Separating Sequence Overlap for Automated Test Sequence Generation

R. M. Hierons, Brunel University, UK

Abstract

Finite state machines have been used to model a number of classes of system and there has thus been much interest in the automatic generation of test sequences from finite state machines. Many finite state machine based test techniques utilize sequences that check the final states of transitions, the most general such sequence being a separating sequence: an input sequence that distinguishes between two states of an FSM. When using such techniques the test sequence length can be reduced by utilizing overlap. This paper investigates overlap for separating sequences and shows how this can be incorporated into test sequence generation.

Keywords: Test sequence generation, finite state machine, separating sequence, characterizing set, overlap.

1 Introduction

The existence of a formal model or specification of the required behaviour of an implementation under test (IUT) introduces the possibility of automating or semi-automating parts of the testing process. This can lead to more effective and efficient testing and thus, potentially, to the development of cheaper and more reliable software.

A finite state machine (FSM) can be used to model or specify the required behaviour of a state-based system (see, for example, [Broekman and Notenboom, 2003, Pomeranz and Reddy, 1997]). If an implementation I has been produced in the presence of an FSM M that describes the required behaviour of I, it is important to verify I against M. Testing usually forms part of this verification. Several specification languages extend the FSM formalism, by adding data to states and actions, to form extended finite state machines (EFSMs). Examples include SDL, ESTELLE, stream X-machines and Statecharts. FSM based test techniques can often be applied where there is a specification in such a language [Duale and Uyar, 2004, Petrenko et al., 2004]. A number of authors have also noted that FSM based techniques can assist when testing against a Z specification [Derrick and Boiten, 1999, Dick and Faivre, 1993, Hierons, 1992] and that such techniques can be used in model-based testing [Farchi et al., 2002, Grieskamp et al., 2002].

When testing against an FSM M, it is common to utilize sequences that distinguish between the states of M and there are three main approaches to this: using a distinguishing sequence (see, for example, [Hennie, 1964, Hierons and Ural, 2002, Ural et al., 1997]); unique input/output sequences (see, for example [Aho et al., 1988, Shen et al., 1990]); or a characterizing set (see, for example, [Chow, 1978, Sidhu and Leung, 1989]). An input sequence is a distinguishing sequence for M if it leads to a different output sequence from each state of M. An input/output sequence $\bar{x}/\bar{y}$ is a unique input/output sequence (UIO) for state s of M if M produces $\bar{y}$ in response to $\bar{x}$ from state s but from no other state of M. Thus, a distinguishing sequence is capable of verifying any state of M and a UIO for state s is capable of verifying state s. Naturally, if a distinguishing sequence exists for an FSM M then it defines a UIO for every state of M. There are FSMs that have states that do not have UIOs. For such an FSM, in order to verify a state s it may be necessary to use several sequences that, between them, distinguish s from every other state of M. These sequences are called *separating sequences*. A set of separating sequences that distinguishes all of the states of M is called a characterizing set. These notions are defined formally in Section 2.

Many FSM based test criteria are expressed in terms of developing a set T of individual test subsequences for the transitions of FSM M and then generating a transition sequence that contains these test subsequences. The test sequence is then formed by extracting the input sequence from this transition sequence. Thus, test sequence generation can be seen as a process in which we connect test subsequences in an optimal manner. The overall test sequence length can be reduced by allowing the test subsequences to overlap. A number of authors have studied the problem of utilizing overlap when UIOs are used for state checking [Hierons, 1996, Hierons, 1997, Miller and Paul, 1993, Yang and Ural, 1990, Zhang and Probert, 2005] and when a distinguishing sequence is used [Ural and Zhu, 1993]. This paper generalizes overlap to the case where a characterizing set is used for state checking and thus when separating sequences are used. This is important since an FSM might not have either a distinguishing sequence or a UIO for every state but every (minimal) FSM has a characterizing set.

This paper makes the following contributions. The primary contribution is that it formalizes the notion of overlap between separating sequences. It then shows how this overlap can be introduced into test sequence generation algorithms. Finally, we prove that these algorithms produce shorter test sequences that test every transition than the previous corresponding algorithms: the test sequence produced by the proposed algorithm cannot be longer than the test sequence produced by the previous algorithms and can be shorter.

This paper is structured as follows. Section 2 provides an introduction to FSMs and FSM based testing. Section 3 explores overlap between separating sequences. Sections 4 and 5 give test sequence generation algorithms and these are evaluated in Section 6. Finally, conclusions are drawn in Section 7.

2 Finite State Machines and Directed Graphs

A (deterministic and completely specified) *finite state machine* M is defined by a tuple $(S, s_0, \delta, \lambda, X, Y)$ in which S is a finite set of *states*, s_0 is the *initial state*, δ is the *state transfer function*, λ is the *output function*, X is the finite *input alphabet*, and Y is the finite *output alphabet*. If M receives input x when in state s it produces

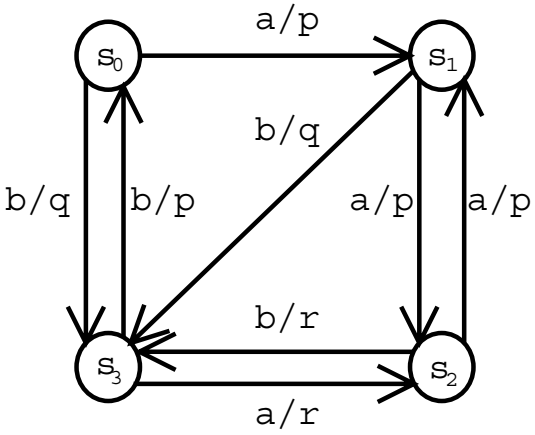

Figure 1. The FSM M_0

output $y = \lambda(s, x)$ and moves to state $s' = \delta(s, x)$. This defines a *transition* $t = (s, s', x/y)$ with *starting state* s and *ending state* s'. The functions δ and λ can be extended to take input sequences giving functions of types $S \times X^* \rightarrow S$ and $S \times X^* \rightarrow Y^*$ respectively. Let $t = (s, s', x/y)$ be a transition of M and let $\bar{x}$ and $\bar{x}'$ be input sequences. Then in $\bar{x}'$ we *follow* the transition t by the input sequence $\bar{x}$ if there exists input sequences $\bar{x}_1$ and $\bar{x}_2$ with $\delta(s_0, \bar{x}_1) = s$ and $\bar{x}' = \bar{x}_1 x \bar{x} \bar{x}_2$. Only deterministic and completely specified FSMs are considered in this paper. For more on FSMs see, for example, [Gill, 1962, Kohavi, 1978]. An FSM, called M_0 throughout the paper, is shown in Figure 1. Here, for example, $\delta(s_1, a) = s_2$ and $\lambda(s_1, a) = p$ and thus M_0 contains the transition $(s_1, s_2, a/p)$. Further, $\delta(s_1, aa) = s_1$ and $\lambda(s_1, aa) = pp$.

A sequence $\bar{\tau} = t_1, \ldots, t_k$, $t_i = (s_i, s_{i+1}, x_i/y_i)$ for $1 \leq i \leq k$, of consecutive transitions of M is called a *transition sequence*. The state $s_1 = head(\bar{\tau})$ is the starting state of $\bar{\tau}$ while $s_{k+1} = tail(\bar{\tau})$ is the ending state of $\bar{\tau}$. The input/output sequence $x_1/y_1 \ldots x_k/y_k$ of transition sequence $\bar{\tau}$ is called the *label* of $\bar{\tau}$ and is denoted $label(\bar{\tau})$. The transition sequence $\bar{\tau}$ can also be represented by the tuple $(s_1, s_{k+1}, x_1/y_1, \ldots, x_k/y_k)$. Given input/output sequence $\bar{z} = x_1/y_1, \ldots, x_k/y_k$, $\bar{x} = In(\bar{z}) = x_1, \ldots, x_k$ is the *input portion* of $\bar{z}$ and $\bar{y} = Out(\bar{z}) = y_1, \ldots, y_k$ is the *output portion* of $\bar{z}$. In a slight abuse of notation, $\bar{z}$ can be represented as $\bar{x}/\bar{y}$.

When testing from an FSM we apply an input sequence, called a test sequence, to the IUT and compare the resultant output sequence with that expected. If the observed input/output sequence is not the label of a transition sequence from M that starts at s_0 then a failure has been observed.

Two states s and s' of M are *equivalent* if for every input sequence $\bar{x}$, $\lambda(s, \bar{x}) = \lambda(s', \bar{x})$. If $\lambda(s, \bar{x}) \neq \lambda(s', \bar{x})$ then $\bar{x}$ is said to be a *separating sequence*. Two FSMs are *equivalent* if their initial states are equivalent. An FSM M is *minimal* if no equivalent FSM has fewer states. Given an FSM M it is possible to generate an equivalent minimal FSM M' [Moore, 1956] and there are algorithms for transforming an FSM with n states and p inputs into such a minimal FSM in $O(pn \log n)$[Hopcroft, 1971]. Thus only minimal FSMs are considered in this paper.

A *reset operation* is an input $reset \in X$ with the property that for every state s of M we have that $\delta(s, reset) =$

s_0. Some test sequence generation methods assume that there is a *reliable reset operation*: a reset operation that is known to be correctly implemented. While many systems have a reset operation, the tester cannot always rely upon the correctness of such an operation. The test sequence generation algorithms introduced in Sections 4 and 5 do not rely on the presence of a reset operation.

A *directed graph (digraph)* G is defined by a set V of vertices and a set E of directed edges between the vertices. Each edge can be given a label and an edge e from vertex v to vertex v' with label l is represented by (v, v', l). A *walk* from digraph $G = (V, E)$ is a sequence of consecutive edges from E. A walk $e_1, \ldots, e_k$, $e_i = (v_i, v_{i+1}, l_i)$ $(1 \leq i < k)$ is a *path* if the vertices $v_1, \ldots, v_k$ are distinct. Walk $e_1, \ldots, e_k$, $e_i = (v_i, v_{i+1}, l_i)$ $(1 \leq i < k)$ is a *tour* if $v_1 = v_{k+1}$ and then $e_j, \ldots, e_k, e_1, \ldots, e_{j-1}$ represents starting $e_1, \ldots, e_k$ at vertex v_j.

An FSM can be represented by a digraph in which the vertices represent states and the edges, labelled with the input/output pairs, represent transitions. A digraph is *strongly connected* if for each ordered pair of vertices (v, v') with $v \neq v'$ there is a path from v to v'.

An FSM is *strongly connected* if the digraph that represents it is strongly connected. An FSM M is *initially connected* if every state s of M can be reached from the initial state of M: for all $s \in S$ there exists some input sequence $\bar{x}$ such that $\delta(s_0, \bar{x}) = s$. Since M is minimal it must be initially connected. If M is initially connected and has a reset operation then M is strongly connected. It is worth noting that an FSM can be strongly connected without having a reset operation. In this paper we assume that M is strongly connected.

Many test sequence generation techniques use sequences that distinguish the states of the specification M in order to verify the states of the IUT. Normally one of the following is used to verify a state of M: a distinguishing sequence (DS); a unique input/output sequence (UIO); or a characterizing set (see, for example, [Sidhu and Leung, 1989]). An input sequence $\bar{x}$ is a distinguishing sequence if each state of M leads to a different output sequence in response to $\bar{x}$: for all s, s' in S with $s \neq s'$ we have that $\lambda(s, \bar{x}) \neq \lambda(s', \bar{x})$. Thus, in order to check the final state of a transition t of M using distinguishing sequence $\bar{x}$ it is sufficient to follow t by $\bar{x}$.

A UIO for a state s is an input/output sequence $\bar{x}/\bar{y}$ with the property that it distinguishes s from every other state of M. Thus, $\bar{x}/\bar{y}$ is a UIO for state s of M if and only if $\bar{y} = \lambda(s, \bar{x})$ and for all $s' \in S \setminus \{s\}$ we have that $\lambda(s, \bar{x}) \neq \lambda(s', \bar{x})$. A UIO for a state s of M can verify that the state of M is s, but need not be able to verify any other state of M. In order to check the final state s' of a transition t of M it is sufficient to follow t by the input portion of the UIO for s'.

Some FSMs do not have either a DS or a UIO for every state. Further, there is no polynomial upper bound on the length of the shortest DS or UIO and the problems of deciding whether an FSM has a DS or a UIO for state s are PSPACE complete [Lee and Yannakakis, 1994]. Where the FSM does not have a known DS or a complete set of UIOs it is normal to use a characterizing set $W = \{\bar{w}_1, \ldots, \bar{w}_m\}$: a set of input sequences with the property that for every pair of states s, s' of M, if $s \neq s'$ then there is some $\bar{w}_i \in W$ such that $\lambda(s, \bar{w}_i) \neq \lambda(s', \bar{w}_i)$. Thus, the set of output sequences produced by executing each $\bar{w}_i \in W$ from state s of M verifies s. Every minimal FSM with n states has a characterizing set that contains at most $n - 1$ sequences each of length at most $n - 1$ and this set can be produced in $O(n^2)$ time (see, for example, [Kohavi, 1978]).

Input	State s_0	State s_1	State s_2	State s_3
b	q	q	r	p
ab	pq	pr	pq	rr

Table 1. The output from the $\bar{w}_i$

Now consider the FSM M_0. It is possible to observe that s_0 does not have a UIO, since any input sequence beginning with a cannot distinguish between s_0 and s_2 and any input sequence beginning with b cannot distinguish between s_0 and s_1. Thus, M_0 does not have a distinguishing sequence. However, M_0 has characterizing set $W = \{\bar{w}_1, \bar{w}_2\}$ for $\bar{w}_1 = b$ and $\bar{w}_2 = ab$.

In order to verify a state s_i of M it is possible to use a set W_i of prefixes of sequences from W with the property that for every state $s_j \neq s_i$ of M, some element of W_i distinguishes s_i from s_j [Fujiwara et al., 1991]. By considering Table 1 we see that the following sets can be used for M_0: $W_0 = \{b, ab\}$, $W_1 = \{ab\}$, $W_2 = \{b\}$, and $W_3 = \{b\}$.

2.1 Test sequence generation

Given FSM M, let Φ_M^k denote the set of FSMs with the same input and output alphabets as M and no more than k states. When testing from an FSM M it is normal to assume that the IUT behaves like an unknown FSM $M_I \in \Phi_M^k$ for some predetermined k. Most techniques for generating test sequences from an FSM M fall into one of two classes.

1. They produce a *checking experiment*: a set of test sequences that, between them, are guaranteed to determine correctness as long as the IUT behaves like some element of Φ_M^k.

2. They produce a set T of transition sequences, called test subsequences, that check the transitions of M and then form a walk $\bar{\rho}$ of M that contains each element of T. The test sequence is produced by taking the input portion of the label of $\bar{\rho}$.

Where feasible it is desirable to produce a checking experiment. Approaches that do not rely on the use of a reliable reset and generate a checking sequence based on a characterizing set produce test sequence lengths that are at least exponential in the number of states of M [Rezaki and Ural, 1995]. Where $k = n$ and there is a reliable reset operation a checking experiment with polynomial length can be produced using the W-method [Chow, 1978] or the Wp-method [Fujiwara et al., 1991]. However, the test sequence length is still greater than that produced by algorithms that devise a single test sequence that checks the transitions of M. Further, the use of resets can increase the cost of testing and reduce its effectiveness [Hierons, 2004, Yao et al., 1993]. Thus, there are a range of problems for which it is not feasible to produce a checking experiment. Interestingly, the experience of [Motteler et al., 1994] and [Sidhu and Leung, 1989] suggests that sequences that include a test subsequence for every transition but are not checking sequences are usually effective at finding faults.

The second class of test sequence generation algorithm identifies a set T of test subsequences and produces a walk $\bar{\rho}$ of M, that starts at s_0 such that each element of T is a subsequence of $\bar{\rho}$. Once such a walk $\bar{\rho}$ has been found, we test with the input portion of the label of $\bar{\rho}$. This is captured by the following notion of a walk covering a test subsequence from T: we insist that each element of T is covered by the final walk.

Definition 1 *A walk* $\bar{\rho}$ covers *a sequence* $\bar{\tau}$ *of transitions from* M *if and only if* $\bar{\tau}$ *is a subsequence of* $\bar{\rho}$.

Suppose that we are using a characterizing set $W = \{\bar{w}_1, \ldots, \bar{w}_m\}$ in test sequence generation. Given a transition $t = (s, s', x/y)$, the input/output sequences $x\bar{w}_1/y\lambda(s', \bar{w}_1), \ldots, x\bar{w}_m/y\lambda(s', \bar{w}_m)$ could form the labels of the corresponding test subsequences from s. In this case $T = \{(s, \delta(s, x\bar{w}_j), x\bar{w}_j/\lambda(s, x\bar{w}_j))|s \in S, x \in X, \delta(s, x) = s_j\}$. Where each state has a UIO and the UIO for state s_i is denoted $\bar{x}_i/\bar{y}_i$, if $s' = s_j$ then usually only $x\bar{x}_j/y\bar{y}_j$ is used for t.

Given a set T of test subsequences for M, if M is strongly connected then it is possible to produce one walk that covers each element of T and extract the corresponding test sequence. The problem of finding a short walk that contains each element of T can be described by representing M by a digraph $G = (V, E)$ and then adding a set E_t of edges that represent the elements of T. Suppose that for each state s_i of M, s_i is represented by the vertex $v_i \in V$. Given a transition subsequence $\bar{\tau}$ from T with starting state s_i and ending state s_j, in E_t we represent $\bar{\tau}$ by an edge with starting vertex v_i and ending vertex v_j. The optimisation problem can then be seen as that of finding the shortest tour of $G_t = (V, E \cup E_t)$ that contains every edge from E_t. Given the tour produced by this, the test sequence is the corresponding input sequence. This is an instance of the *Rural Chinese Postman Problem (RCPP)* [Aho et al., 1988]. While the RCPP is NP-complete Aho et al. [Aho et al., 1988] use a polynomial time algorithm that is optimal under certain well defined conditions. This algorithm has time complexity of $O(qn \log n)$ [Aho et al., 1988] for an FSM with n states and q transitions. Where it is suboptimal, a set of tours is produced and these can be connected.

2.2 Previous work on test sequence generation and overlap

It has been noted that the test subsequences in T need not be disjoint: they may *overlap* [Aho et al., 1988]. Yang and Ural [Yang and Ural, 1990] introduced a heuristic to utilize this overlap where UIOs are used. They note that where test subsequences overlap several of them can be combined to form a single sequence, called a *fully overlapped transition sequence (FOTS)*, that utilizes overlap. A set of FOTS is generated and then combined optimally. However, there may be a number of alternative ways of devising the FOTS and the choice made in test sequence generation might be suboptimal. A similar approach has been applied using distinguishing sequences [Ural and Zhu, 1993].

Miller and Paul [Miller and Paul, 1993] represent overlap, when using UIOs, in terms of *non-convergent transitions* and Hierons [Hierons, 1992, Hierons, 1996] represents overlap by using *invertible transitions*. A transition $(s, s', x/y)$ is *invertible* if it is the only transition with input x and output y that enters state s'. Invertible transitions preserve state information: if the final state of an invertible transition has been checked then the initial state

has also been checked. Thus if $t = (s, s', x/y)$ is an invertible transition and $\bar{x}/\bar{y}$ is a UIO for s', then $x\bar{x}/y\bar{y}$ is a UIO for s. Based on this, it is possible to reduce the number of UIOs required in testing and thus the total test sequence length. This approach effectively includes the process of choosing the FOTS within test optimization. Invertibility can be extended to *invertible sequences* [Hierons, 1997].

Where there is more than one known UIO for each state it is possible to produce a shorter test sequence even if overlap is not used [Shen et al., 1990]. The optimisation algorithm chooses an appropriate UIO for each transition and may use different UIOs for two transitions that end at the same state. The approaches of Miller and Paul [Miller and Paul, 1993] and Hierons [Hierons, 1992, Hierons, 1996] take advantage of the fact that there may be many alternative UIOs for a state: in effect they generate additional UIOs.

Previous approaches to utilizing overlap assume that UIOs or a distinguishing sequence are being used for state checking. However, an FSM need not have a distinguishing sequence or a UIO for each state. Since every minimal FSM has a characterizing set, this paper generalizes the approaches of Miller and Paul [Miller and Paul, 1993] and Hierons [Hierons, 1992, Hierons, 1996] to the case where a characterizing set is being used.

3 Separating Sequences and Overlap

In this section we formalize overlap when using separating sequences. While every minimal FSM has a characterizing set, this set is not unique. Throughout this section we assume that a characterizing set $W = \{\bar{w}_1, \ldots, \bar{w}_m\}$ of M has been chosen. We now define terminology that underlies the analysis of overlap.

Definition 2 *A transition sequence $\bar{\tau}$, with starting state $s = head(\bar{\tau})$ and a label whose input portion is $\bar{x}$, is a* $\bar{w}_j$ sequence *for state s if and only if for all $s' \in S \setminus \{s\}$ we have that if $\lambda(s', \bar{w}_j) \neq \lambda(s, \bar{w}_j)$ then $\lambda(s', \bar{x}) \neq \lambda(s, \bar{x})$.*

Thus, in terms of distinguishing state s from other states of M, the input portion of the label of any $\bar{w}_j$ sequence is *at least* as powerful as $\bar{w}_j$. Where $\bar{\tau}$ is a $\bar{w}_j$ sequence for s, the input portion $\bar{x}$ of the label of $\bar{\tau}$ can be used, when checking state s, in place of $\bar{w}_j$. It transpires that this fact can be used to allow overlap to be utilized. It is worth noting that this says nothing about the ability of $\bar{x}$ to distinguish states other than s.

Definition 3 *A transition sequence $\bar{\tau}$ whose label has input portion $\bar{x}$ is said to be* $(\bar{w}_i, \bar{w}_j)$ converting *if $\bar{x}\bar{w}_i$ is a $\bar{w}_j$ sequence for state $head(\bar{\tau})$.*

We now define the following variants on Definition 3.

Definition 4 *Suppose $\bar{\tau}$ has length 1, containing transition t. If $\bar{\tau}$ is $(\bar{w}_i, \bar{w}_j)$ converting then t is said to be a* $(\bar{w}_i,\bar{w}_j)$ converting transition.

Definition 5 *If there is some k such that $\bar{\tau}$ is $(\bar{w}_k, \bar{w}_i)$ converting then $\bar{\tau}$ is* $(\_, \bar{w}_i)$ converting. *Similarly, if there is some k such that $\bar{\tau}$ is $(\bar{w}_i, \bar{w}_k)$ converting then $\bar{\tau}$ is* $(\bar{w}_i, \_)$ converting.

Input	State s_0	State s_1	State s_2	State s_3
$\bar{w}_1$ (b)	q	q	r	p
$\bar{w}_2$ (ab)	pq	pr	pq	rr
$b\bar{w}_1$ (bb)	qp	qp	rp	pq
$a\bar{w}_2$ (aab)	ppr	ppq	ppr	rpq
$b\bar{w}_2$ (bab)	qrr	qrr	rrr	ppq

Table 2. The output from the $\bar{w}_j$ and $x\bar{w}_i$

Definition 6 *If all the transitions for some input value $x \in X$ are $(\bar{w}_i, \bar{w}_j)$ converting then x is said to be* strongly $(\bar{w}_i, \bar{w}_j)$ converting.

The key point is that if $\bar{\tau}$ is $(\bar{w}_i, \bar{w}_j)$ converting and has a label whose input portion is $\bar{x}$ then $\bar{x}\bar{w}_i$ can be used in place of $\bar{w}_j$. This allows a test subsequence for the final transition of $\bar{\tau}$ to be used as part of a test subsequence for a transition preceding $\bar{\tau}$ and thus for there to be overlap between these two test subsequences. To see this let us suppose that the last transition of $\bar{\tau}$ is t' and t is a transition with input x whose ending state is $head(\bar{\tau})$. As $\bar{x}\bar{w}_i$ (from state $head(t)$) applies $\bar{w}_i$ after t' and $\bar{\tau}$ is $(\bar{w}_i, \bar{w}_j)$ converting, in testing $x\bar{x}\bar{w}_i$ can replace both the transition t followed by $\bar{w}_j$ and t' followed by $\bar{w}_i$ when applied in state $head(t)$. This can be used to reduce the test effort by introducing overlap.

It is straightforward to decide which transitions involving input $x \in X$ are $(\bar{w}_i, \bar{w}_j)$ converting, for each i, j. This can be achieved by finding the output, from each state, produced by $\bar{w}_i$ and $x\bar{w}_i$. The values for M_0 are shown in Table 2. As $\bar{w}_2 = a\bar{w}_1$, a is strongly $(\bar{w}_1, \bar{w}_2)$ converting. Input a also produces $(\bar{w}_2, \bar{w}_1)$ converting transitions when executed from states s_1 and s_3, b produces $(\bar{w}_1, \bar{w}_2)$ converting transitions when executed from s_2 or s_3, and b is strongly $(\bar{w}_2, \bar{w}_1)$ converting. Input b is also strongly $(\bar{w}_1, \bar{w}_1)$ converting, a is strongly $(\bar{w}_2, \bar{w}_2)$ converting, b produces a $(\bar{w}_2, \bar{w}_2)$ converting transition for states s_2 and s_3, and a produces a $(\bar{w}_1, \bar{w}_1)$ converting transition for states s_1 and s_3.

Suppose, for example, that we execute $ba\bar{w}_1$ from the initial state of M_0. Then we achieve two things.

1. We follow the transition $(s_3, s_2, a/r)$ by $\bar{w}_1$.

2. We follow the transition $(s_0, s_3, b/q)$ by a $\bar{w}_2$ sequence since a is strongly $(\bar{w}_1, \bar{w}_2)$ converting.

Thus, the execution of $ba\bar{w}_1$ from the initial state of M_0 includes two test subsequences that overlap.

Not all $(\bar{w}_i, \bar{w}_j)$ converting transitions are invertible, an example being the transition $(s_0, s_1, a/p)$ in M_0, which is $(\bar{w}_2, \bar{w}_1)$ converting but is not invertible since there is another transition (from state s_2) with label a/p and ending state s_1. The following result thus demonstrates that this notion, of $(\bar{w}_i, \bar{w}_j)$ converting transitions, is a generalization of invertibility.

Proposition 1 *If $t = (s, s', x/y)$ is invertible, $\bar{w}_i$ is the input portion of a UIO for s', and $\bar{w}_j$ is the input from a UIO for s then t is $(\bar{w}_i, \bar{w}_j)$ converting.*

Proof

It is sufficient to demonstrate that given any $s_k \neq s$, $x\bar{w}_i$ distinguishes between s and s_k. If $\lambda(s, x) \neq \lambda(s_k, x)$ then $x\bar{w}_i$ distinguishes between s and s_k, as required. If $\lambda(s, x) = \lambda(s_k, x)$ then, as t is invertible, $\delta(s, x) \neq \delta(s_k, x)$. Let $s'_k = \delta(s_k, x)$. Then, as $s'_k \neq s'$ and $\bar{w}_i$ is the input portion of a UIO for s', $\lambda(s', \bar{w}_i) \neq \lambda(s'_k, \bar{w}_i)$ and thus $\lambda(s, x\bar{w}_i) \neq \lambda(s_k, x\bar{w}_i)$, as required. □

It is possible to efficiently determine whether a sequence of transitions is $(\bar{w}_i, \bar{w}_j)$ converting.

Proposition 2 *Given an FSM M with n states and transition sequence $\bar{\tau}$ it is possible to determine whether $\bar{\tau}$ is $(\bar{w}_i, \bar{w}_j)$ converting in time of $O(n(|\bar{\tau}| + |\bar{w}_i| + |\bar{w}_j|))$.*

Proof

Let $\bar{x}$ denote the input portion of the label of $\bar{\tau}$. It is sufficient to observe that we need to determine the response of M to both $\bar{x}\bar{w}_i$ and $\bar{w}_j$ from the n states of M and determine which states are distinguished from $head(\bar{\tau})$ by these input sequences. □

The following result demonstrates that it is possible to combine $(\bar{w}_i, \bar{w}_j)$ converting sequences in the natural way.

Proposition 3 *If $\bar{\tau}$ is $(\bar{w}_i, \bar{w}_j)$ converting, $\bar{\tau}'$ is $(\bar{w}_j, \bar{w}_k)$ converting, and the ending state of $\bar{\tau}'$ is the starting state of $\bar{\tau}$ then $\bar{\tau}'\bar{\tau}$ is $(\bar{w}_i, \bar{w}_k)$ converting.*

Proof

Let s denote the starting state of $\bar{\tau}$ and s' denote the starting state of $\bar{\tau}'$. Further, let $\bar{x}$ denote the input portion of the label of $\bar{\tau}$ and $\bar{x}'$ denote the input portion of the label of $\bar{\tau}'$. As $\bar{\tau}$ is $(\bar{w}_i, \bar{w}_j)$ converting, the input sequence $\bar{x}\bar{w}_i$ distinguishes between s and s_p whenever $\bar{w}_j$ distinguishes between s and s_p. Similarly, whenever $\bar{w}_k$ distinguishes between s' and s_q, $\bar{x}'\bar{w}_j$ distinguishes between s' and s_q.

Suppose $\bar{w}_k$ distinguishes between s' and s_q. It is sufficient to prove that $\bar{x}'\bar{x}\bar{w}_i$ also distinguishes between s' and s_q. Let $s_p = \delta(s_q, \bar{x}')$. As $\bar{\tau}'$ is $(\bar{w}_j, \bar{w}_k)$ converting and $\bar{w}_k$ distinguishes between s' and s_q, $\bar{x}'\bar{w}_j$ distinguishes between s' and s_q. Thus $\lambda(s', \bar{x}'\bar{w}_j) \neq \lambda(s_q, \bar{x}'\bar{w}_j)$. Clearly if $\lambda(s', \bar{x}') \neq \lambda(s_q, \bar{x}')$ then $\bar{x}'\bar{x}\bar{w}_i$ distinguishes between s' and s_q as required. If $\lambda(s', \bar{x}') = \lambda(s_q, \bar{x}')$ then $\lambda(s, \bar{w}_j) \neq \lambda(s_p, \bar{w}_j)$. Then, as $\bar{\tau}$ is $(\bar{w}_i, \bar{w}_j)$ converting and $\bar{w}_j$ distinguishes between s and s_p, $\lambda(s, \bar{x}\bar{w}_i) \neq \lambda(s_p, \bar{x}\bar{w}_i)$. Thus $\lambda(s', \bar{x}'\bar{x}\bar{w}_i) \neq \lambda(s_q, \bar{x}'\bar{x}\bar{w}_i)$ as required. □

Note that a $(\bar{w}_i, \bar{w}_j)$ converting sequence can have length in excess of 1. However, for purpose of test generation we focus on the use of $(\bar{w}_i, \bar{w}_j)$ converting transitions.

4 Test Sequence Generation

4.1 Objective

Where we wish to produce a single test sequence that tests each transition of an FSM M using characterising set W and no overlap, we require the use of a test sequence that satisfies the following criterion.

Definition 7 *Given FSM M and characterising set W, let T denote the set of test subsequences for the transitions of M created using W. Thus, for each transition $t = (s, s', x/y)$ of M and $\bar{w}_i \in W$, T contains a transition sequence that has starting state s and label $x\bar{w}_i/y\lambda(s', \bar{w}_i)$. Let $T = \{\bar{\tau}_1, \ldots, \bar{\tau}_p\}$. A test sequence $\bar{x}$ is a* W-test sequence *for FSM M and characterising set W if and only if $\bar{x}$ is the input portion of the label of a transition sequence $\bar{\rho} = \bar{\tau}'_1\bar{\tau}_1\bar{\tau}'_2 \ldots \bar{\tau}_p\bar{\tau}'_{p+1}$, with starting state s_0, for some $\bar{\tau}'_1, \ldots \bar{\tau}'_{p+1}$.*

This criterion may be weakened in the following manner, allowing overlap and for a transition to be followed by a $\bar{w}_i$ sequence rather than $\bar{w}_i$.

Definition 8 *A test sequence $\bar{x}$ is a* W-overlapping test sequence *for FSM M and characterising set W if and only if $\bar{x}$ is the input portion of the label of a transition sequence $\bar{\rho}$ with starting state s_0 that has the property that for every transition t of M and $\bar{w}_i \in W$, $\bar{\rho}$ contains a subsequence $t\bar{\tau}'$ such that $\bar{\tau}'$ is a $\bar{w}_i$ sequence.*

Thus, our problem is to generate a W-overlapping test sequence for FSM M given characterising set W.

4.2 Overview of the algorithm

This section describes an algorithm that produces a W-overlapping test sequence that utilizes the overlap produced by $(\bar{w}_i, \bar{w}_j)$ converting transitions[1]. The first step is to determine which transitions are $(\bar{w}_i, \bar{w}_j)$ converting for each $\bar{w}_i, \bar{w}_j \in W$. The $(\bar{w}_i, \bar{w}_j)$ converting transitions for M_0 can be derived from Table 2 and are shown in Table 3. Here (i, j) in a column associated with state s and in a row with label x denotes that the input of x in state s produces a $(\bar{w}_i, \bar{w}_j)$ converting transition.

The test sequence generation algorithm involves the following steps.

1. Generate a digraph that, for each transition t and $\bar{w}_i \in W$, includes an edge e_t representing t in a context within which it must be followed by a $\bar{w}_i$ sequence in any tour that contains e_t.
2. Produce a minimal tour that contains each of these edges.
3. Start the tour at the initial state to form a walk $\bar{\rho}$ and return the input portion of the label of $\bar{\rho}$.

These steps are now described.

[1] While this paper only considers the use of $(\bar{w}_i, \bar{w}_j)$ converting transitions in test sequence generation it should be possible to utilize sequences that are $(\bar{w}_i, \bar{w}_j)$ converting in a similar way.

	From s_0	From s_1	From s_2	From s_3
$(\bar{w}_1, \_)$ for a	$(1,2)$	$(1,1)$, $(1,2)$	$(1,2)$	$(1,1)$, $(1,2)$
$(\bar{w}_1, \_)$ for b	$(1,1)$	$(1,1)$	$(1,1)$, $(1,2)$	$(1,1)$, $(1,2)$
$(\bar{w}_2, \_)$ for a	$(2,2)$	$(2,1)$, $(2,2)$	$(2,2)$	$(2,1)$, $(2,2)$
$(\bar{w}_2, \_)$ for b	$(2,1)$	$(2,1)$	$(2,1)$, $(2,2)$	$(2,1)$, $(2,2)$

Table 3. The $(\bar{w}_i, \bar{w}_j)$ converting transitions used

4.3 Representing tests

In this subsection we define a digraph $G = (V, E)$ to be used as the basis of test sequence generation. G contains a number of copies of each state of M: one for each element of W and one extra. These copies are labelled: for $\bar{w}_k \in W$ and state s_i the corresponding copy is called v_i^k; the extra copy is called v_i^*.

For each transition t and $\bar{w}_k \in W$ there is an additional vertex v_t^k. This represents the test subsequence formed by following transition t by a $\bar{w}_k$ sequence. Thus G has vertex set $V = V_0 \cup V_1 \cup \ldots \cup V_{n-1} \cup V^{Tr}$, with $V_i = \{v_i^1, \ldots, v_i^m, v_i^*\}$ $(0 \leq i < n, m = |W|)$, and $V^{Tr} = \{v_t^k | t = (s, \delta(s,x), x/\lambda(s,x)), s \in S, x \in X, 1 \leq k \leq m\}$. We now define the edge set $E = E_t \cup E'$ of G.

1. Given $l \in \{1, \ldots, m\}$ and transition $t = (s_i, s_j, x/y)$ there is an edge in E_t from v_t^l to v_j^l representing the use of t to be followed by a $\bar{w}_l$ sequence starting at s_j. This edge has cost 1 and label t.

2. There is an edge (v_0^*, v_0^*, ϵ) in E_t, with cost 0, where ϵ represents null input. The inclusion of this edge guarantees that any tour that contains every edge from E_t passes through v_0^* and thus can be started from the initial state.

3. Given $l \in \{1, \ldots, m\}$ and transition $t = (s_i, s_j, x/y)$ that is $(\bar{w}_l, \bar{w}_k)$ converting, there is an edge in E' from v_i^k to v_t^l with label ϵ. This represents the use of t as a $(\bar{w}_l, \bar{w}_k)$ converting transition: if we pass through this edge in a tour then by construction we must follow it by an edge representing t and then a sequence of edges that represents a $\bar{w}_l$ sequence. This edge has cost 0, since if we pass through this edge then we must pass through the edge from v_t^l, representing the use of t, and this has cost 1.

4. Given $i \in \{0, \ldots, n-1\}$, $k \in \{1, \ldots, m\}$ and $s_j = \delta(s_i, \bar{w}_k)$ there is an edge in E' from v_i^k to v_j^* with label $\bar{w}_k$ representing the execution of $\bar{w}_k$ from s_i. This edge has cost $|\bar{w}_k|$.

5. Given transition $t = (s_i, s_j, x/y)$ there is an edge in E' from v_i^* to v_j^* with label t that represents the execution of t in order to connect transition tests. This edge has cost 1.

6. Given transition $t = (s_i, s_j, x/y)$ and $l \in \{1, \ldots, m\}$ there is an edge in E' from v_i^* to v_t^l with label ϵ and cost 0. These edges are used to guarantee connectivity and to allow a $(\bar{w}_l, \bar{w}_k)$ converting transition to be treated as a normal transition (i.e. not to be involved in overlap).

An edge leaving vertex v_i^k represents either a $\bar{w}_k$ sequence or the beginning of a $\bar{w}_k$ sequence. Thus, an edge leaving v_i^k represents either a transition sequence whose label has input portion $\bar{w}_k$ or a $(\bar{w}_r, \bar{w}_k)$ converting transition, for some $\bar{w}_r \in W$. Vertices of the form v_i^* are included in order to allow the use of transitions that connect the individual test subsequences.

The edge set E_t is the set of edges that must be included in producing a transition sequence. All but one of these edges represent a transition to be followed by a $\bar{w}_k$ sequence (some k). The remainder, from v_0^* to v_0^*, guarantees that any tour that contains every edge from E_t must pass through v_0^*. This latter property is important for two reasons. First, we need a transition sequence that starts at s_0. Second, we cannot start the tour with an edge e that represents a transition being used as $(\bar{w}_i, \bar{w}_j)$ converting (some $\bar{w}_i, \bar{w}_j \in W$) since by starting the tour with e this edge and the following sequence no longer provides a $\bar{w}_j$ sequence for the previous edge.

The digraph, without the edges between the vertices in V^*, produced from M_0 is shown in Figure 2 where if a transition t with starting state s_i is $(\bar{w}_l, \bar{w}_k)$ converting then there is a corresponding edge from v_i^k to v_t^l; these edges are based on Table 2. In this figure, the edges from E_t are represented by thick lines and the edges representing $\bar{w}_1$ and $\bar{w}_2$ are represented by lines with no arrowheads (but all go to vertices of the form v_i^*). Given a transition $t = (s_i, s_j, x/y)$ and $k \in \{1, 2\}$, the vertex v_t^k is denoted v_{ix}^k, since the starting state and input fully define a transition of M_0. In Figure 2, for example, vertex v_{1a}^2 represents the transition with starting state s_1 and input a to be followed by a $\bar{w}_2$ sequence and there is an edge from this vertex to v_2^2 since the ending state of the transition from s_1 with input a is s_2. Further, there is an edge from v_2^2 to v_{2b}^1 since the transition from s_2 with input b is $(\bar{w}_1, \bar{w}_2)$ converting; we can form a $\bar{w}_2$ sequence by following it by a $\bar{w}_1$ sequence.

The next step in test sequence generation is to find a minimum cost tour that contains a copy of each edge from E_t. This problem is an instance of the RCPP and can be solved using the approach discussed in [Aho et al., 1988]. The following proves that there always exist tours that satisfy the test criterion and include the vertex v_0^*.

Proposition 4 *Suppose $v_t^l \in V^{Tr}$. Then G contains a path from v_0^* to v_t^l and a path from v_t^l to v_0^*.*

Proof

Let s_i denote the starting state of t. Observe that G restricted to the vertices in V^* is strongly connected. Also, there is an edge from v_t^l to some v_j^* representing the execution of $\bar{w}_l$ and there is an edge from v_i^* to v_t^l with label ϵ. The result thus follows. □

Once $\bar{\rho}$ has been found, a test sequence can be generated from this by starting it at vertex v_0^* and using the input portion of the label of the corresponding walk.

The overall algorithm can be summarized in the following manner.

Algorithm 1

1. *For each $\bar{w}_i, \bar{w}_j \in W$, state s and input x, determine whether x is $(\bar{w}_i, \bar{w}_j)$ converting from s.*
2. *Devise the digraph G.*
3. *Find a minimum cost tour of G that contains every edge from E_t and start this from v_0^* to form a transition sequence $\bar{\rho}$.*

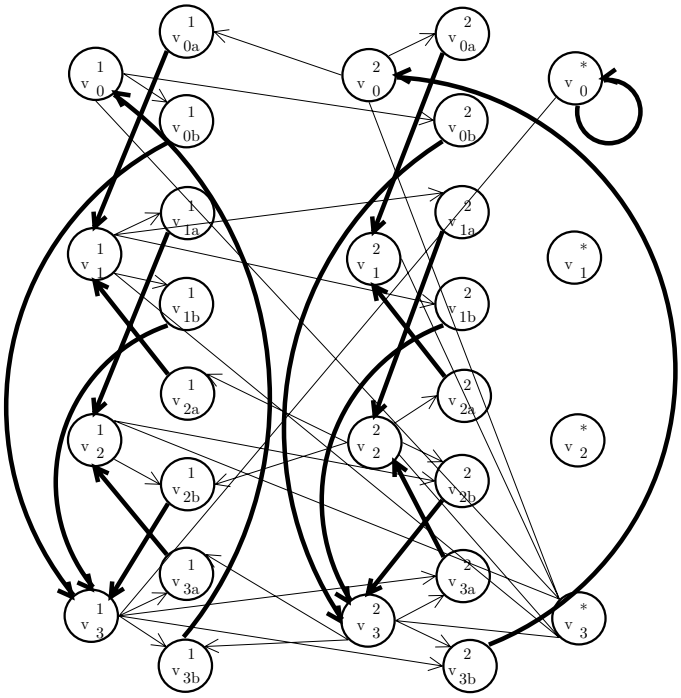

Figure 2. The Digraph G for M_0

4. Extract the test sequence from $\bar{\rho}$.

The following is straightforward to prove.

Proposition 5 *Suppose that FSM M has n states, p inputs, and a characterizing set with m elements. Then the digraph G produced in Algorithm 1 has $O(pnm)$ vertices and $O(pnm^2)$ edges.*

Suppose that G has v vertices and e edges. Then the optimization approach described in [Aho et al., 1988] can be achieved using an algorithm with time complexity $O(ev \log v))$. This gives an overall time complexity of $O(p^2n^2m^3 \log pnm)$

Note that if we do not utilize overlap then we generate the transition sequence from a digraph that has $O(pnm)$ vertices and $O(pnm)$ edges since we have to combine pnm test subsequences to form a single transition sequence. Thus, the use of overlap does have a cost; it increases the test sequence generation effort.

5 Test sequence generation using prefixes

In order to verify a state s_i of M it may be possible to use a set W_i of prefixes of sequences from W with the property that for every state $s_j \neq s_i$ of M, some element of W_i distinguishes s_i from s_j [Fujiwara et al., 1991]. We assume that a set $\mathcal{W} = \{W_0, \ldots, W_{n-1}\}$ is given and $W_i = \{\bar{w}_{i1}, \ldots, \bar{w}_{im_i}\}$ $(0 \leq i < n,\ 1 \leq m_i \leq m)$. Earlier we saw that the following sets can be used for M_0:

Input	State s_0	State s_1	State s_2	State s_3
$\bar{w}_{01}$ (b)	q	q	r	p
$\bar{w}_{02}$ (ab)	pq	pr	pq	rr
$a\bar{w}_{11}$ (aab)	ppr	ppq	ppr	rpq

Table 4. The transition from s_0 with input a

1. $W_0 = \{\bar{w}_{01}, \bar{w}_{02}\}$ where $\bar{w}_{01} = b, \bar{w}_{02} = ab$;
2. $W_1 = \{\bar{w}_{11}\}$ where $\bar{w}_{11} = ab$;
3. $W_2 = \{\bar{w}_{21}\}$ where $\bar{w}_{21} = b$; and
4. $W_3 = \{\bar{w}_{31}\}$ where $\bar{w}_{31} = b$.

If we wish to produce a single test sequence that tests every transition of FSM M using $\mathcal{W}$ and no overlap, we require the use of a test sequence that satisfies the following criterion.

Definition 9 *Given FSM M and set $\mathcal{W}$, let T denote the set of test subsequences for the transitions of M created using $\mathcal{W}$. Thus, for each transition $t = (s_i, s_j, x/y)$ of M and $\bar{w}_{jl} \in W_j$, T contains a subsequence that has starting state s_i and label $x\bar{w}_{jl}/y\lambda(s_j, \bar{w}_{jl})$. Let $T = \{\bar{\tau}_1, \ldots, \bar{\tau}_p\}$. A test sequence $\bar{x}$ is a* $\mathcal{W}$-test sequence *if and only if it is the input portion of the label of a transition sequence $\bar{\rho}$ with starting state s_0 such that $\bar{\rho} = \bar{\tau}'_1\bar{\tau}_1\bar{\tau}'_2 \ldots \bar{\tau}_p\bar{\tau}'_{p+1}$ for some $\bar{\tau}'_1, \ldots \bar{\tau}'_{p+1}$.*

Again, the test criterion may be weakened.

Definition 10 *A test sequence $\bar{x}$ is a* $\mathcal{W}$-overlapping test sequence *for FSM M and set $\mathcal{W}$ if and only if it is the input portion of the label of a transition sequence $\bar{\rho}$ with starting state s_0 such that for every transition $t = (s_i, s_j, x/y)$ of M and $\bar{w}_{jl} \in \mathcal{W}$, $\bar{\rho}$ contains a subsequence $t\bar{\tau}'$ such that $\bar{\tau}'$ is a $\bar{w}_{jl}$ sequence.*

Thus, our problem is to generate a $\mathcal{W}$-overlapping test sequence for M. We can represent overlap in a similar manner to before.

Definition 11 *A transition sequence $\bar{\tau} = (s_i, s_j, \bar{x}/\bar{y})$ is said to be* $(\bar{w}_{jl}, \bar{w}_{ik})$ converting *if, whenever the sequence $\bar{w}_{ik}$ distinguishes a state s' from s_i the sequence $\bar{x}\bar{w}_{jl}$ also distinguished s' from s_i.*

Consider the example M_0. The transition from s_0 with input a ends at state s_1 and thus we have to determine whether this is $(\bar{w}_{11}, \bar{w}_{01})$ converting and whether it is $(\bar{w}_{11}, \bar{w}_{02})$ converting. Here $a\bar{w}_{11} = aab$ and this sequence distinguishes s_0 from both s_1 and s_3 but not from s_2. This is illustrated in Table 4. Thus, the transition from s_0 with input a is $(\bar{w}_{11}, \bar{w}_{02})$ converting.

Similarly, we have that: the transition from s_0 with input b is $(\bar{w}_{31}, \bar{w}_{01})$ converting; the transition from s_1 with input a is $(\bar{w}_{21}, \bar{w}_{11})$ converting; the transition from s_2 with input b is $(\bar{w}_{31}, \bar{w}_{11})$ converting; the transition from s_3 with input a is $(\bar{w}_{21}, \bar{w}_{31})$ converting; and the transition from s_3 with input b is both $(\bar{w}_{01}, \bar{w}_{31})$ converting and $(\bar{w}_{02}, \bar{w}_{31})$ converting.

We are now in a position to adapt the algorithm from Section 4 to use prefixes from sets in $\mathcal{W}$ rather than the entire characterizing set.

Test sequence generation is based on a digraph $G_{\mathcal{W}}$ that has a number of copies of each state of M: for state s_i there one copy for each element of W_i and one extra. These copies are labelled in the following way: for $\bar{w}_{ik} \in W$ and state s_i the corresponding copy is called v_i^k; the extra copy is called v_i^*.

For each transition $t = (s_i, s_j, x/y)$ and $\bar{w}_{jk} \in W_j$ there is a vertex v_t^k that represents t to be followed by a $\bar{w}_{jk}$ sequence. Thus $G_{\mathcal{W}}$ has vertex set $V = V_0 \cup V_1 \cup \ldots \cup V_{n-1} \cup V^{Tr}$, with $V_i = \{v_i^1, \ldots, v_i^{m_i}, v_i^*\}$ $(0 \leq i < n)$, and $V^{Tr} = \{v_t^k | t = (s_i, s_j, x/y), 1 \leq k \leq m_j\}$. The edge set $E_{\mathcal{W}} = E_{t\mathcal{W}} \cup E'_{\mathcal{W}}$ of $G_{\mathcal{W}}$ is now defined[2].

1. Given $l \in \{1, \ldots, m_j\}$ and transition $t = (s_i, s_j, x/y)$ there is an edge in $E_{t\mathcal{W}}$ from v_t^l to v_j^l with label t representing the use of t to be followed by an $\bar{w}_{jl}$ sequence. This edge has cost 1.

2. There is an edge (v_0^*, v_0^*, ϵ) in $E_{t\mathcal{W}}$, with cost 0. The inclusion of this edge guarantees that any tour that contains the edges from $E_{t\mathcal{W}}$ passes through v_0^* and thus can be started from the initial state.

3. Given $l \in \{1, \ldots, m_j\}$ and transition $t = (s_i, s_j, x/y)$ that is $(\bar{w}_{jl}, \bar{w}_{ik})$ converting, there is an edge in $E'_{\mathcal{W}}$ from v_i^k with label ϵ to v_t^l. This represents the use of t as a $(\bar{w}_{jl}, \bar{w}_{ik})$ converting transition. This edge has cost 0.

4. Given $i \in \{0, \ldots, n-1\}$, $k \in \{1, \ldots, m_i\}$ and $s_j = \delta(s_i, \bar{w}_{ik})$ there is an edge in $E'_{\mathcal{W}}$ from v_i^k to v_j^* with label $\bar{w}_{ik}$ representing the input of $\bar{w}_{ik}$ from s_i. This edge has cost $|\bar{w}_{ik}|$.

5. Given transition $t = (s_i, s_j, x/y)$ there is an edge in $E'_{\mathcal{W}}$ from v_i^* to v_j^* with label t that represents the use of t in order to connect tests. This edge has cost 1.

6. Given transition $t = (s_i, s_j, x/y)$ and $l \in \{1, \ldots, m_j\}$ there is an edge in $E'_{\mathcal{W}}$ from v_i^* to v_t^l with label ϵ and cost 0.

The digraph, without the edges between the vertices in V^*, produced from M_0 is shown in Figure 3. In this figure, the edges from $E_{\mathcal{W}t}$ are represented by thick lines and the edges representing sequences from the W_i are represented by lines with no arrowheads (but all go to vertices in V^*). Given a transition $t = (s_i, s_j, x/y)$, the vertex v_t^k is denoted v_{ix}^k, since the starting state and input fully define a transition of M_0.

The next step in test sequence generation involves finding a minimum cost tour that contains a copy of each edge from $E_{t\mathcal{W}}$. As before, connectivity is ensured. Once $\bar{\tau}$ has been found, a test sequence can be generated from this by starting $\bar{\tau}$ at vertex v_0^* and using the input portion of the label of the corresponding walk $\bar{\rho}$.

[2]The structures of G and $G_{\mathcal{W}}$ are very similar, the difference being the use of $\bar{w}_i$ sequences in G and $\bar{w}_{jl}$ sequences in $G_{\mathcal{W}}$

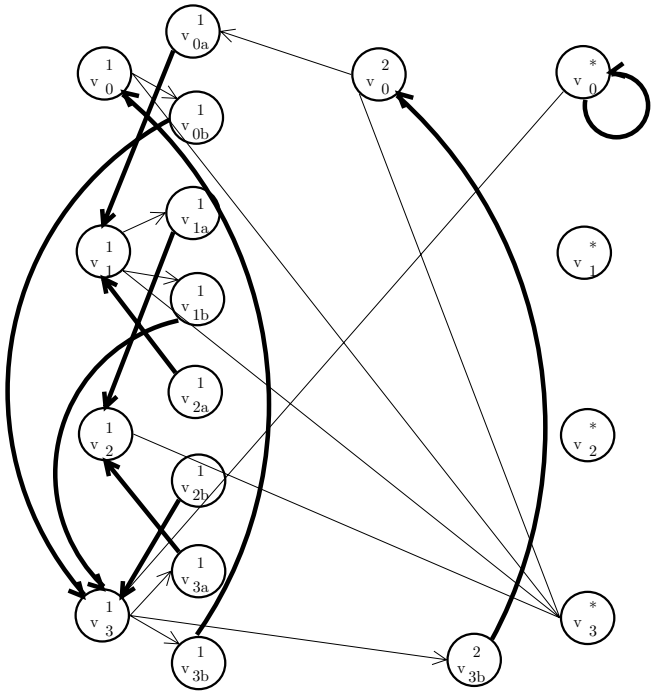

Figure 3. The Digraph $G_{\mathcal{W}}$ for M_0

The overall algorithm can be summarized in the following manner.

Algorithm 2

1. *For each $\bar{w}_{jl} \in W_j$, $\bar{w}_{ik} \in W_i$ such that there is a transition t from s_i to s_j, determine whether t is $(\bar{w}_{jl}, \bar{w}_{ik})$ converting.*
2. *Devise the digraph $G_{\mathcal{W}}$.*
3. *Find a minimum cost tour of $G_{\mathcal{W}}$ that contains every edge from $E_{t\mathcal{W}}$ and start this at v_0^* to form a transition sequence $\bar{\rho}$.*
4. *Extract the test sequence from $\bar{\rho}$.*

Proposition 6 *Suppose that FSM M has n states, p inputs, and a characterizing set with m elements. Then the digraph $G_{\mathcal{W}}$ produced in Algorithm 2 has $O(pnm)$ vertices and $O(pnm^2)$ edges.*

6 Evaluation

The test sequence generation algorithms given in this paper extend the applicability of test overlap. They differ from the approaches, for using overlap with UIOs or a DS, in scope: they generalise the problem to the case where we are not using either a DS or a set of UIOs by allowing overlap where separating sequences are used. This is

useful since, while every minimal FSM has a characterizing set, a minimal FSM need not have a distinguishing sequence or a UIO for each state.

First consider the case where prefixes of the sequences in W are not used. Optimization occurs over a set of tours of a digraph G. Importantly, this set of tours includes those that represent test sequences that do not utilize overlap.

Proposition 7 *Every W-test sequence for M is represented by a tour through G that contains each edge from E_t.*

The following shows that the test sequence produced using overlap will be no longer than any produced without overlap.

Theorem 8 *Suppose test sequence $\bar{x}$ is generated from some shortest tour of G that contains every element of E_t. Then no W-test sequence is shorter than $\bar{x}$.*

Proof

This is an immediate consequence of Proposition 7. □

We now apply the algorithm given in Section 4 to the FSM M_0 given in Figure 1. Recall that the $(\bar{w}_i, \bar{w}_j)$ converting transitions are shown in Table 3. The digraph G leads to the following tour.

$$v_0^* \to v_{0a}^1 \to v_1^1 \to v_{1b}^1 \to v_3^1 \to v_{3a}^2 \to v_2^2 \to v_{2a}^2 \to v_1^2 \to v_{1a}^2 \to v_2^2 \to v_{2a}^1 \to$$

$$v_1^1 \to v_{1b}^2 \to v_3^2 \to v_{3b}^1 \to v_0^1 \to v_{0b}^2 \to v_3^2 \to v_{3b}^2 \to v_0^2 \to v_{0a}^2 \to v_1^2 \to$$

$$v_{1a}^1 \to v_2^1 \to v_{2b}^2 \to v_3^2 \to v_{3a}^1 \to v_2^1 \to v_{2b}^1 \to v_3^1 - \bar{w}_1 \to v_0^* \to v_{0b}^1 \to v_3^1 - \bar{w}_1 \to v_0^*$$

This leads to the following test sequence of length 18: $abaaaabbbbaababw_1bw_1$. It is easy to check that every transition is tested twice, once by a $\bar{w}_1$ sequence and once by a $\bar{w}_2$ sequence. If overlap is not used, the test sequence is guaranteed to have length at least that of the total for the individual tests in T. Since M has $|X||S|$ transitions, this is $|X||S|\sum_i(|\bar{w}_i| + 1)$. In this case, we get the value 40: any test sequence generated without overlap must have length *at least* 40. Overlap has therefore reduced the test sequence length by a factor in excess of 2. We have thus shown that the use of overlap produces a test sequence no longer than that produced without overlap and is capable of producing a test sequence that is shorter than that generated without overlap.

Now consider the use of prefixes in state verification. We get results that are equivalent to those for the use of a full characterizing set.

Theorem 9 *Suppose that the set $\mathcal{W}$ is used for state verification and that test sequence $\bar{x}$ is generated from some shortest tour of $G_{\mathcal{W}}$ that contains every element of E_t. Then no $\mathcal{W}$-test sequence is shorter than $\bar{x}$.*

Now consider the use of $\mathcal{W}$ and overlap in testing from M_0. The following is a tour of $G_{\mathcal{W}}$ that includes every edge from $E_{t\mathcal{W}}$.

$$v_0^* \rightarrow v_0^* - a/p \rightarrow v_1^* - a/p \rightarrow v_2^* \rightarrow v_{2a}^1 \rightarrow v_1^1 \rightarrow v_{1a}^1 \rightarrow v_2^1 - \bar{w}_{21} \rightarrow v_3^* \rightarrow v_{3a}^1 \rightarrow v_2^1 - \bar{w}_{21} \rightarrow v_3^* - a/r \rightarrow v_2^* \rightarrow$$

$$v_{2b}^1 \rightarrow v_3^1 \rightarrow v_{3b}^1 \rightarrow v_0^1 \rightarrow v_{0b}^1 \rightarrow v_3^1 \rightarrow v_{3b}^2 \rightarrow v_0^2 \rightarrow v_{0a}^1 \rightarrow v_1^1 \rightarrow v_{1b}^1 \rightarrow v_3^1 - \bar{w}_{31} \rightarrow v_0^*$$

In addition to the transitions being tested, this includes three connecting transitions and three separating sequences (all of length one). Thus the overall test sequence length is the number of transitions counting each transition to state s_0 twice as $|W_0| = 2$ (nine) plus the number of connecting transitions (three) plus the total length of the separating sequences used (three) and so is 15. We get the following test sequence: $aaaa\bar{w}_{21}a\bar{w}_{21}abbbbab\bar{w}_{31}$. If overlap is not used, the test sequence is guaranteed to have length at least that of the total for the individual test subsequences in T. In this case, we get the value 21: any test sequence generated without overlap must have length *at least* 21. The use of overlap thus led to a reduction in the test sequence length. Interestingly, in this case the test sequence produced using overlap and the entire set W is shorter than that produced using prefixes ($\mathcal{W}$) but no overlap.

7 Conclusions

When testing from an FSM, overlap may be used in order to reduce the test effort. While previous work has considered the case when a DS or a set of UIOs are used for state checking, an FSM need not have such sequences. However, every minimal FSM has a characterizing set. This paper has thus considered the problem of utilizing overlap where a characterizing set is used for state checking. This paper has generalized the notion of invertibility, used for UIOs, to $(\bar{w}_i, \bar{w}_j)$ converting transitions. This allowed test subsequence overlap to be represented when a characterizing set is used for state checking.

Having studied test subsequence overlap we considered the problem of generating a test sequence, from an FSM M, in the presence of a characterizing set W. We considered the test criterion that for each transition t and $\bar{w}_i \in W$, the test sequence contains t followed by a sequence that has at least the ability of $\bar{w}_i$ to distinguish the final state of t from other states of M. Overlap was utilized in order to reduce the test sequence length.

The test sequence generation algorithm presented is based around a digraph G that represents possible test sequences. The problem reduces to that of finding a minimal tour in G that contains the edges in some set E_t and thus becomes an instance of the Rural Chinese Postman Problem. Importantly, any test sequence that does not utilize overlap is also represented by a tour of G that contains the edges in E_t and thus the optimal test sequence, produced by the proposed algorithm, can be no longer than that produced without overlap. We have also shown that the proposed test generation algorithm is capable of leading to a reduction in the length of the resultant test sequence.

Sometimes it is sufficient to apply only a set W_i of prefixes of sequences in W in order to identify a state s_i. We thus extended the test sequence generation algorithm to using a set $\mathcal{W} = \{W_0, \ldots, W_{n-1}\}$ of sets for identifying

the states $s_0, \ldots, s_{n-1}$ of M. The approaches to representing overlap and generating a test sequence extended naturally to the use of $\mathcal{W}$ rather than W. Test sequence generation was again represented as the problem of finding a minimal tour of a digraph $G_{\mathcal{W}}$ that contains some set $E_{t\mathcal{W}}$ of edges. Any test sequence that uses $\mathcal{W}$ but does not utilize overlap is also represented by a tour of $G_{\mathcal{W}}$ that contains the edges in $E_{t\mathcal{W}}$ and thus the optimal test sequence produced by the proposed algorithm can be no longer than that produced without overlap. Again, when applied to an example the use of overlap led to a reduction in the test sequence length.

While the use of overlap can lead to shorter test sequences it increases the cost of test sequence generation. In practice there will be a trade-off between the cost of test sequence generation and the cost of test sequence execution. In some cases the use of overlap will not be justified. However, the use of overlap will be more appealing if test execution is expensive or the test sequence is to be applied to many implementations.

References

[Aho et al., 1988] Aho, A. V., Dahbura, A. T., Lee, D., and Uyar, M. U. (1988). 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. Elsevier (North–Holland).

[Broekman and Notenboom, 2003] Broekman, B. and Notenboom, E. (2003). *Testing Embedded Software*. Addison–Wesley, London.

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

[Derrick and Boiten, 1999] Derrick, J. and Boiten, E. (1999). Testing refinements of state–based formal specifications. *Journal of Software Testing, Verification and Reliability*, 9(1):27–50.

[Dick and Faivre, 1993] Dick, J. and Faivre, A. (1993). Automating the generation and sequencing of test cases from model–based specifications. In *FME '93, First International Symposium on Formal Methods in Europe*, pages 268–284, Odense, Denmark. Springer–Verlag, Lecture Notes in Computer Science 670.

[Duale and Uyar, 2004] Duale, A. Y. and Uyar, M. U. (2004). A method enabling feasible conformance test sequence generation for EFSM models. *IEEE Transactions on Computers*, 53(5):614–627.

[Farchi et al., 2002] Farchi, E., Hartman, A., and Pinter, S. (2002). Using a model-based test generator to test for standard conformance. *IBM systems journal*, 41(1):89–110.

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

[Gill, 1962] Gill, A. (1962). *Introduction to The Theory of Finite State Machines*. McGraw–Hill, New York.

[Grieskamp et al., 2002] Grieskamp, W., Gurevich, Y., Schulte, W., and Veanes, M. (2002). Generating finite state machines from abstract state machines. In *Proceedings of the ACM SIGSOFT Symposium on Software Testing and Analysis*, pages 112–122.

[Hennie, 1964] Hennie, F. C. (1964). Fault–detecting experiments for sequential circuits. In *Proceedings of Fifth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 95–110, Princeton, New Jersey.

[Hierons, 1992] Hierons, R. M. (1992). *Using Formal Specifications to Enhance the Software Testing Process*. PhD thesis, Brunel University.

[Hierons, 1996] Hierons, R. M. (1996). Extending test sequence overlap by invertibility. *The Computer Journal*, 39(4):325–330.

[Hierons, 1997] Hierons, R. M. (1997). Testing from a finite state machine: Extending invertibility to sequences. *The Computer Journal*, 40(4):220–230.

[Hierons, 2004] Hierons, R. M. (2004). Minimizing the number of resets when testing from a finite state machine. *Information Processing Letters*, 90(6):287–292.

[Hierons and Ural, 2002] Hierons, R. M. and Ural, H. (2002). Reduced length checking sequences. *IEEE Transactions on Computers*, 51(9):1111–1117.

[Hopcroft, 1971] Hopcroft, J. E. (1971). An n log n algorithm for minimizing the states in a finite automaton. In Kohavi, Z., editor, *The theory of Machines and Computation*, pages 189–196. Academic Press.

[Kohavi, 1978] Kohavi, Z. (1978). *Switching and Finite State Automata Theory*. McGraw–Hill, New York.

[Lee and Yannakakis, 1994] Lee, D. and Yannakakis, M. (1994). Testing finite-state machines: State identification and verification. *IEEE Transactions on Computers*, 43(3):306–320.

[Miller and Paul, 1993] Miller, R. E. and Paul, S. (1993). On the generation of minimal length conformance tests for communications protocols. *IEEE/ACM Transactions on Networking*, 1(1):116–129.

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

[Motteler et al., 1994] Motteler, H., Chung, A., and Sidhu, D. (1994). Fault coverage of UIO-based methods for protocol testing. In *Proceedings of Protocol Test Systems VI*, pages 21–33.

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

[Pomeranz and Reddy, 1997] Pomeranz, I. and Reddy, S. M. (1997). Test generation for multiple state–table faults in finite–state machines. *IEEE Transactions on Computers*, 46(7):783–794.

[Rezaki and Ural, 1995] Rezaki, A. and Ural, H. (1995). Construction of checking sequences based on characterization sets. *Computer Communications*, 18(12):911–920.

[Shen et al., 1990] Shen, Y. N., Lombardi, F., and Dahbura, A. T. (1990). Protocol conformance testing using multiple UIO sequences. In *Proceedings of Protocol Specification, Testing, and Verification IX*, pages 131–143, Twente, Netherlands. North–Holland.

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

[Ural et al., 1997] Ural, H., Wu, X., and Zhang, F. (1997). On minimizing the lengths of checking sequences. *IEEE Transactions on Computers*, 46(1):93–99.

[Ural and Zhu, 1993] Ural, H. and Zhu, K. (1993). Optimal length test sequence generation using distinguishing sequences. *IEEE/ACM Transactions on Networking*, 1(3):358–371.

[Yang and Ural, 1990] Yang, B. and Ural, H. (1990). Protocol conformance test generation using multiple UIO sequences with overlapping. In *ACM SIGCOMM 90: Communications, Architectures, and Protocols*, pages 118–125, Twente, The Netherlands.

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

[Zhang and Probert, 2005] Zhang, F. and Probert, R. L. (2005). Minimizing the lengths of test sequences with overlapping. In *Intrumentation and Measurement Technology Conference (IMTC)*, Ottawa, Canada.