler Tour $\mathcal{T}$ of G''. A path may be produced from $\mathcal{T}$ as explained above. The algorithm is summarized below.

(1) Produce G_T and determine the cost of each edge.
(2) Using the (polynomial time) algorithm described in [1], produce a minimum symmetric augmentation G' of E_T in G_T.
(3) If G' is not strongly connected, form G'' by adding to each component G_i in G' (that does not contain v_1) a circuit that passes through v_1 and some vertex of G_i and that contains no resets. Otherwise $G'' = G'$.
(4) Find an Euler Tour $\mathcal{T}$ of G''.
(5) If $\mathcal{T}$ contains resets, form a test sequence p by starting $\mathcal{T}$ immediately after some instance of the reset and delete the final reset. Otherwise form a test sequence p by starting $\mathcal{T}$ at v_1.

Proposition 1 *The time complexity of the test generation algorithm is polynomial in n and $|X|$.*

4 Example

In this section we will apply the test generation algorithm to the example. Table 1 gives a set of sequences, with names, that test the transitions of M_0 (using UIOs). These form the set T. Each edge is given a cost. For example, the edge with label t_{1a} has cost 3 while the edge with label t_{3a} has cost 2. Each edge in R is given cost $c^T_{M_0} = 49$ since each edge from E_T has length at most $U = 3$, M_0 has 4 states, and $|E_T| = 8$.

The digraph (V, E_T) is not symmetric; a minimal symmetric augmentation of E_T may be produced by adding a reset from s_2, a reset from s_3, and two copies of the edge $(s_3, s_4, a/0)$. The corresponding digraph G' is shown in Figure 2. Naturally, this choice need not be unique. If the reset from s_3 is removed the test sequence $t_{1a}, t_{2a}, t_{2b}, r/-, t_{1b}, t_{3a}, t_{3b}, a/0, t_{4a}, a/0, t_{4b}$ is produced. This test sequence contains one reset and thus clearly minimizes the number of resets.

Table 1
A set of transition tests for M_0

Names	Initial State	Sequence	Final State
t_{1a}	s_1	a/0,b/1,a/0	s_2
t_{1b}	s_1	b/0,a/0,a/1	s_3
t_{2a}	s_2	a/0,b/1,a/0	s_2
t_{2b}	s_2	b/1,b/1,a/0	s_2
t_{3a}	s_3	a/0,a/1	s_3
t_{3b}	s_3	b/1,a/1	s_3
t_{4a}	s_4	a/1,a/0,a/1	s_3
t_{4b}	s_4	b/1,a/1	s_3

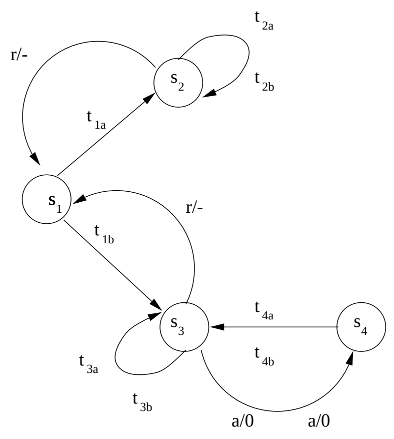

Fig. 2. The digraph G'

5 Proof of correctness

First, it will be proved that test generation may be represented in terms of generating a tour with a minimum number of resets.

Theorem 2 *Suppose a tour $\mathcal{T}$ includes each element of T while introducing as few resets as possible. Let p denote a path produced in the following way: If $\mathcal{T}$ contains no reset then form p by starting $\mathcal{T}$ at the initial vertex; otherwise choose some edge e that represents a reset in $\mathcal{T}$, start $\mathcal{T}$ immediately after e and delete e. Then p is a path that contains each element of T while minimizing the number of resets added.*

Proof

Where $\mathcal{T}$ contains no resets, clearly p minimizes the number of resets. Suppose $\mathcal{T}$ contains one or more resets. Proof by contradiction: suppose there is some path p' that contains every element of T and has fewer resets than p. Form a tour $\mathcal{T}'$ from p' by ending p' with a reset. Then $\mathcal{T}'$ is a tour that contains every element from T and has fewer resets than $\mathcal{T}$. This contradicts $\mathcal{T}$ being a tour that minimizes the number of resets. $\square$

Theorem 3 *If G' is not strongly connected then it may be partitioned into a set of components.*

Proof

G' may be partitioned into a set $G_1, \ldots, G_k$ of maximal weakly connected subdigraphs. Since G' is symmetric, each G_i is symmetric and weakly connected and so is strongly connected. $\square$

Definition 1 *Suppose $C \subseteq E \cup E_T \cup R$ and $G[C]$ is strongly connected. The* closure*, $\overline{C}$, of C in G is the largest subset of $E \cup E_T \cup C$ such that $C \subseteq \overline{C}$ and $G[\overline{C}]$ is strongly connected.*

The following shows that if C_i is the edge set of component G_i of G' then $\overline{C_i}$ contains an edge connected to v_1. Thus for each component G_i that does not contain v_1 (and thus contains no reset), since $G[\overline{C_i}]$ is strongly connected there is some circuit of G that passes through a vertex of G_i and v_1 and contains no reset [3]. By adding such circuits we get a strongly connected digraph G'' that contains G'. G'' has no more resets than G'. Since G' is symmetric and G'' is formed by adding circuits to G', G'' is symmetric and so has an Euler Tour.

Theorem 4 *Suppose that the algorithm leads to digraph G' with components represented by edge sets $C_1, \ldots, C_k$. Then for all $1 \leq i \leq k$, $\overline{C_i}$ contains an edge connected to the initial vertex v_1.*

Proof

Proof by contradiction: suppose $\overline{C_i}$ does not have an edge connected to v_1.

Case 1: No edge of $E \setminus \overline{C_i}$ leaves a vertex of $G[\overline{C_i}]$. Since $\overline{C_i}$ does not contain an edge connected to v_1, some $e \in E \setminus \overline{C_i}$ ends in a vertex of $G[\overline{C_i}]$. Consider an edge $e' \in E_T$ that represents an element of T that starts with the transition corresponding to e. Observe that, since no edge of $E \setminus \overline{C_i}$ leaves a vertex of $G[\overline{C_i}]$, e' ends at a vertex from $G[\overline{C_i}]$. If e' starts at a vertex of $G[\overline{C_i}]$, the edges in e' can be added to $\overline{C_i}$ while preserving strong connectivity. So, by the maximality of $\overline{C_i}$, e' must start with a vertex not in $G[\overline{C_i}]$ and so $e' \in \overline{C_j}$ for some C_j with $\overline{C_j} \neq \overline{C_i}$. Thus, $\overline{C_i}$ and $\overline{C_j}$ have edges connected to the vertex that ends e' and so $\overline{C_i} \cup \overline{C_j}$ is strongly connected. Thus, by the maximality of $\overline{C_i}$ and $\overline{C_j}$, $\overline{C_i} = \overline{C_j}$, providing a contradiction as required.

[3] Since the reset is not in M none of the edges in $E \cup E_T \cup C_i$ contains a reset.

Case 2: There is an edge $e \in E \backslash \overline{C_i}$ that leaves a vertex of $G[\overline{C_i}]$. Consider some edge $e' \in E_T$ that represents an element of T that starts with the transition corresponding to e. If e' ends at a vertex of $G[\overline{C_i}]$, the edges in e' can be added to $\overline{C_i}$ while preserving connectivity. Thus, e' must end in a vertex not in $G[\overline{C_i}]$ and so $e' \in \overline{C_j}$ for some C_j with $\overline{C_j} \neq \overline{C_i}$. Since $\overline{C_i}$ and $\overline{C_j}$ both have edges connected to the vertex starting e', $\overline{C_i} = \overline{C_j}$, providing a contradiction as required. □

Lemma 5 *G' is a symmetric augmentation of E_T in the augmented digraph G_T with the minimal number of resets.*

Proof

Given a symmetric augmentation H of E_T in G_T let $r(H)$ denote the number of reset edges in H and $nr(H)$ denote the total cost of the other edges in H. The cost of H is $c(H) = nr(H) + r(H)c_M^T$.

Let H_1 denote a symmetric augmentation of E_T in G_T with a minimal number of reset edges. Further, let H_1 be a minimum cost symmetric augmentation of E_T in G_T that has this number of resets. Then $nr(H_1) < c_M^T$ and thus $c(H_1) < (r(H_1)+1)c_M^T$. Observe that $c(G') \geq r(G')c_M^T$. Since G' is a minimum cost symmetric augmentation of E_T in G_T, $c(G') \leq c(H_1)$. Thus $r(G')c_M^T \leq c(G') \leq c(H_1) < (r(H_1)+1)c_M^T$ and so $r(G')c_M^T < (r(H_1)+1)c_M^T$. Thus $r(G') \leq r(H_1)$ and the result follows. □

Theorem 6 *The test sequence produced by the test generation algorithm minimizes the number of resets used.*

Proof

Suppose test sequence τ_1 is produced from G'' and τ_2 is another test sequence that (separately) contains every element of T. Let τ_2' denote an extension, to τ_2, that ends at s_1 and adds a minimal number of resets to τ_2. τ_2' corresponds to a rural postman tour of E_T in G_T which, in turn, corresponds to a symmetric augmentation H of E_T. Given a symmetric augmentation H_1 of E_T, let $r(H_1)$ denote the number of reset edges in H_1. By Theorem 4, $r(G'') = r(G')$ and thus, by Lemma 5, $r(H) \geq r(G'')$. If G' has no reset edges then τ_1 minimizes the number of resets. Otherwise τ_1 contains $r(G'') - 1$ reset edges. Further, τ_2 contains at least $r(H) - 1$ reset edges and so τ_1 contains at most the same number of reset edges as τ_2. □

6 Conclusions

Many approaches, to generating a test from an FSM M, are based around producing a test sequence that contains some set T of predefined sequences that, between them, test the transitions of M. In some cases, in order to include each element of T it is necessary to use resets. The use of resets may *increase the cost* of testing and *reduce the effectiveness* of testing. This paper has considered the problem of producing a test sequence that contains the elements of T while using as few resets as possible. The paper has introduced an algorithm that represents the optimization problem in terms of the Rural Chinese Postman Problem (RCPP). Since the RCPP is NP-hard, a heuristic is adapted. The resultant polynomial time algorithm is *guaranteed* to minimize the number of resets used, when overlap between the elements of T is not utilized. Sometimes, overlap can be used to further reduce the number of resets. Future work will consider how overlap may be incorporated.

References

[1] A. V. Aho, A. T. Dahbura, D. Lee, and M. U. Uyar. An optimization technique for protocol conformance test generation based on UIO sequences and Rural Chinese Postman Tours. In *Protocol Specification, Testing, and Verification VIII*, pages 75–86, Atlantic City, 1988. Elsevier (North-Holland).

[2] B. Broekman and E. Notenboom. *Testing Embedded Software*. Addison-Wesley, London, 2003.

[3] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite state models. *IEEE Transactions on Software Engineering*, 17(6):591–603, 1991.

[4] A. Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[5] D. Lee and M. Yannakakis. Principles and methods of testing finite-state machines. *Proceedings of the IEEE*, 84(8):1089–1123, 1996.

[6] E. P. Moore. Gedanken-Experiments. In C. Shannon and J. McCarthy, editors, *Automata Studies*. Princeton University Press, 1956.

[7] A. Petrenko, N. Yevtushenko, and G. v. Bochmann. Testing deterministic implementations from nondeterministic FSM specifications. In *Testing of Communicating Systems*, pages 125–141, Darmstadt, Germany, 9-11 September 1996. Chapman and Hall.

[8] D. P. Sidhu and T.-K. Leung. Formal methods for protocol testing: A detailed study. *IEEE Transactions on Software Engineering*, 15(4):413–426, 1989.

[9] M. Yao, A. Petrenko, and G. v. Bochmann. Conformance testing of protocol machines without reset. In *Protocol Specification, Testing and Verification, XIII (C-16)*, pages 241–256. Elsevier (North-Holland), 1993.