

# Ordering adaptive test cases

R. M. Hierons

Brunel University, UK

[rob.hierons@brunel.ac.uk](mailto:rob.hierons@brunel.ac.uk)

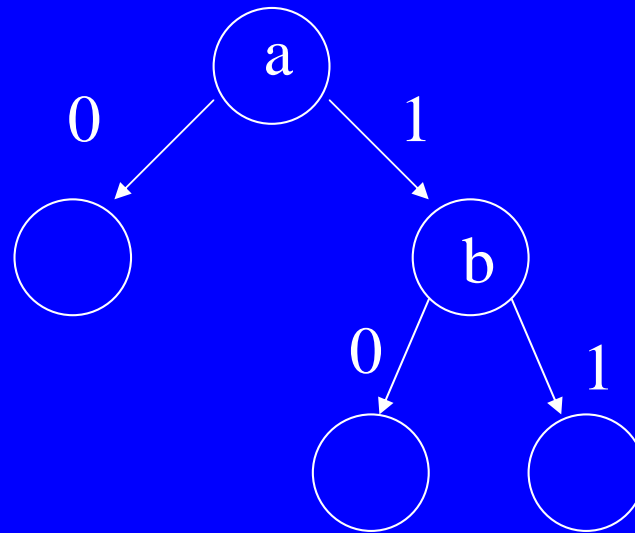
<http://people.brunel.ac.uk/~csstrmh>

# Testing state based systems

- When testing a state based system we apply sequences of inputs and check the resultant output sequences.
- We might use adaptive test cases:
  - In an adaptive test case the next input applied depends on the input/output sequence that has been observed.

# Example

- Can be thought of as a decision tree.

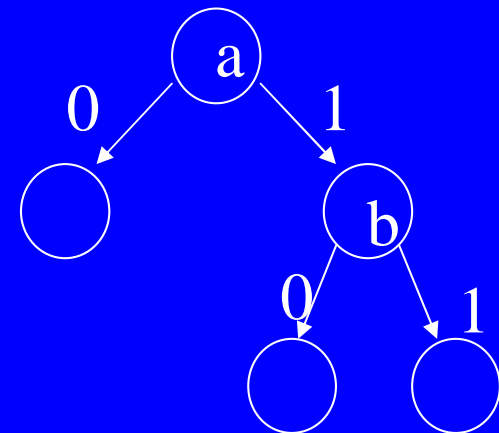


# Why use adaptive test cases?

- These may be required where:
  - We have a non-deterministic specification.
  - We want our test to be resilient to certain types of fault.
- Note that:
  - TTCN has this notion of adaptivity at its core.
  - When testing from a specification written using a process algebra we usually use adaptive tests.

# Formally defining (finite) adaptive test cases

- We will initially consider adaptive test cases that represented by *finite* trees.
- Such an adaptive test case is one of:
  - *null* (apply no input)
  - $(x, f)$  for an input  $x$  and function  $f$  from outputs to adaptive test cases.



# The cost of testing

- Normally we have:
  - The cost of generating our adaptive test cases
  - The cost of executing our adaptive test cases
  - The cost of checking the test output
- We will:
  - Assume that the adaptive test cases are already given
  - Assume that we reset between adaptive test cases
  - Focus on minimising the cost of executing our adaptive test cases
- Note: sometimes this is important, but not always!

# Cases

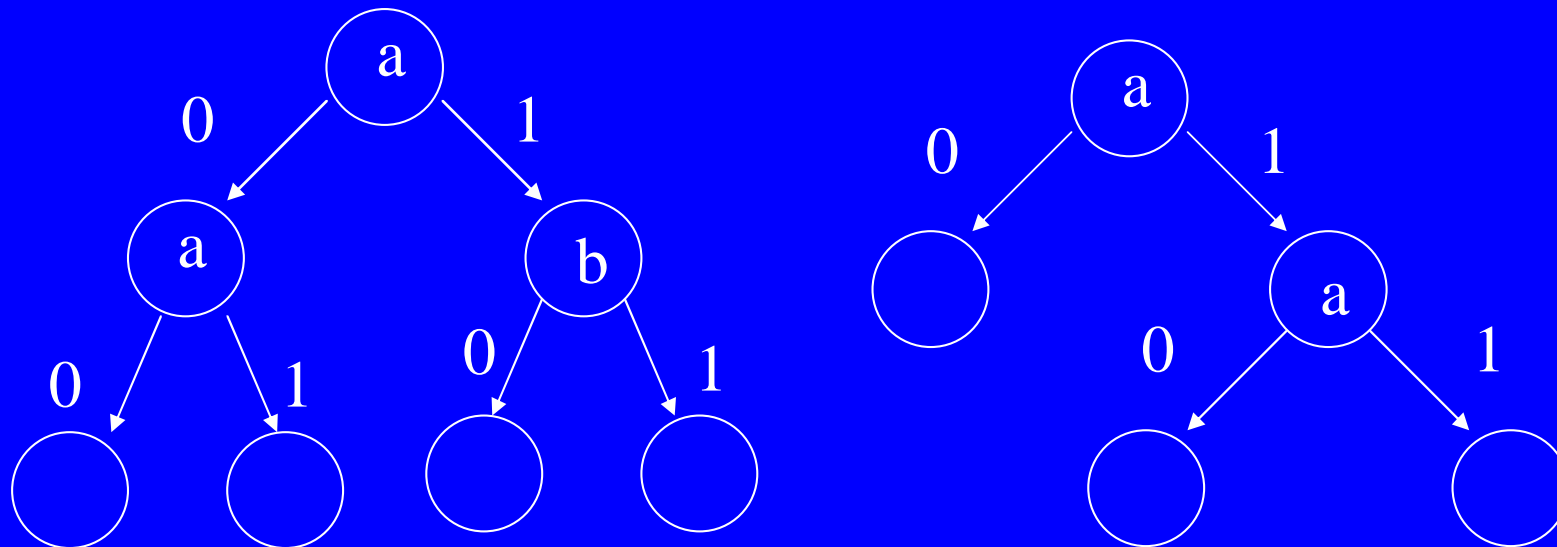
- We will assume that we have a set of adaptive test cases.
- We will consider three cases:
  - Testing a deterministic implementation.
  - Testing a non-deterministic implementation.
  - Applying adaptive test cases on-the-fly for deterministic implementations.
- We wish to minimise the cost of testing *without reducing its inherent effectiveness*.

# Testing deterministic systems

# An initial observation

- Suppose we apply adaptive test case  $\gamma$ , observe trace  $x/y$ , reset and apply  $\gamma$  again.
- Since the system under test (SUT)  $I$  is deterministic we will again observe  $x/y$ .
  - There is no need to apply an adaptive test case more than once.
  - If we have already observed trace  $x/y$  then we do not have to apply  $\gamma$ .

# Test cases can relate



- Here a/0,a/0 for the first adaptive test case tells us that we will get response a/0 to the second (applied after a reset).

# Utilising the savings

- We could simply repeat the following:
  - Apply an adaptive test case and observe the resultant trace  $x/y$ .
  - Remove from the test suite any adaptive test case  $\gamma$  such that the response to  $\gamma$  is predicted by  $x/y$ .
- We may not have to apply all of our adaptive test cases.

# So

- We might have adaptive test cases  $\gamma_1$  and  $\gamma_2$  such that:
  - There is some *possible* response to  $\gamma_1$  that would determine the response of the SUT to  $\gamma_2$ .
- We denote this  $\gamma_2 \leq \gamma_1$ .
- Clearly is not  $\leq$  symmetric.
- Note: it may be the case that some possibly responses of an SUT to  $\gamma_1$  will determine the response of this SUT to  $\gamma_2$ , but others will not.

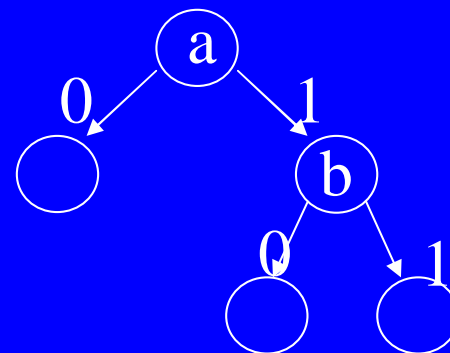
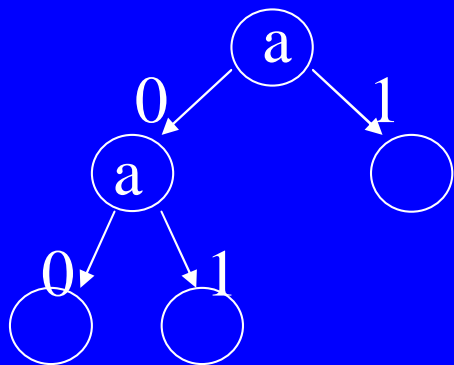
# Consequence

- The expected cost of testing depends upon the *order* in which the adaptive test cases are to be applied.
- Question: how can we find an order that minimises the expected cost of testing?

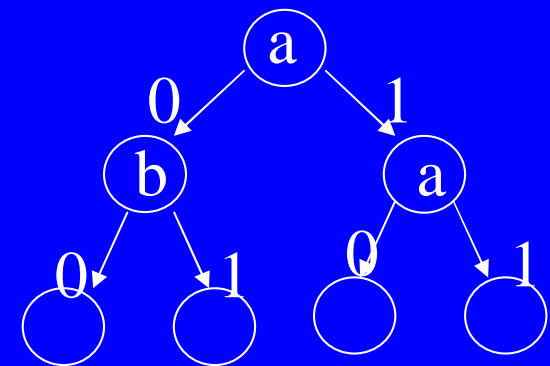
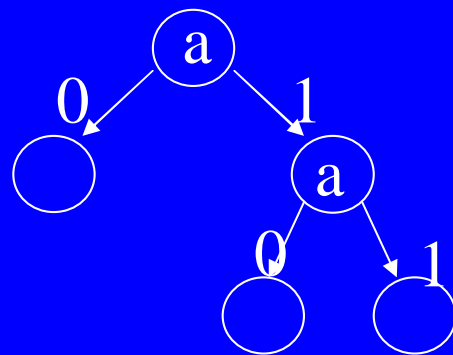
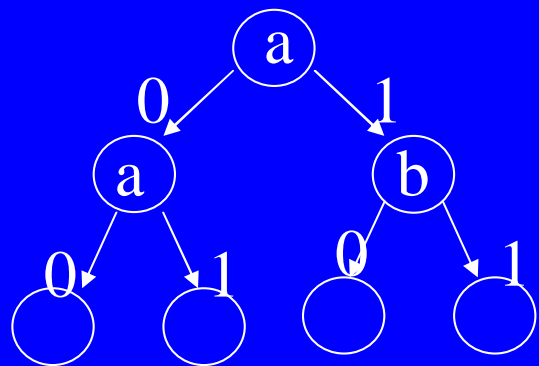
# Deciding $\leq$

- $\gamma_2 \leq \gamma_1$  if and only if  $\text{sav}(\gamma_1, \gamma_2)$  where:  
 $\text{sav}(\gamma, \text{null}) := \text{true}$   
 $\text{sav}(\text{null}, (x, f)) := \text{false}$   
 $\text{sav}((x_1, f_1), (x_2, f_2)) := (x_1 = x_2) \wedge \exists y. \text{sav}(f_1(y), f_2(y))$
- Good news: this requires time that is **linear** in the size of the adaptive test cases.

The relation  $\leq$  is not antisymmetric



# The relation $\leq$ is not transitive

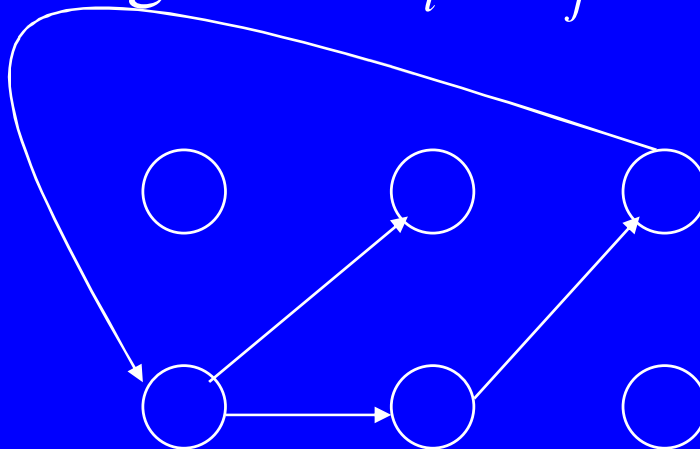


# The optimisation problem

- Given our set of adaptive test cases we want to:
  - Find an ordering that minimises the expected cost of testing.
- We can rephrase this as:
  - Find an ordering that maximises the expected saving through not having to apply some of the adaptive test cases.

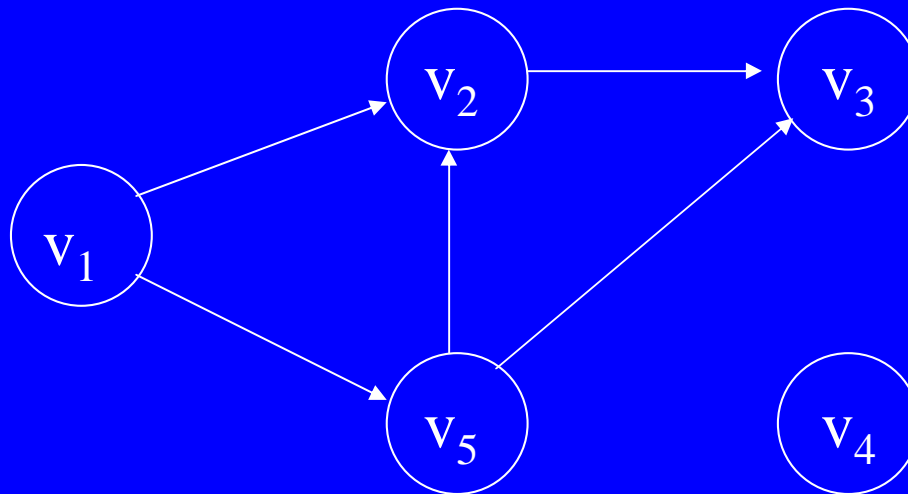
# The dependence digraph

- Given set  $\Gamma = \{\gamma_1, \dots, \gamma_n\}$  of adaptive test cases the dependence digraph  $G=(V,E)$  is:
  - $V = \{v_1, \dots, v_n\}$
  - There is an edge from  $v_i$  to  $v_j$  in  $E$  if and only if  $\gamma_j \leq \gamma_i$ .

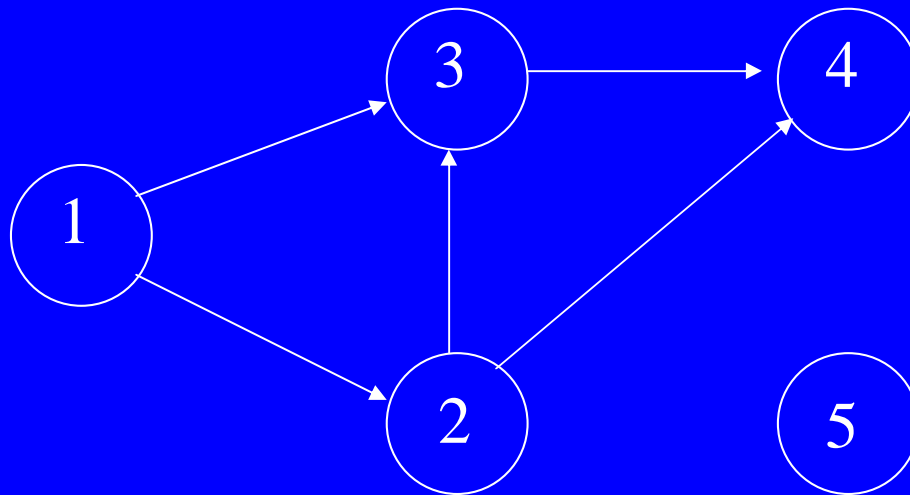


# Example

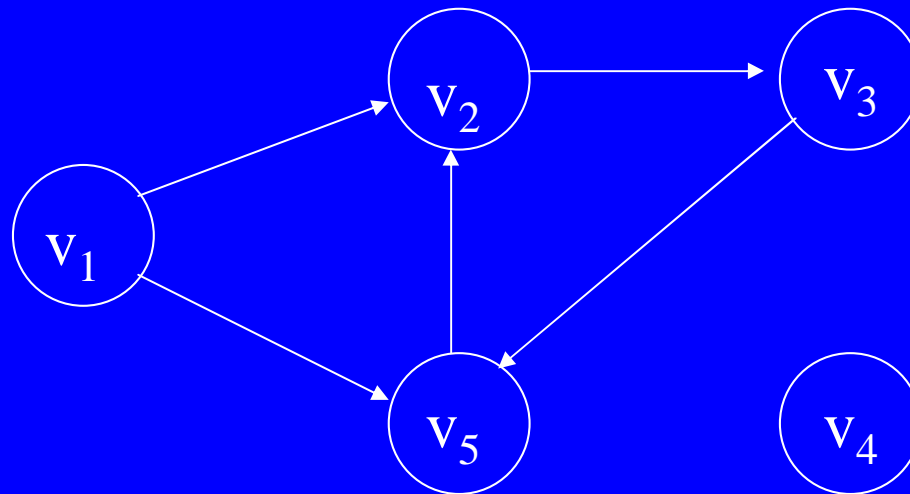
- What is the best ordering given the following dependence digraph?



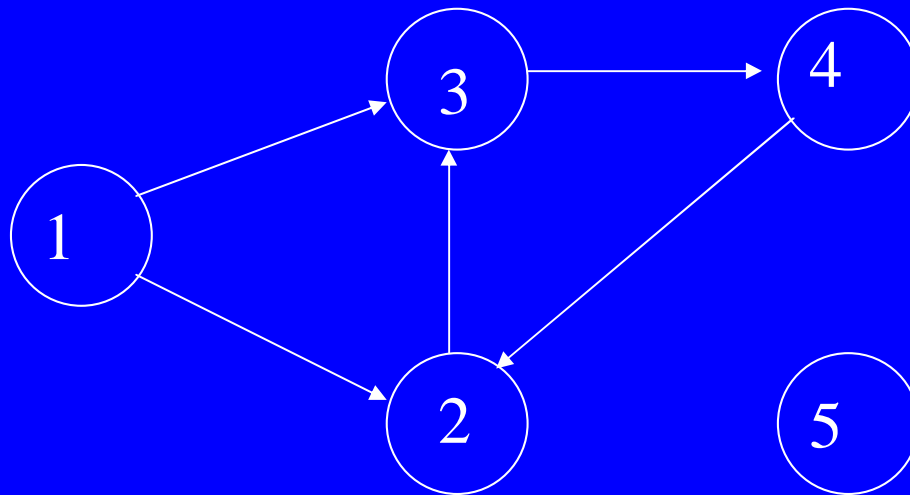
# This order?



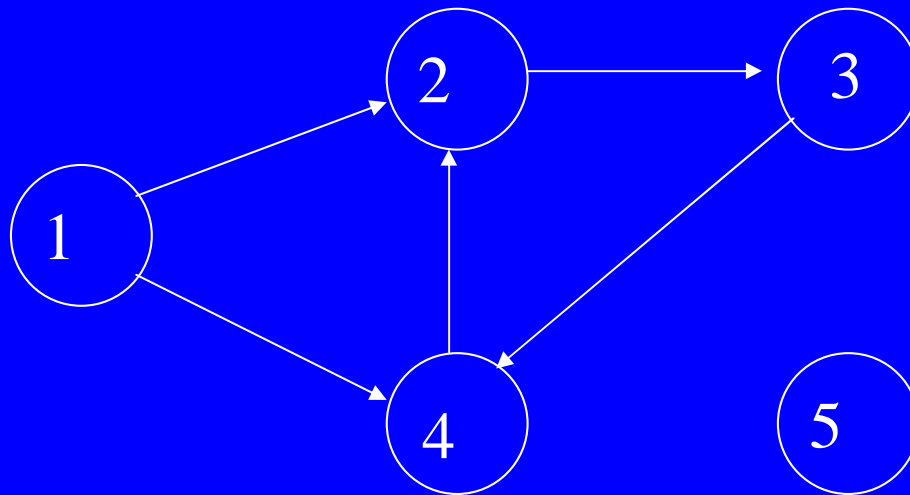
# And this one?



# One order



# An alternative



# An observation

- If we order the vertices of a digraph and remove the edges that go backwards we get an acyclic subgraph.
- Every acyclic subgraph is consistent with a set of orderings of the vertices.
- So: there is a correspondence between orderings of vertices and acyclic subgraphs.

# Solving in terms of the dependence digraph

- If we consider *only*  $G$  then the optimal order is:
  - The order that minimises the number of edges that ‘point backwards’
- Finding this is an instance of the Feedback Arc Set (FAS) problem.

# Encoding the FAS problem

- Suppose we have a digraph  $G'$  and we wish to solve the FAS problem.
- We can show that we can define a set of adaptive test cases such that:
  - An optimal ordering of these defines a solution to the FAS problem.
- For each vertex  $v_i$  of  $G$  we define an adaptive test case  $\gamma_i$ .

# The structure of the adaptive test cases

- For all  $\gamma_j \neq \gamma_k$  we have an output  $b_{jk}$
- Adaptive test case  $\gamma_i$  is of the form:
  - The first input is  $a_0$
  - We have to decide what adaptive test case we get from  $\gamma_i$  after  $a_0/b_{jk}$ .
  - If  $i \neq j$  and  $i \neq k$  then there is input  $a_i$  (only every applied in  $\gamma_i$ ) and all output take us to null.
  - So we know that  $\gamma_i \leq \gamma_k$  can only result from behaviour after  $a_0/b_{jl}$  if  $\{i,k\} = \{j,l\}$ .

## Where $(v_i, v_k)$ is an edge

- We set  $\gamma_i$  and  $\gamma_k$  so that we have  $\gamma_k \leq \gamma_i$ .
- We do this by having the following after  $a_0/b_{ik}$ :
  - $\gamma_k$  leads to input  $a_0$  and then null whatever the next output
  - $\gamma_i$  leads to input  $a_0$  and then input  $a_0$  whatever the next output (and then null).
- We have that  $\gamma_k \leq \gamma_i$  as a result of the behaviour after  $a_0/b_{ik}$  – and as a result of this *only*.

# Where $(v_i, v_k)$ is not an edge

- We set  $\gamma_i$  and  $\gamma_k$  so that we do not have  $\gamma_k \leq \gamma_i$ .
- We do this by having the following after  $a_0/b_{ik}$ :
  - $\gamma_k$  leads to input  $a_k$  and then null whatever the next output
  - $\gamma_i$  leads to input  $a_i$  and then null whatever the next output
- This ensures that we do not have  $\gamma_k \leq \gamma_i$

# Properties of these adaptive test cases

- We have that:
  - $(v_i, v_k)$  is an edge if and only if  $\gamma_k \leq \gamma_i$ .
  - The different possible savings are due to different (and unique) outputs in response to  $a_0$ :
    - they are mutually exclusive
    - they are equiprobable
    - they give the same saving of effort
  - So, an order is optimal if it is one that has fewest edges ‘going backwards’ in  $G$ .

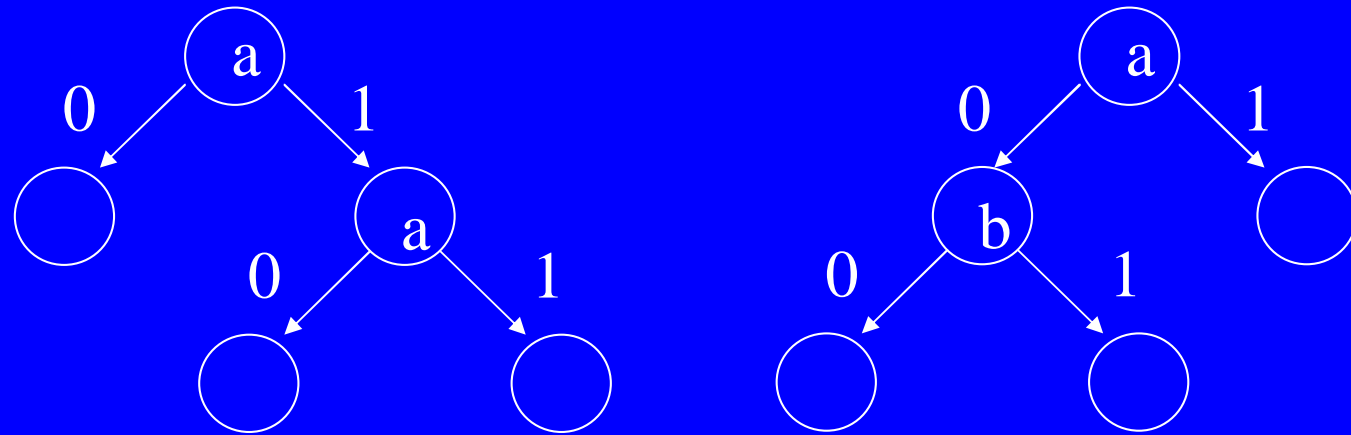
# Complexity

- We have encoded the FAS problem in terms of our problem of finding an optimal order
- We can do so in polynomial time.
- The FAS problem is NP-hard.
- So the problem of finding the optimal ordering is NP-hard.

# Problems

- We will consider:
  - Reducing the size of the optimisation problem:
    - Merging adaptive test cases.
    - Dividing the problem.
  - Producing a ‘good’ ordering based on the dependence digraph.

# Merging adaptive test cases



- These may be merged

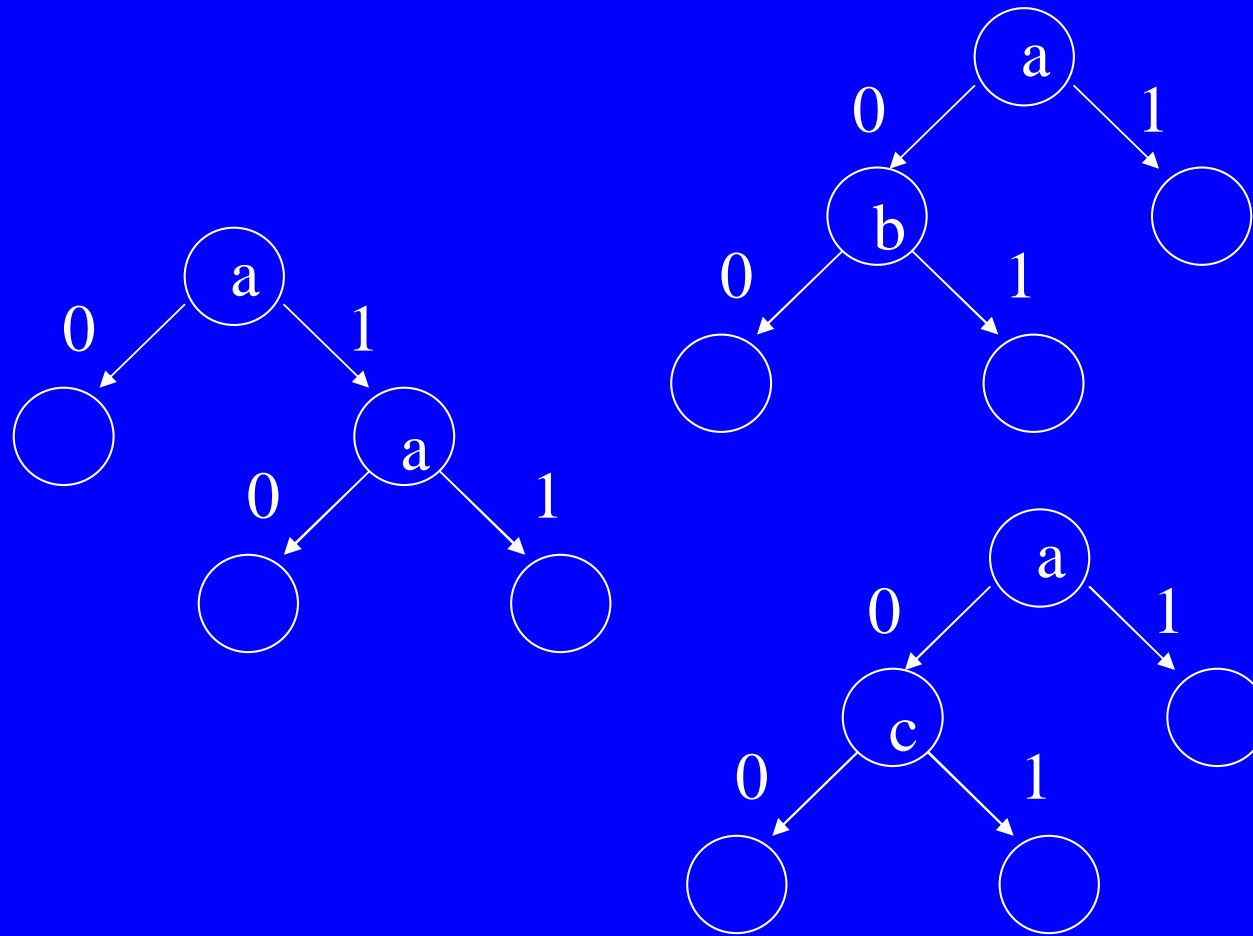
# Compatible adaptive test cases

- Two adaptive test cases  $\gamma_1$  and  $\gamma_2$  are compatible if and only if  $compatible(\gamma_1, \gamma_2) = true$  where:
  - $compatible(\gamma, null) := true$
  - $compatible(null, \gamma) := true$
  - $compatible((x_1, f_1), (x_2, f_2)) := (x_1 = x_2) \wedge \forall y. compatible(f_1(y), f_2(y))$
- Deciding this takes linear time.

# When we can merge adaptive test cases

- Result
  - We can merge two adaptive test cases if and only if they are compatible.

# Merging is not confluent



# Question

- There may be more than one possible result of merging elements of a set of adaptive test cases.
- How can we find a ‘best’ way of merging?
- A problem for future work!

# Independent adaptive test cases

- We might remove the direction of the edges in the dependence digraph to form  $G_1$ .
- Then two adaptive test cases  $\gamma_i$  and  $\gamma_j$  are said to be *independent* if and only if there is no path from  $v_i$  to  $v_j$  in  $G_1$

# Reducing the size of the problem

- We can:
  - Merge compatible adaptive test cases
  - Separately consider the classes of independent adaptive test cases.
- Result:
  - these two approaches do not ‘conflict’.

# A special case: acyclic dependence digraph

- Here we simply repeat the following until all adaptive test cases have been chosen:
  - Choose a vertex  $v_i$  of  $G$  with no edge entering it.
  - Add  $\gamma_i$  to the end of the current order and delete  $v_i$  and the corresponding edges from  $G$ .
- We are finding an ordering based on a DAG.

# Where $G$ contains cycles

- We can:
  - Solve the FAS problem to find some feedback arc set  $A$ .
  - Let  $G' = (V, E \setminus A)$
  - Apply the earlier algorithm to  $G'$

# Another factor: potential saving

- The potential saving varies.
- This depends on the lengths of the traces that we would no longer have to use.
- The potential saving increases as these get longer.

# The probability of 'saving'

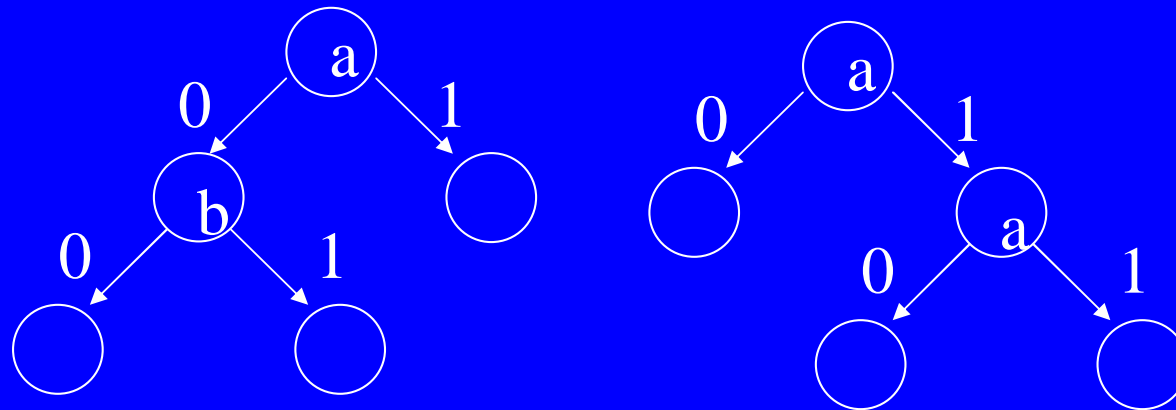
- Suppose that we have adaptive test cases  $\gamma_1$  and  $\gamma_2$  such that  $\gamma_2 \leq \gamma_1$ .
- How likely is it that we will have to use  $\gamma_2$  if we first use  $\gamma_1$ ?
- The probabilities can vary.

# Bringing these together

- Suppose that we have adaptive test cases  $\gamma_1$  and  $\gamma_2$  such that  $\gamma_2 \leq \gamma_1$ .
- We might estimate the *expected saving* from using  $\gamma_2$  after  $\gamma_1$ ?
- Question – how can we use this information in choosing an order?
- A simple approach: give the dependence digraph weighted edges.

# A complication

- There can be a relationship between the edges of the dependence digraph.



- Each is related to the other, but the savings are mutually exclusive.

# Infinite Adaptive Test Cases

# Adaptive test case need not be bounded

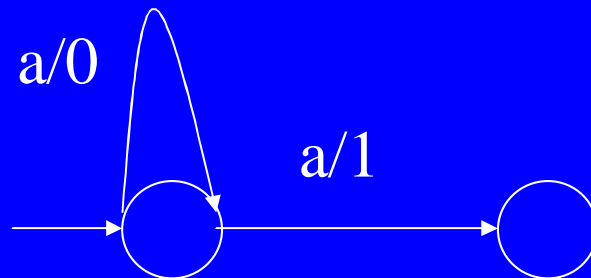
- We have assumed that our adaptive test cases are given by finite trees.
- This does not allow us to include tests such as: continue doing 'x' until 'y' happens and then ...
- In order to represent these as trees we need infinite trees.

# Tests are FSMs

- We can represent a test case as a finite state machine (Petrenko and Yevtushenko, FATES 2005).
- We have an initial state and transitions containing the input to be applied and possible resultant outputs.
- Some states are marked as final states: if we reach a final state then we stop testing.

# Example

- An FSM representing an adaptive test case in which we repeatedly apply a, looping until we receive output 1.



- There is one final state.

# Languages

- Given an FSM  $M$  we let  $L(M)$  denote the corresponding regular language.
- If we have FSMs  $M_1$  and  $M_2$  representing adaptive test cases  $\gamma_1$  and  $\gamma_2$  respectively then the response to  $\gamma_1$  can predict the response to  $\gamma_2$  if and only if:
  - There is a sequence in  $L(M_2)$  that is a prefix of a sequence in  $L(M_1)$ .

# A decision procedure

- Given adaptive test cases  $\gamma_1$  and  $\gamma_2$  then it is sufficient to do the following:
  - Define FSMs  $M_1$  and  $M_2$  representing  $\gamma_1$  and  $\gamma_2$  respectively in which every state of  $M_1$  is a final state.
  - The response to  $\gamma_1$  can predict the response to  $\gamma_2$  if and only if  $L(M_1) \cap L(M_2) \neq \emptyset$ .
- This is decidable in polynomial time.

# Merging

- Given FSMs  $M_1$  and  $M_2$  representing adaptive test cases  $\gamma_1$  and  $\gamma_2$ :
  - We can generate an FSM  $M'$  such that  $L(M')=L(M_1)\cup L(M_2)$
  - $\gamma_1$  and  $\gamma_2$  are *compatible* if and only if  $M'$  has no state with two or more transitions with different inputs leaving it.

# Nondeterministic Implementations

# The issue

- We can no longer predict the behaviour of an adaptive test case based on a trace.
- Even if we apply the same adaptive test case again, we can observe a different trace.

# Repeating tests

- Suppose we apply an adaptive test case  $\gamma$  10 times and observe only two traces.
- Is this different from only seeing two traces having applied it 1000 times?

# Possible approaches

- We can produce results by either:
  - Making a fairness assumption
  - Assuming that all possible observations have at least a given probability
  - Making no assumptions
- The stronger the assumptions made:
  - the greater the potential for reducing the cost of testing
  - the greater the potential for reducing test effectiveness.

# Fairness Assumptions

- We assume that for some predetermined  $k$ , if we apply an adaptive test case  $k$  times then we see all possible responses.
- If in testing we apply each adaptive test case  $k$  times then we can apply analysis similar to that for deterministic implementations.

# Deciding $\leq$ for fairness

- We might observe just one trace in response to adaptive test case  $\gamma_1$ .
- So, if  $\text{sav}(\gamma_1, \gamma_2)$  then it is possible for  $\gamma_1$  to determine the response to  $\gamma_2$ .
- If the responses to  $\gamma_1$  can determine the possible responses to  $\gamma_2$  then some trace in  $L(\gamma_1)$  must contain a trace in  $L(\gamma_2)$  as a prefix and so we have that  $\text{sav}(\gamma_1, \gamma_2)$ .
- So: we can still use our function  $\text{sav}$ .

# When does a set of traces determine the response

- We are interested in the following problem:
  - We have so far observed trace  $z$  in testing and are to apply  $\gamma$  after  $z$ .
  - In previous testing we have observed set  $O_z$  of traces after  $z$ .
  - In previous testing we know that we have observed all possible traces in response to the adaptive test cases used.
- We wish to know: does  $O_z$  tell us what will happen in response to  $\gamma$  after  $z$ ?

# The solution (fairness)

- Base case:
  - We know the response when  $\gamma = \text{null}$ .
- Recursive case is true if:
  - We have observed at least one response to  $x$  after  $z$  (i.e. there is some  $y$  such that  $x/y \in \text{Pre}(O_z)$ ).
  - For every  $y$  such that  $x/y \in \text{Pre}(O_z)$ , the response to  $f(y)$  after  $zx/y$  is determined by the set  $O_{zx/y}$  of traces previously observed after  $zx/y$ .

# Bounds on probabilities

- We could assume that:
  - In every state  $s$  of the SUT and for input  $x$ , each possible response of the SUT to  $x$  when in  $s$  has probability at least  $p$  for some fixed/known  $p$ .
- We can then use results from statistical sampling theory to provide a degree of confidence in having observed all possible traces in response to an adaptive test case.

# How it works

- We apply each adaptive test case a sufficient number of times for us to have the required confidence that no other traces can result from its application.
- We can then apply the function defined for the fairness assumption.

# Making no assumptions

- We have observed all possible responses of the SUT to  $\gamma$  if we have observed every trace than *any* implementation can produce in response to  $\gamma$ .
- So: the response to  $\gamma$  can be determined by a set of adaptive test cases  $\gamma_1, \dots, \gamma_n$  if and only if:
  - $L(\gamma) \subseteq L(\gamma_1) \cup \dots \cup L(\gamma_n)$

# A possible fourth approach

- We might only be interested in traces whose probability of appearing is above some threshold.
- We can use statistical sampling theory.

# Finding a good order on-the-fly for deterministic implementations

Based on work by:

Guy-Vincent Jourdan, Hasan Ural, Nejib  
Zaguia (The University of Ottawa, Canada)

# On-the-fly approaches

- We don't need a preset order.
- We can determine the order as we test (an adaptive approach).
- This allows us to utilize more information.

# A first algorithm

- We can repeat the following:
  - choose a ‘best’ initial adaptive test case based on the dependence digraph;
  - Apply this adaptive test case, observe the resultant output sequence;
  - Eliminate redundant adaptive test cases from our set;
  - Terminate if finished; otherwise produce a new dependence digraph and repeated.

# Merits

- Advantages
  - We can use behaviour observed in testing to determine the order.
  - We expect to do no worse than the preset approach.
- Disadvantages:
  - How do we choose our initial adaptive test?
  - Is this choice optimal?

# An observation

- Suppose that at least one adaptive test case starts with input  $a$ .
- Then we can apply input  $a$  to the SUT and observe the resultant output.
- We can then choose the next input to be consistent with at least one adaptive test case, or terminate.

# Algorithm

- We repeat the following:
  - Choose an input that starts some remaining adaptive test case (terminate if there is none);
  - Apply the input to the SUT and observe the resultant output;
  - Eliminate any adaptive test case whose behaviour has been determined;
  - Choose a next input that is consistent with at least one adaptive test case or, if there is no such input, reset.

# Why this works

- Suppose we have observed input/output sequence  $z$  (and have not reset) and some adaptive test case has input  $a$  after  $z$ .
- Then at some point in testing we will have to apply  $a$  after  $z$ .
- Essentially we are exploring the traces that can result (in the implementation under test) from the adaptive test cases.

# Result

- This approach minimizes the number of inputs used.

## Preset or on-the-fly: relative merits

- The on-the-fly technique allows us to utilize more information and can lead to fewer test inputs being applied.
- The preset technique requires a less sophisticated test environment – most of the processing is done in advance.

# Papers

The following are particularly relevant.

- R.M. Hierons and H. Ural, 2003, Concerning the ordering of adaptive test sequences, FORTE.
- R.M. Hierons and H. Ural, 2007, Reducing the cost of applying adaptive test cases, Computer Networks, **51** 1, pp. 224-238.
- R. M. Hierons, 2006, Applying adaptive test cases to nondeterministic implementations, Information Processing Letters, **98** 2, pp. 56-60.
- A. Petrenko and N. Yevtushenko, 2005, Conformance Tests as Checking Experiments for Partial Nondeterministic FSM, FATES.
- Jourdan, G-V, Ural, H., and Zaguia, N., 2005, Minimizing the number of inputs while applying adaptive tests, Information Processing Letters, **94** 4, pp. 165-169.

# Future work

- The following problems have yet to be investigated:
  - Choosing the order in which to merge adaptive test cases.
  - The relative effectiveness of optimisation algorithms.
  - Optimisation when considering a wider range of sources of information.
  - On-the-fly techniques for non-deterministic implementations.
  - Other features (time, distributed testers ...)

# Conclusions

- The expected cost of applying a set of adaptive test cases depends on the order in which they are applied.
- The problem of finding an optimal preset order is NP-hard.
- Despite this there are:
  - approaches that reduce the scale of the problem;
  - polynomial time algorithms that we might expect to be effective;
  - On-the-fly techniques.

Questions?