Concerning the Ordering of Adaptive Test Sequences

Robert M. Hierons[1] and Hasan Ural[2]

[1] Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK
[2] School of Information Technology and Engineering, Faculty of Engineering, University of Ottawa, 800 King Edward Avenue, Ottawa, Ontario, K1N 6N5, Canada

Abstract. The testing of a state-based system may involve the application of a number of adaptive test sequences. Where the implementation under test (IUT) is deterministic, the response of the IUT to some adaptive test sequence γ_1 may be capable of determining the response of the IUT to some other adaptive test sequence γ_2. Thus, the expected cost of applying a set of adaptive test sequences depends upon the order in which they are applied. This paper explores properties of adaptive test sequences and the problem of finding an order of application, of the elements from some set of adaptive test sequences, that minimises the expected cost of testing.

1 Introduction

Where a system is state-based, testing involves the application of sequences of input values. Such sequences are called test sequences. A test sequence, or a set of test sequences, may represent some test purpose or test objective.

An adaptive test sequence is applied in an adaptive experiment which is a process in which at each stage the input applied depends upon the output that has been produced. An adaptive test typically consists of a number of adaptive test sequences. The use of adaptive tests has been proposed within the area of testing from a (possibly non-deterministic) finite state machine (see, for example, [1, 4, 7, 9, 10]), and generally in protocol conformance testing [6]. Further, algorithms for generating tests from a specification written in a Process Algebra such as LOTOS typically produce adaptive test sequences with verdicts (see, for example, [2, 8]). Adaptivity is thus a core element of the test description language TTCN (see, for example, [3]). Adaptivity may lead to more efficient tests. For example, when testing against a non-deterministic finite state machine there may be no single preset input sequence that reaches a given state s and thus a set of input sequences might be used to reach s. Instead, it may be possible to apply a single adaptive test sequence to reach s.

When the *implementation under test (IUT)* is deterministic, the response to one adaptive test sequence γ_1 may be capable of fully deciding the response of the IUT to some other adaptive test sequence γ_2. Where this is the case, using γ_1 before γ_2 may reduce the expected test execution effort. However, relationships involving other adaptive test sequences may affect the best relative ordering of γ_1 and γ_2. This paper considers the problem of finding an order of execution, of a set of adaptive test sequences, that maximises the expected saving where the IUT is known to be deterministic. Such an ordering is said to be an *optimal ordering.*

This paper makes a number of contributions. First, we make the observation that the order in which adaptive test sequences are applied may affect the expected cost of testing. We formalise this notion and define a function that determines whether the execution of one adaptive test sequence γ_1 is capable of removing the necessity to apply another adaptive test sequence γ_2. We then show how the scale of the optimisation problem may be reduced. Finally, we introduce an algorithm that produces the optimal ordering where the problem has particular properties. Future work will consider how this algorithm may be generalised.

This paper is structured as follows. Section 2 describes test sequences, directed graphs, and adaptive test sequences. Section 3 then considers conditions under which the relative ordering of adaptive test sequences may be significant. Section 4 considers two ways in which the optimisation problem may be simplified. Section 5 introduces a polynomial time algorithm that generates the optimal ordering under a well-defined condition. Section 6 describes potential future work and finally, in Section 7 conclusions are drawn.

2 Preliminaries

2.1 Sequences

Throughout this paper X and Y will denote the input and output domains of the IUT. Given a set A, A^* will denote the set of sequences of elements from A, including the empty sequence ϵ. It will be assumed that the IUT is a state-based system and thus that testing leads to the observation of input/output sequences of the form $\langle x_1/y_1, \ldots, x_k/y_k \rangle \in (X/Y)^*$ where $x_1, \ldots, x_k \in X$ and $y_1, \ldots, y_k \in Y$. A preset input sequence is some element of X^*.

Given sequences c and d, cd will denote the result of concatenating c and d. For example, $\langle a/0 \rangle \langle a/1, b/0 \rangle = \langle a/0, a/1, b/0 \rangle$. Given sets C and D of sequences, CD will denote the set formed by concatenating the elements of C with the elements of D. Thus $CD = \{cd | c \in C \wedge d \in D\}$. Given a sequence $b \in A^*$, $pre(b)$ will denote the set of prefixes of b. Given a set B of sequences $Pre(B)$ will denote the set of prefixes of sequences from B. These are defined more formally by the following.

Definition 1. *Given sequence $b \in A^*$ and set $B \subseteq A^*$:*

$$pre(b) = \{b' \in A^* | \exists b'' \in A^*. b = b'b''\}$$

$$Pre(B) = \bigcup_{b \in B} pre(b)$$

If a set of adaptive test sequences is applied then the IUT is returned to its initial state after each test using a test postamble [5]. This ensures that each test is applied in the same state of the IUT. The postamble might involve some sequence of inputs or a single action such as a disconnect or reset, for a connection-oriented communications protocol, or the system being switched off and then on again.

2.2 Directed Graphs

A directed graph is a set of vertices with arcs between them. The following is a more formal definition.

Definition 2. *A* directed graph (digraph) *G is defined by a pair (V, E) in which $V = \{v_1, \ldots, v_n\}$ is a finite set of vertices and $E \subseteq V \times V$ is a set of directed edges between the vertices of G. An element $e = (v_i, v_j) \in E$ represents an edge from v_i to v_j. Given edge $e = (v_i, v_j)$, $start(e)$ denotes v_i and $end(e)$ denotes v_j.*

Definition 3. *Given digraph $G = (V, E)$ and vertex $v \in V$, $indegree_E(v)$ denotes the number of edges from E that end at v and $outdegree_E(v)$ denotes the number of edges from E that start at v.*

Given a digraph $G = (V, E)$, a sequence $e_1, \ldots, e_m$ of edges from E, in which for all $1 \leq k < m$ $end(e_k) = start(e_{k+1})$, is a *path* from $start(e_1)$ to $end(e_m)$ in G. A path is a *cycle* if its initial and final vertices are the same and no other vertex is repeated. A digraph is said to be *acyclic* if it has no cycles.

Given a digraph G, the following defines the result of removing one or more vertices from G.

Definition 4. *Let $G = (V, E)$ denote a digraph and $V' \subseteq V$. Then $G \setminus V'$ denotes the digraph formed by removing every vertex contained in V', and the associated edges, from G. This is formally defined by the following.*

$$G \setminus V' = (V \setminus V', \{(v_i, v_j) \in E | v_j \notin V' \wedge v_i \notin V'\})$$

2.3 Adaptive Test Sequences

Testing typically involves applying input sequences to the IUT and observing the output produced. Sometimes the input sequences used are preset: each is fully determined before it is applied. However, a test may be adaptive: the next input provided in a sequence may depend on the outputs produced in response to the previous input values. Such a test is called an *adaptive test sequence*. The use of adaptive test sequences has been proposed in a number of areas including protocol conformance testing (see, for example, [1, 6, 9]).

In this paper $\mathcal{T}$ will denote the set of all adaptive test sequences that use inputs from X and refer to outputs from Y. The set $\mathcal{T}$ may be defined recursively in the following manner.

Definition 5. *Each element $\gamma \in \mathcal{T}$ is one of:*

- *null*
- *a pair (x, f) in which $x \in X$ and f is a function from Y to $\mathcal{T}$.*

An adaptive test sequence γ is applied in the following manner. If $\gamma = null$ then the adaptive test sequence ends. If $\gamma = (x, f)$ then x is applied and the output y is observed. The adaptive test sequence $f(y)$ is then applied.

We will assume that the adaptive test sequences are finite. An adaptive test sequence may be represented by a tree. For example, the tree in Figure 1 represents an adaptive test sequence in which the first input is a, no further input is provided if the output is 0, and the input b is provided if the output is 1. Whatever the response to b after a, the test then terminates. The adaptive test sequence γ given in Figure 1 may be defined in the following way: $\gamma = (a, f)$, $f(0) = null$, $f(1) = (b, f')$, $f'(0) = null$, and $f'(1) = null$.

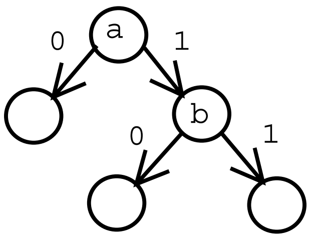

Fig. 1. An adaptive test sequence

Given $\gamma \in \mathcal{T}$ it is possible to define the set $IO(\gamma)$ of input/output sequences that may be observed using γ.

Definition 6. *Given $\gamma \in \mathcal{T}$,*

$$IO(\gamma) = \begin{cases} \{\epsilon\} & \textit{if } \gamma = null \\ \bigcup_{y \in Y} \{\langle x/y \rangle\} IO(f(y)) & \textit{if } \gamma = (x, f) \end{cases}$$

The first rule states that if $\gamma = null$ then no input/output behaviour is seen and thus the empty sequence is observed. The second rule is recursive, stating that given $y \in Y$, γ may lead to an input/output behaviour in the form of x/y followed by some input/output behaviour formed by then applying $f(y)$. For example, the adaptive test sequence γ given in Figure 1 has: $IO(\gamma) = \{\langle a/0 \rangle, \langle a/1, b/0 \rangle, \langle a/1, b/1 \rangle\}$. Since every adaptive test sequence has finite length, it follows that $IO(\gamma)$ is finite and every element of $IO(\gamma)$ is finite.

Definition 7. *The length of $\gamma \in \mathcal{T}$ is defined by:*

$$length(\gamma) = \begin{cases} 0 & \textit{if } \gamma = null \\ 1 + max_{y \in Y}\ length(f(y)) & \textit{if } \gamma = (x, f) \end{cases}$$

The length of an adaptive test sequence γ is thus the length of the longest input/output sequence that may occur through the application of γ.

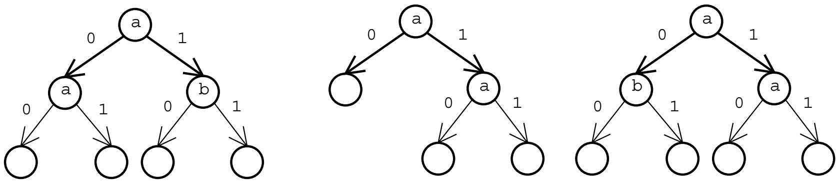

Fig. 2. Adaptive test sequences γ_1, γ_2, and γ_3

3 Conditions under Which Savings Occur

This section will explore conditions under which it may not be necessary to use $\gamma_2 \in \mathcal{T}$ if we first apply $\gamma_1 \in \mathcal{T}$. Consider adaptive test sequences γ_1 and γ_2 in Figure 2. Suppose the use of γ_1 leads to the input/output sequence $a/0, a/1$. Then, since the IUT is deterministic we know that the response of the IUT to γ_2 will be $a/0$. Thus there is no need to apply γ_2.

Suppose we intend to execute some adaptive test sequence γ_2 after some adaptive test sequence γ_1, γ_1 being followed by a postamble that returns the system to its initial state. Then the response of the IUT to γ_1 might determine the response of the IUT to γ_2 if one of the possible responses to γ_2 is a prefix of some possible response to γ_1. The condition under which this may happen will now be defined.

Proposition 1. *The response of the IUT to $\gamma_1 \in \mathcal{T}$ may determine the response of the IUT to $\gamma_2 \in \mathcal{T}$ if there is some $x_1/y_1 \in IO(\gamma_1)$ and $x_2/y_2 \in IO(\gamma_2)$ such that $x_2/y_2 \in pre(x_1/y_1)$.*

Definition 8. *If there is some $x_1/y_1 \in IO(\gamma_1)$ and $x_2/y_2 \in IO(\gamma_2)$, $\gamma_1, \gamma_2 \in \mathcal{T}$, such that $x_2/y_2 \in pre(x_1/y_1)$ then we write $\gamma_2 \leq \gamma_1$.*

Note that $\leq$ is not an ordering: there are distinct adaptive test sequences γ_1 and γ_2 such that $\gamma_2 \leq \gamma_1$ and $\gamma_1 \leq \gamma_2$. Such a case is illustrated in Figure 3.

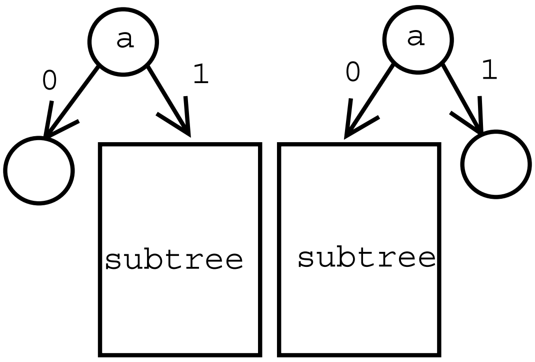

Fig. 3. Related adaptive test sequences

Proposition 2. *The relation $\leq$ is reflexive but in general it is not symmetric and is not transitive.*

Proof

The fact that $\leq$ is reflexive is an immediate consequence of the definition. γ_1 and γ_2 in Figure 2 demonstrate that $\leq$ is not symmetric.

In order to see that $\leq$ is not transitive consider the example in Figure 2. It is straightforward to show that $\gamma_2 \leq \gamma_1$, $\gamma_3 \leq \gamma_2$ but $\gamma_3 \not\leq \gamma_1$. □

Clearly, if $\gamma_2 \leq \gamma_1$ but $\gamma_1 \not\leq \gamma_2$ it may make sense to use γ_1 before γ_2. The function *sav*, defined below, may be used to decide whether $\gamma_1 \leq \gamma_2$.

Definition 9.

$$\begin{aligned} sav(\gamma, null) &= true \\ sav(null, (x, f)) &= false \\ sav((x_1, f_1), (x_2, f_2)) &= (x_1 = x_2) \wedge \\ &\quad \exists y \in Y.sav(f_1(y), f_2(y)) \end{aligned}$$

The following result is clear.

Proposition 3. *Given $\gamma_1, \gamma_2 \in \mathcal{T}$, $\gamma_2 \leq \gamma_1$ if and only if $sav(\gamma_1, \gamma_2)$.*

Based on the relation $\leq$ it is possible to define a digraph.

Definition 10. *Given some set $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ of adaptive test sequences, the* dependence digraph *is the digraph $G = (V, E)$ in which $V = \{v_1, \ldots, v_n\}$ and there is an edge (v_i, v_j) in E if and only if $\gamma_j \leq \gamma_i$ and $\gamma_j \neq \gamma_i$.*

4 Simplifying the Optimisation Problem

This section describes two ways of reducing the scale of the optimisation problem and proves that these approaches do not conflict: by applying one approach we do not reduce the scope for applying the other approach.

4.1 Merging Adaptive Test Sequences

Given $\gamma_1, \gamma_2 \in \mathcal{T}$, it may be possible to combine γ_1 and γ_2 to form one adaptive test sequence. This may reduce the size of the optimisation problem. This section considers conditions under which this may be done and defines an algorithm that merges adaptive test sequences.

Consider, for example, the adaptive test sequences given in Figure 4. Here γ_3 is the result of merging γ_1 and γ_2. By applying γ_3 to a deterministic implementation we get exactly the same information as if we had separately applied γ_1 and γ_2 to some implementation since:

1. Each possible response to γ_3 is a possible response to one of γ_1 and γ_2; and
2. Each possible response to one of γ_1 and γ_2 is a prefix of some possible response to γ_3.

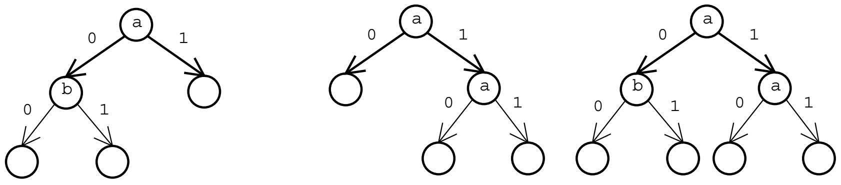

Fig. 4. Merging γ_1 and γ_2 to form γ_3

Before considering how adaptive test sequences may be merged we will define what it means for one adaptive test sequence to be smaller than or equal to another.

Definition 11. *$\gamma_1 \in \mathcal{T}$ is* smaller than or equal to *$\gamma_2 \in \mathcal{T}$ if and only if $IO(\gamma_1) \subseteq Pre(IO(\gamma_2))$.*

This says that all behaviours observable under γ_1 are also observable under γ_2.

Definition 12. *$\gamma_3 \in \mathcal{T}$ is the result of* merging *$\gamma_1 \in \mathcal{T}$ and $\gamma_2 \in \mathcal{T}$ if and only if the following hold:*

1. *$IO(\gamma_1) \cup IO(\gamma_2) \subseteq Pre(IO(\gamma_3))$ and*
2. *for every $\gamma_4 \in \mathcal{T}$, if $IO(\gamma_1) \cup IO(\gamma_2) \subseteq Pre(IO(\gamma_4))$ then γ_3 is smaller than or equal to γ_4.*

The first element of this definition insists that all behaviours observable under γ_1 and all behaviours observable under γ_2 are also observable under γ_3. The second element says that γ_3 is the smallest adaptive test sequence that has this property.

Sufficient and necessary conditions, for it to be possible to merge two adaptive test sequences, will be defined in terms of the notion of two adaptive test sequences being compatible.

Definition 13. *$\gamma_1, \gamma_2 \in \mathcal{T}$ are said to be* compatible *if and only if there are no pairs of sequences $\langle x_1/y_1, x_2/y_2, \ldots, x_{k-1}/y_{k-1}, x_k/y_k\rangle \in Pre(IO(\gamma_1))$ and $\langle x_1/y_1, x_2/y_2, \ldots, x_{k-1}/y_{k-1}, x'_k/y'_k\rangle \in Pre(IO(\gamma_2))$ with $x_k \neq x'_k$ $(k \geq 1)$.*

Thus, γ_1 and γ_2 are compatible if there does not exist an input/output sequence x/y that might result from both γ_1 and γ_2 such that γ_1 and γ_2 apply different input values after x/y. It should be clear that in order to be able to merge two adaptive test sequences γ_1 and γ_2 it is necessary for them to be compatible: otherwise the resultant adaptive test sequence γ_3 must be able to provide two different input values after some input/output sequence x/y. This is not allowed by the definition of an adaptive test sequence.

The function *compatible*, defined below, decides whether two adaptive test sequences are compatible.

Definition 14.

$$\begin{aligned}
\mathit{compatible}(\gamma, \mathit{null}) &= \mathit{true} \\
\mathit{compatible}(\mathit{null}, \gamma) &= \mathit{true} \\
\mathit{compatible}((x_1, f_1), (x_2, f_2)) &= (x_1 = x_2) \wedge \forall y \in Y. \\
&\qquad \mathit{compatible}(f_1(y), f_2(y))
\end{aligned}$$

Proposition 4. *$\gamma_1, \gamma_2 \in \mathcal{T}$ are compatible if and only if $\mathit{compatible}(\gamma_1, \gamma_2) = \mathit{true}$.*

Given compatible γ_1 and γ_2, the following algorithm merges γ_1 and γ_2.

Definition 15.

$$\begin{aligned}
\mathit{merge}(\gamma, \mathit{null}) &= \gamma \\
\mathit{merge}(\mathit{null}, \gamma) &= \gamma \\
\mathit{merge}((x, f_1), (x, f_2)) &= (x, \{y \to \mathit{merge}(f_1(y), f_2(y)) \\
&\qquad |y \in Y\})
\end{aligned}$$

Proposition 5. *If $\gamma_1 \in \mathcal{T}$ and $\gamma_2 \in \mathcal{T}$ are compatible then $\mathit{merge}(\gamma_1, \gamma_2)$ is the result of merging γ_1 and γ_2.*

Proof

First observe that, by definition, if γ_1 and γ_2 are compatible then $\mathit{merge}(\gamma_1,\gamma_2)$ is defined. Let $\gamma_3 = \mathit{merge}(\gamma_1, \gamma_2)$.

Clearly every sequence in $IO(\gamma_1) \cup IO(\gamma_2)$ is a prefix of some sequence in $IO(\gamma_3)$. It thus suffices to prove that γ_3 is the smallest adaptive test sequence with this property.

Proof by induction on the length of γ_3 will be applied. The base case, where $\gamma_3 = \mathit{null}$, is clear since *null* is smaller than or equal to every other adaptive test sequence. Inductive hypothesis: if $\gamma_3 = \mathit{merge}(\gamma_1, \gamma_2)$ has length less than k and γ_4 has the property that $IO(\gamma_1) \cup IO(\gamma_2) \subseteq \mathit{Pre}(\gamma_4)$ then γ_3 is smaller than or equal to γ_4.

Suppose $\gamma_3 = \mathit{merge}(\gamma_1, \gamma_2)$ has length k and γ_4 has the property that $IO(\gamma_1) \cup IO(\gamma_2) \subseteq \mathit{Pre}(IO(\gamma_4))$. It now suffices to prove that γ_3 is smaller than or equal to γ_4.

Note that γ_1 and γ_2 are smaller than or equal to γ_4. Thus the result is clear if either $\gamma_1 = \mathit{null}$ (so $\gamma_3 = \gamma_2$) or $\gamma_2 = \mathit{null}$ (and so $\gamma_3 = \gamma_1$). Suppose $\gamma_1 \neq \mathit{null}$ and $\gamma_2 \neq \mathit{null}$. Then there exists $x_1 \in X$ such that $\gamma_1 = (x_1, f_1)$, $\gamma_2 = (x_1, f_2)$, $\gamma_3 = (x_1, f_3)$, and $\gamma_4 = (x_1, f_4)$.

It is now sufficient to prove that the set $IO(\gamma_3) \setminus \mathit{Pre}(IO(\gamma_4))$ is empty. Proof by contradiction: suppose $\langle x_1/y_1, \ldots, x_m/y_m\rangle \in IO(\gamma_3) \setminus \mathit{Pre}(IO(\gamma_4))$.

Let $\gamma_1' = f_1(y_1)$, $\gamma_2' = f_2(y_1)$, $\gamma_3' = f_3(y_1)$, and $\gamma_4' = f_4(y_1)$. Now consider the relationships between γ_1', γ_2', γ_3', and γ_4'. By the definition of *merge*, $\gamma_3' = \mathit{merge}(\gamma_1', \gamma_2')$. Further, $IO(\gamma_1') \cup IO(\gamma_2') \subseteq \mathit{Pre}(\gamma_4')$ and $\langle x_2/y_2, \ldots, x_m/y_m\rangle \in IO(\gamma_3') \setminus \mathit{Pre}(IO(\gamma_4'))$. Since γ_3' has length at most $k - 1$, this contradicts the inductive hypothesis. The result thus follows. □

Proposition 6. *It is possible to merge adaptive test sequences if and only if they are compatible.*

Proof

By Proposition 5, it is possible to merge γ_1 and γ_2 if they are compatible. It thus suffices to prove that if γ_1 and γ_2 can be merged then they are compatible. Proof by contradiction: suppose that γ_1 and γ_2 can be merged but they are not compatible. Let γ_3 denote the result of merging γ_1 and γ_2.

Since γ_1 and γ_2 are not compatible there exists $v = \langle x_1/y_1, \ldots, x_{m-1}/y_{m-1}\rangle$, $v\,\langle x_m/y_m\rangle \in Pre(IO(\gamma_1))$, and $v\,\langle x'_m/y'_m\rangle \in Pre(IO(\gamma_2))$ with $x_m \neq x'_m$. By definition, $vx_m/y_m \in Pre(IO(\gamma_3))$ and $vx'_m/y'_m \in Pre(IO(\gamma_3))$. This contradicts the definition of an adaptive test sequence, since γ_3 must allow two different input values after the input/output sequence v. The result thus follows. □

The following result is an immediate consequence of the definitions.

Proposition 7. *If $\gamma_1 \in \mathcal{T}$ and $\gamma_2 \in \mathcal{T}$ are compatible then $\gamma_1 \leq merge(\gamma_1, \gamma_2)$ and $\gamma_2 \leq merge(\gamma_1, \gamma_2)$.*

By merging compatible adaptive test sequences it is possible to reduce the number of sequences considered.

Definition 16. *A set Γ of adaptive test sequences is* irreducible *if no two elements of Γ are compatible. Otherwise Γ is* reducible.

It will be assumed that any set of adaptive test sequences considered is irreducible: where there are compatible adaptive test sequences these are merged before the order of application is decided.

Observe that given a reducible set Γ of adaptive test sequences, there may be more than one way in which to merge the elements of Γ in order to produce an irreducible set. Future work will consider the problem of choosing an optimal irreducible set.

4.2 Independent Adaptive Test Sequences

We have seen that the order of the application of adaptive test sequences may be important. However, there may be adaptive test sequences in the test whose relative order is irrelevant. Where this is identified, the problem of determining the optimal ordering may be reduced to that of determining the optimal ordering amongst the elements within a number of sets of adaptive test sequences.

Suppose that $\gamma_1, \gamma_2 \in \mathcal{T}$, $\gamma_1 \not\leq \gamma_2$ and $\gamma_2 \not\leq \gamma_1$. Then it may appear that the relative order of γ_1 and γ_2 is irrelevant. However, since $\leq$ is not transitive, there may be some γ_3 such that $\gamma_1 \leq \gamma_3$ and $\gamma_3 \leq \gamma_2$. Thus, in order to define some notion of independence we form an equivalence relation from $\leq$.

Definition 17. *γ_1 and γ_2 are* dependent *in a set Γ of adaptive test sequences if and only if there exist $\gamma^1, \ldots, \gamma^k \in \Gamma$ with $\gamma_1 = \gamma^1$, $\gamma_2 = \gamma^k$ and for all $1 \leq i < k$, $\gamma^i \leq \gamma^{i+1}$ or $\gamma^{i+1} \leq \gamma^i$. If γ_1 and γ_2 are dependent we write $\gamma_1 \sim \gamma_2$; otherwise we say they are* independent *and write $\gamma_1 \not\sim \gamma_2$.*

In Lemma 4 we will prove that if two adaptive test sequences are compatible then they are dependent. The relation $\sim$ is the symmetric, transitive closure of the reflexive relation $\leq$. Thus $\sim$ is an equivalence relation. Given two equivalence classes Q_1 and Q_2 if $Q_1 \neq Q_2$, $\gamma_1 \in Q_1$, and $\gamma_2 \in Q_2$, γ_1 and γ_2 must be independent. Thus it is possible to split the set of adaptive test sequences used into these equivalence classes and determine an order of application for the elements of each equivalence class. This observation may simplify the problem of choosing the optimal ordering.

We have seen two different approaches to simplifying the problem: dividing the set of adaptive test sequences into a set of equivalence classes and merging adaptive test sequences where possible. In Theorem 1 we will prove that these approaches do not conflict: by merging two adaptive test sequences we cannot combine two equivalence classes.

Definition 18. *Given $\gamma \in \mathcal{T}$ and $v \in IO(\gamma)$, $prune(v, \gamma)$ is the adaptive test sequence formed by taking γ and replacing the node reached by input/output sequence v by null.*

$$\begin{aligned} prune(\epsilon, (x, f)) &= null \\ prune(\langle x_1/y_1\rangle\, v, (x_1, f)) &= (x_1, \{y \rightarrow f(y) | y \in Y \setminus \{y_1\}\} \\ &\quad \cup \{y_1 \rightarrow prune(v, f(y_1))\}) \end{aligned}$$

Thus, $prune(v, \gamma)$ removes all extensions of v from $IO(\gamma)$. The following is clear.

Lemma 1. *Given $v\,\langle x/y\rangle \in IO(\gamma), \gamma \in \mathcal{T}$ and $\gamma' = prune(v, \gamma)$, we have that $IO(\gamma') = (IO(\gamma) \cup \{v\}) \setminus \{v\,\langle x/y'\rangle\, | y' \in Y\}$*

We have seen two different approaches to simplifying the problem: dividing the set of adaptive test sequences into a set of equivalence classes and merging adaptive test sequences where possible. In Theorem 1 we will prove that these approaches do not conflict: by merging two adaptive test sequences we cannot combine two equivalence classes.

Lemma 2. *If $\gamma = merge(\gamma_1, \gamma_2)$, $\gamma, \gamma_1, \gamma_2 \in \mathcal{T}$ then $Pre(IO(\gamma)) = Pre(IO(\gamma_1)) \cup Pre(IO(\gamma_2))$.*

Proof

By definition, $Pre(IO(\gamma_1)) \cup Pre(IO(\gamma_2)) \subseteq Pre(IO(\gamma))$. It thus suffices to prove that $Pre(IO(\gamma)) \subseteq Pre(IO(\gamma_1)) \cup Pre(IO(\gamma_2))$. This holds if $IO(\gamma) \subseteq Pre(IO(\gamma_1)) \cup Pre(IO(\gamma_2))$.

Proof by contradiction will be applied: suppose there exists some input/output sequence $v\,\langle x/y\rangle \in IO(\gamma) \setminus (Pre(IO(\gamma_1)) \cup Pre(IO(\gamma_2)))$. Then we may create a new adaptive test sequence $\gamma' = prune(v, \gamma)$. From Definition 15 we know that $IO(\gamma_1) \cup IO(\gamma_2) \subseteq Pre(IO(\gamma))$. Since $v\,\langle x/y\rangle \notin Pre(IO(\gamma_1)) \cup Pre(IO(\gamma_2))$, for all $y' \in Y$ we have that $v\,\langle x/y'\rangle \notin Pre(IO(\gamma_1)) \cup Pre(IO(\gamma_2))$. Thus, by Lemma 1, we have that $IO(\gamma_1) \cup IO(\gamma_2) \subseteq Pre(IO(\gamma'))$. But γ is not smaller than or equal to γ', contradicting γ being the result of merging γ_1 and γ_2. The result thus follows. □

We thus get the following result.

Lemma 3. *Given $\gamma, \gamma_1, \gamma_2 \in \mathcal{T}$, if $\gamma = merge(\gamma_1, \gamma_2)$ and $v \in IO(\gamma)$ then either $v \in IO(\gamma_1)$ or $v \in IO(\gamma_2)$.*

Lemma 4. *If $\gamma_1 \not\sim \gamma_2$, $\gamma_1, \gamma_2 \in \mathcal{T}$, then we cannot merge γ_1 and γ_2.*

Proof

It is sufficient to prove that if we can merge γ_1 and γ_2 then $\gamma_1 \sim \gamma_2$. Suppose that we can merge γ_1 and γ_2 and let $\gamma = merge(\gamma_1, \gamma_2)$.

Let $v = \langle x_1/y_1, \ldots, x_m/y_m \rangle \in IO(\gamma)$. By Lemma 3, $v \in IO(\gamma_1) \cup IO(\gamma_2)$. Without loss of generality, $v \in IO(\gamma_1)$. If $v \in Pre(IO(\gamma_2))$ then $\gamma_1 \leq \gamma_2$, and so $\gamma_1 \sim \gamma_2$ as required. It will thus be assumed that $v \notin Pre(IO(\gamma_2))$.

Let v' denote the longest prefix of v contained in $Pre(IO(\gamma_2))$. Thus $v' = \langle x_1/y_1, \ldots, x_k/y_k \rangle$ for some $k < m$. Suppose there is some extension, $v' \langle x/y \rangle$ of v' in $Pre(IO(\gamma_2))$. Observe that, since γ_1 and γ_2 may be merged, by Proposition 6, γ_1 and γ_2 are compatible. Thus $x = x_{k+1}$, contradicting the maximality of v'. Thus $v' \in IO(\gamma_2)$ and so $\gamma_2 \sim \gamma_1$ as required. □

Lemma 5. *If $\gamma_1 \not\sim \gamma$, $\gamma_2 \not\sim \gamma$ and $\gamma_3 = merge(\gamma_1, \gamma_2)$, $\gamma_1, \gamma_1, \gamma_3 \in \mathcal{T}$, then $\gamma \not\leq \gamma_3$ and $\gamma_3 \not\leq \gamma$.*

Proof

Assume that $\gamma_1 \not\sim \gamma$, $\gamma_2 \not\sim \gamma$, and $\gamma_3 = merge(\gamma_1, \gamma_2)$. Proof by contradiction will be applied. There are two cases to consider.

Case 1: $\gamma_3 \leq \gamma$.

Thus there exists input/output sequences $v \in IO(\gamma_3)$ and $v' \in IO(\gamma)$ such that $v \in pre(v')$. By Lemma 3, $v \in IO(\gamma_1) \cup IO(\gamma_2)$. Without loss of generality, $v \in IO(\gamma_1)$. Thus, $\gamma_1 \leq \gamma$, providing a contradiction as required.

Case 2: $\gamma \leq \gamma_3$.

Thus there exists input/output sequences $v \in IO(\gamma)$ and $v' \in IO(\gamma_3)$ such that $v \in pre(v')$. By Lemma 3, $v' \in IO(\gamma_1) \cup IO(\gamma_2)$. Without loss of generality, $v' \in IO(\gamma_1)$. Thus, $\gamma \leq \gamma_1$, providing a contradiction as required. □

Theorem 1. *We cannot combine two equivalence classes of $\sim$ by merging two adaptive test sequences.*

Proof

Observe that the merging of adaptive test sequences γ_1 and γ_2 can only lead to equivalence classes being merged if one of the following happens:

1. γ_1 and γ_2 are in different equivalence classes.
2. γ_1 and γ_2 are in the same equivalence class Q of $\sim$ but there is some γ in an equivalence class $Q' \neq Q$ such that $\gamma \leq merge(\gamma_1, \gamma_2)$ or $merge(\gamma_1, \gamma_2) \leq \gamma$.

The result thus follows from Lemmas 4 and 5. □

5 Ordering Based on the Dependence Digraph

This section will consider the problem of finding an ordering based on the dependence digraph where the dependence digraph G is acyclic. It transpires that in this case an exact solution may be found in polynomial time. Future work will consider general algorithms.

The following is a useful property of acyclic digraphs.

Proposition 8. *If digraph $G = (V, E)$ is acyclic and $|V| > 0$ then there exists some $v \in V$ such that $indegree_E(v) = 0$.*

Thus, if the dependence digraph G for Γ is acyclic, it is possible to choose some $\gamma \in \Gamma$ such that the application of another element of Γ cannot lead to γ not being required. Thus, in choosing an order in which to apply the elements of Γ it is acceptable to start with γ.

Proposition 9. *Suppose digraph $G = (V, E)$ is acyclic and $v \in V$. Then the digraph $G \setminus \{v\}$ is acyclic.*

From this it is clear that if the dependence digraph G is acyclic, then having chosen some $\gamma \in \Gamma$ as described above, when γ is removed from Γ the resultant digraph is acyclic.

Based on these results, we get the following algorithm for the case when $\leq$ defines an acyclic dependency digraph. This algorithm essentially chooses some ordering based on a directed acyclic graph (DAG).

Algorithm 1

```
1. Input the dependence digraph $G = (V, E)$ ($|V| = n$), where $v_i \in V$ represents
   adaptive test sequence $\gamma_i$.
2. Set $T = \epsilon$, $G_0 = (V_0, E_0) = (V, E)$, $k = 0$.
3. While ($V_k \neq \emptyset$) do
4. Choose some $v_i \in V_k$ such that $indegree_{E_k}(v_i) = 0$.
5. Set $T = T \langle\gamma_i\rangle$, $k = k + 1$, $G_k = (V_k, E_k) = G_{k-1} \setminus \{v_i\}$.
6. od
7. Output $T$
```

The following result is clear.

Proposition 10. *Algorithm 1 has computational complexity $O(n^2)$ where n denotes the number of adaptive test sequences in Γ.*

Thus, where the dependency digraph is acyclic the problem of finding an optimal ordering for the adaptive test sequences may be solved in polynomial time.

6 Future Work

This paper has formalised properties of adaptive test sequences that may be utilised in order to reduce the expected cost of testing. However, a number of problems remain. Where there are cycles in the dependence digraph, any ordering chosen will allow some possible savings while precluding others. This leads to a more complex optimisation problem which will form an important element of future work.

We have shown that, given a set of adaptive test sequences, it may be possible to merge some of these sequences to produce an irreducible set. However, there may be a number of possible orders in which the merging may occur and a number of irreducible sets that may result from this process. Future work will consider algorithms that determine the order in which adaptive test sequences should be merged.

This paper has focussed on the case where the IUT is known to be deterministic and thus will always respond to an input sequence with the same output sequence. Where the IUT is non-deterministic, it is sometimes possible to make a fairness assumption: it is assumed that there is some k such that the use of an adaptive test sequence k times will lead to all possible responses being observed. Where such an assumption is made it is possible for the responses of the IUT to one adaptive test sequence to be capable of determining the response to another adaptive test sequence. Future work will consider the problem of finding an optimal ordering of a set of adaptive test sequences where a fairness assumption can be made.

7 Conclusion

Adaptive test sequences may be used when testing a state-based system. Where the implementation under test I is known to be deterministic, the response of I to one adaptive test sequence γ_1 may fully determine the response of I to another adaptive test sequence γ_2. Thus, the order in which the adaptive test sequences are applied may affect the expected cost of testing. This paper has considered the problem of finding an ordering that minimises the expected cost of testing.

This paper has introduced two ways of reducing the size of the optimisation problem: by merging adaptive test sequences and by splitting the set of adaptive test sequences into a set of equivalence classes that may be considered separately. We have proved that they do not affect one another: merging adaptive test sequences cannot lead to equivalence classes being combined.

The relation, that states when the use of one adaptive test sequence may lead to another not being used, has been represented as the dependence digraph. This paper has given a low-order polynomial time algorithm that produces the optimal ordering where the dependence digraph is acyclic. Future work will consider how the optimal ordering may be produced where the dependence digraph contains cycles.

References

1. H. AboElFotoh, O. Abou-Rabia, and H. Ural. A test generation algorithm for protocols modeled as non-deterministic FSMs. *The Software Engineering Journal*, 8:184–188, 1993.
2. E. Brinksma. A theory for the derivation of tests. In *Proceedings of Protocol Specification, Testing, and Verification VIII*, pages 63–74, Atlantic City, 1988. North-Holland.

3. J. Grabowski, A. Wiles, C. Willcock, and D. Hogrefe. On the design of the new testing language TTCN-3. In *Testing of Communicating Systems*, pages 161–176, Ottawa, August 29 - September 1 2000. Kluwer Academic Publishers.
4. R. M. Hierons. Generating candidates when testing a deterministic implementation against a non-deterministic finite state machine. *The Computer Journal*, 46:307–318, 2003.
5. Joint Technical Committee ISO/IEC JTC 1. *International Standard ISO/IEC 9646-1. Information Technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General concepts.* ISO/IEC, 1994.
6. ITU-T. *Recommendation Z.500 Framework on formal methods in conformance testing.* International Telecommunications Union, Geneva, Switzerland, 1997.
7. D. Lee and M. Yannakakis. Principles and methods of testing finite-state machines - a survey. *Proceedings of the IEEE*, 84:1089–1123, 1996.
8. J. Tretmans. Conformance testing with labelled transitions systems: Implementation relations and test generation. *Computer Networks and ISDN Systems*, 29:49–79, 1996.
9. P. Tripathy and K. Naik. Generation of adaptive test cases from non-deterministic finite state models. In *Proceedings of the 5th International Workshop on Protocol Test Systems*, pages 309–320, Montreal, September 1992.
10. S. Yoo, M. Kim, and D. Kang. An approach to dynamic protocol testing. In *IFIP TC6 10th International Workshop on Testing of Communicating Systems*, pages 183–199, Cheju Island, Korea, September 1997. Chapman and Hall, London.