Complete Controllable Distributed Testing

R. M. Hierons

Brunel University London, UK

rob.hierons@brunel.ac.uk

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

UCM April 2018

Finite State Machines and MBT

UCM April 2018

Finite State Machines

 The behaviour of M in state s_i is defined by the (regular) set of input/output sequences (traces) from s_i



Implementation relations

- Assuming all models are completely-specified, these are:
 - Equivalence for deterministic FSMs.
 - Language inclusion for nondeterministic FSMs.
- There are efficient algorithms for deciding these properties, so:
 - If we know that the SUT behaves like FSM N and we have specification FSM M then we can determine whether N conforms to M.
- We will focus on: *deterministic FSMs*.

Fault Domains

- A set of models that represent potential behaviours of the system.
- Standard fault domains for testing from an FSM *M* with *n* states:
 - The SUT behaves like an unknown FSM N with at most n states.
 - The SUT behaves like an unknown FSM N with at most m states (some m>n).

Complete test suites

- A test suite *T* is m-complete when testing against *M* if:
 - For every FSM N with no more than m states, if N does not conform to M then there is a test sequence in T that demonstrates this.
 - Implicit: fixed input and output alphabets.
- If the SUT satisfies these conditions then such a test suite *determines correctness*:
 - If the SUT passes the test suite then either it is correct or has more than m states.

Existence of m-complete test suites

- We can produce an m-complete test suite:
 - For each FSM *N* with no more than m states we:
 - Determine whether *N* conforms to specification *M*.
 - If *N* does not conform to *M* then we add a test sequence that demonstrates this.

• These steps are computable (and there are finitely many FSMs to consider).

Smaller test suites

- There are more efficient algorithms.
- Many build test sequences from 'parts' that:
 - Reach a state s.
 - Distinguish two states s and s' (or distinguish every pair of states).

• For deterministic FSMs these 'parts' can be produced in low-order polynomial time.

Summary: using a single tester

- For (deterministic) FSM specification *M*:
 - We can efficiently decide whether an observation is allowed by *M* (the Oracle Problem).
 - We can efficiently produce tests to reach states or distinguish states.
 - We can efficiently decide whether an FSM N conforms to M.
 - We can generate an m-complete test suite for *M*.

Distributed Testing

UCM April 2018

Independent Testers



Consequences

• Each tester observes only the sequence of interactions (*local trace*) at its port



 The tester at port 1 observes x₁y₁x₁y₁ and the tester at port 2 observes y₂ only.

What the testers observe

• Given global trace z, the tester at p observes a local trace $\pi_p(z)$.



Controllability problems

• This test has a controllability problem: introduces nondeterminism into testing.



Observability problems

• The following look the same



A simple output fault

• Input x_1 detects the fault.



Test effectiveness is not monotonic

• However: x_1x_1 does not detect the fault.



Using an external network

• Sometimes we can overcome controllability and observability problems.



Distributed Testing and Deterministic Finite State Machines

UCM April 2018

An allowed behaviour

• Given specification M, a trace σ is allowed if

 $\exists \sigma' \in L(M). \sigma' \sim \sigma$

The language defined by an FSM

• With distributed observations, this is:

$$\mathcal{L}(M) = \{ \sigma' | \exists \sigma \in L(M) . \sigma' \sim \sigma \}$$

• So, a behaviour is correct if $\sigma \in \mathcal{L}(M)$

• N conforms to M if and only if $L(N) \subseteq \mathcal{L}(M)$

UCM April 2018

The language need not be regular

The following 'cheats' – does not have any inputs.



(-, y')

• Clearly, $\mathcal{L}(M)$ is not regular.

The Oracle Problem in Distributed Testing

• We observe projections

 σ_1,\ldots,σ_m

• We want to know whether the following holds:

$$\exists \sigma \in L(M). \forall p \in \mathcal{P}.\pi_p(\sigma) = \sigma_p$$

• Essentially, a membership problem.

$$\sigma_1 \dots \sigma_m \in \mathcal{L}(M)$$

- It is *decidable*, since we could:
 - Form all interleavings of the projections.
 - For each such global trace, determine whether the global trace is allowed by the specification.
- This leads to a combinatorial explosion.

Results

- For single port: Oracle Problem can be solved in low order polynomial time.
- For DFSMs in distributed testing:
 - Can be solved in polynomial time for controllable test sequences; otherwise NP-complete.
- For NFSMs:
 - NP-complete even for controllable testing.
- However, problems become polynomial if we place bounds on the number of ports.

Distinguishing FSMs: result

• Similar to a multi-player game problem.

• It is undecidable whether N conforms to M (and so also whether N is faulty).

• **Consequence**: there is no general algorithm for generating finite m-complete test suites for distributed testing.

Controllable testing

UCM April 2018

This is not controllable



Examples of controllability

• Two controllable scenarios



What makes an input sequence controllable?

- In controllable testing:
 - We can follow the input of x in state s by input x'
 if:
 - x and x' are at the same port; or
 - input x' is at a port p that receives output in response to x.
 - The first case relies on the atomicity of input/output pairs.

Distinguishing states s and s'

- If we restrict to controllable testing we need:
 - *(input sequence) x* causes *no controllability problems* from *s* and *s'*.
 - x leads to different sequences of interactions, for s and s', at some port.
- We say that *x locally s*-*distinguishes s* and *s*'.
- If no input sequence locally distinguishes s and s' they are *locally s-equivalent*.

Distinguishing two states

- Given port p and states s₁ and s₂ of a k-port FSM M with n states:
 - s_1 and s_2 are locally s-distinguishable by an input sequence starting at p if and only if they are locally s-distinguished by some such input sequence of length at most k(n-1).
- This bound is tight.
- The sequences can be found in low-order polynomial time.

Complete testing

- We know that:
 - There is no general algorithm for computing mcomplete test suites.
 - There are benefits to using controllable test sequences.
- We might:
 - Try to achieve 'as much as possible' given that testing is controllable.

c(m)-complete test suites

- Given FSM M we say that test suite T is c(m)complete if:
 - All test sequences in T are controllable.
 - For every FSM N with the same input/output alphabets as M and at most m states:
 - If N and M are locally s-distinguishable then some test sequence in T achieves this.
- i.e. T distinguishes between M and an SUT with at most m states *if this is possible in controllable distributed testing*.

Generating c(m)-complete test suites

UCM April 2018

Restricting attention to controllable test sequences

• We would like to represent the set of controllable test sequences.

• We will use a partial FSM M_{min} to do this.





Results

• A path in M with label σ is controllable if and only if $M_{\rm min}$ has a path with label σ .

• So: *M*_{min} captures 'controllable testing'

Canonical FSMs

• Given FSM *M*, we can find:

- Minimal M_{\min} that is locally s-equivalent to M.

– Maximal (nondeterministic) M_{max} . Created by adding 'chaos state' to complete M_{min} .

• We can find them efficiently.

Relevance of max and min machines

- Machine M_{min} captures all of the traces that FSM N has to implement to conform to M (under s-equivalence).
- Machine M_{max} contains all of the traces that an SUT can have without being distinguishable from M in controllable testing:
 - We can examine M_{max} to determine whether it is acceptable to restrict attention to controllable test cases.

Reaching states

• State s of M is reachable in controllable testing if and only if:

- There is some P such that s^{P} is reachable in M_{min}

• Decidable in polynomial time.

Distinguishing states

- We have that s_1^P and $s_2^{P'}$ are distinguishable in controllable testing if and only if:
 - There is a port $p \in P \cap P'$ and input sequence x starting at p such that x s-distinguishes s_1 and s_2 .
- Decidable in polynomial time.

Refinement and Testing



Generating a c(m)-complete test suite

- It is now straightforward:
 - We generate an m-complete test suite from partial FSM $M_{\rm min}$.
 - or
 - We generate an m-complete test suite from nondeterministic FSM M_{max} .
- There are standard algorithms that can be adapted (e.g. using state counting).

Some papers (FSMs)

- B. Sarikara and G. Von Bochmann, Synthesis and Specification Issues in Protocol Testing, IEEE Transactions on Communications, 32 4, pp. 389-395: 1984.
- R. Dssouli and G. von Bochmann. Error detection with multiple observers, Protocol Specification, Testing and Verification V, pp. 483-494: 1985.
- R. Dssouli and G. von Bochmann, Conformance testing with multiple observers, Protocol Specification, Testing and Verification VI, pp. 217-229: 1986.
- R. M. Hierons and H. Ural. The effect of the distributed test architecture on the power of testing, *The Computer Journal*, **51** 4, pp. 497-510: 2008.
- R. M. Hierons: Canonical Finite State Machines for Distributed Systems, *Theoretical Computer Science*, **411** 2, pp. 566-580: 2010.
- R. M. Hierons: Verifying and Comparing Finite State Machines for Systems that have Distributed Interfaces, *IEEE Transactions on Computers*, **62** 8, pp. 1673-1683, 2013.
- R. M. Hierons: Oracles for Distributed Testing, *IEEE Transactions on Software Engineering*, **38** 3, pp. 629-641, 2012.
- R. M. Hierons: Generating Complete Controllable Test Suites for Distributed Testing, IEEE Transactions on Software Engineering, 41 3, pp. 279-293, 2015.
- R. M. Hierons and Uraz C. Turker: Distinguishing Sequences for Distributed Testing: Adaptive Distinguishing Sequences, *The Computer Journal* (to appear).

Thanks

- Many people have contributed through discussions and collaboration, including:
 - Ana Cavalcanti, Haitao Dan, Christophe Gaston, Marie-Claude Gaudel, Pascale Le Gall, Mercedes Merayo, Manuel Nunez, Uraz Turker, Hasan Ural, Husnu Yenigun.

• The work was partially funded by the EU under the TAROT network.

Conclusions

- If a system has distributed interfaces/ports then we have different implementation relations.
- This can affect testing and also development.
- We have new notions of correctness and corresponding test generation algorithms.
- Restricting attention to controllable test sequences brings practical benefits.

Questions?

UCM April 2018

UCM April 2018

Solving the Oracle Problem

- We observe projections $\sigma_1, \ldots, \sigma_m$
- We can form a finite automata whose language is the set of corresponding global traces (the oracle problem is then FA intersection).
- A state is a vector whose *i*th component is the latest event from σ_i

Example

• Two ports, local traces aa', b.



How this works (1)

- We define a partial order < on events in σ : a < a' if (from the observations) we know that a must have been before a'.
- In this case:
 - Two events are related iff they are at the same port.
- Important property:
 - For a to occur we must have all events before a (under <).
 - Downwardly closed sets correspond to sets of events that can form a prefix of a trace equivalent to σ .
- Note label events to make them unique if required.

How this works (2)

- We can also construct the FA as:
 - States are downwardly closed sets of event.
 - {} is the initial state
 - The complete set of events is the final state.
 - There is a transition from set E to set E' with event e iff {e} = E' \ E.

- The number of states of the FA is the product of the lengths of the σ_i (plus 1)
- So, exponential space is required.

 However, polynomial time if *m* is bounded above.

Testing is weaker

- We cannot locally s-distinguish s_1 and s_4 but x_1x_2 can distinguish them.



Efficiency issue

Many test generation methods use:

- Sets of pairwise distinguishable states.

• Size of test suite depends on how large these are.

Graphs and cliques

- Given an undirected graph G=(V,E) we can generate an FSM M as follows:
 - Each vertex v_i in V is represented by a corresponding state s_i of M.
 - We can distinguish states s_i and s_j if and only if there is an edge between v_i and v_j.
- Consequence:
 - Finding a maximal set of pairwise distinguishable states of M is equivalent to finding a maximal clique of G.

Consequence

• The problem of finding largest sets of pairwise distinguishable states is NP-hard.

- There are potential efficiency issues.
- Note:
 - This result also holds for single-port testing from a nondeterministic FSM or a partial FSM.

Copies of states

- Let us suppose that:
 - t is the transition (s',s,x/y).
 - P is the set of ports involved (p is in P if x is at p and/or y contains output at p).
- We will represent the situation 'after t' by state:
 s^P
- The state s^P denotes the situation in which:
 - The FSM is in state s and can receive input at any port in P in controllable testing.

Transitions leaving a 'new state'

• Let us suppose that:

-t is the transition (s,s',x/y).

- We will include a copy of t from every state of the form s^P such that:
 - Input x is at a port in P.
- We also include an initial state (initial state of M, input at any port).
- The combination defines the FSM M_{\min}