Outline

- FSM based testing – what is it?
- Testing based on deterministic FSMs
  - white box testing approach
  - black box testing approach
  - grey box testing approach
- Testing against nondeterministic FSMs
  - Why do we need to perform nondeterministic FSM based testing?
  - testing trace equivalence relation
  - testing reduction relation
  - testing separability relation
  - testing $\sim$-compatibility relation
Active and Passive Testing

**Active testing**

- Tester
- IUT
- PC – points of control
- PO – points of observation

**Passive testing**

- Tester
- IUT
- PO – points of observation

Verona University
Active and Passive testing (cont-d)

*Active (off-line) testing*
- Test are generated and applied during the special mode at PCs

*Passive (on-line) testing*
- No PCs, no special mode
- an IUT is observed at POs and a conclusion is drawn based on these observations
Model based conformance testing

Fault model

Spec

Tests

Imp

Test execution

Does Imp conform to Spec?
Formal model

- What is a test architecture?
- What is the formal description of the specification and of an implementation?
- What is the formal description of a fault?
- What conformance relation is used for testing?
Finite State Machines (FSM)

- When an input is applied an FSM moves from state to state and produces an output
- FSM maps input sequences into output sequences

Initialized FSM is a 5-tuple \((S, I, O, T, s_1)\)
Behaviors of the specification and of an implementation are described by FSMs.

Applying input sequences to an implementation and observing its output sequences one could be able to conclude whether the implementation conforms to its specification.
TRADITIONAL FAULT MODEL WHEN PERFORMING FSM BASED TESTING

\[
< \xi, \sim, FD >
\]

- Specification FSM
- fault domain: the set of all possible implementation FSMs
- a conformance relation

- Each implementation FSM is complete
- A fault is modeled by a faulty FSM
Variety of FSM based fault models

Fault model: \(<Spec, \sim, FD>\)

- \(Spec\) can be deterministic or nondeterministic, complete or partial
- the conformance relation \(\sim\) is

For deterministic FSMs - is
- equivalence (for complete FSMs)
- quasi-equivalence (for partial FSMs)

For nondeterministic FSMs
- trace equivalence (for complete FSMs)
- trace quasi-equivalence (for partial FSMs)
- reduction (for complete FSMs)
- quasi-reduction (for partial FSMs)
- \(\sim\)-compatibility
- non-separability

- fault domain \(FD\)
- FSMs of FD are explicitly enumerated
- only the upper bound on the number of states of an implementation FSM is known
- implementation FSM is a submachine of a given mutation FSM; usually this machine is based on the specification and faulty implementations that should be detected by a derived test suite
What faults can be detected when performing FSM based testing

- Output fault: the output of a transition is wrong
- Transfer fault: the next state of a transition is wrong
- Mixed faults (output and transfer faults can occur in an implementation under test)
Traditional FSM based testing for complete deterministic FSMs

Fault model: \(<\mathcal{S}, \equiv, \mathcal{R}>\)

- \(\mathcal{S}\) is an initialized complete deterministic FSM
- \(\equiv\) is the trace equivalence relation
- \(\mathcal{R}\) is the set of initialized implementation FSMs
  - has a finite (not too big) list of faulty FSMs (white box testing)
  - only the upper bound on the number of states of an implementation FSM is known (black box testing)
  - implementation FSM is a submachine of some nondeterministic FSM (grey box testing)
Test architecture

Test architecture

Test Generator

Spec

Imp

comparator

Verona University
Complete test suite

**Fault domain** $\mathcal{R}$ is the set of FSMs that describe all possible faults (all possible implementations):

$$\mathcal{I} = \{\text{Imp}_1, \ldots, \text{Imp}_n, \ldots\}$$

A **test suite** $TS$ is a finite set of finite input sequences of $\mathcal{I}$

$TS$ is **complete** w.r.t. $\langle \mathcal{S}, \cong, \mathcal{R} \rangle$ if it detects each FSM $\text{Imp} \in \mathcal{I}$ that is not trace equivalent to $\mathcal{S}$

! If the fault domain is not limited then there is no complete test suite w.r.t. such fault domain

! A test suite is a set of test sequences (not a single input sequence), i.e., a reliable reset is assumed in the specification and in an implementation
Trace equivalence relation

To be equivalent FSMs $Imp$ and $Spec$ have to have the same output response to each input sequence.

The number of input sequences is infinite, while we can apply only finite number of input sequences when testing the conformance.

Trace equivalent FSMs have the same sets of traces:

$$\begin{align*}
&\text{Imp} \\
&\begin{array}{c|c|c}
p_1 & \ldots & p_m \\
\end{array}
\end{align*}$$

$$\begin{align*}
&\text{Spec} \\
&\begin{array}{c|c|c}
s_1 & \ldots & s_n \\
\end{array}
\end{align*}$$
White box testing for complete deterministic FSMs

Where to apply
- Software testing (mutant testing)
  Given software golden sample, extract a flow table (FSM $S$)
  Insert a fault and extract a flow table (FSM $Imp$)
  Determine sequence to distinguish $S$ and $Imp$ (if it exists)

- Testing logic networks (gate level)
  For each single stuck-at-fault that is hard to be detected, a distinguishing sequence is derived by comparison of the specification network and a network with this stuck-at-fault

! More applications for other discrete event systems
Explicit enumeration of mutants

- to explicitly enumerate all possible implementation FSMs
- for each non-conforming $Imp$ to determine a test case that detects $Imp$
- to collect all the sequences into a single test suite (minimize if possible)

! When testing logic networks the approach is applied only for single stuck-at-faults which are hard to be detected

! Moreover, a test suite can be derived simultaneously for a set of single stuck-at-faults
Explicit enumeration of mutants (cont-d)

Advantage: test suites are short enough and detect faults which can often occur

Disadvantage:
The number of mutants (faults) should be small enough

Possible solution: to derive a test suite without explicit mutant enumeration, i.e., a complete test suite is derived based on the specification and some knowledge of an implementations under test
Specification coverage

**Theorem.** A test suite that *traverses* each transition of the specification is *complete* w.r.t. output faults if the implementation has the same number of states as *Spec*

! How to check transition faults depends on the selected fault model
Deriving tests for easily testable logic networks

If a combinational circuit is constructed in a special way in order to simplify the procedure of test derivation then a test suite that detects not only single stuck-at-faults but also multi stuck-at-faults can be derived based on a corresponding system of Boolean functions.
Testing FSM output and transition faults (black box testing)

*Fault model:* $\langle S, \cong, \Omega_n \rangle$

$S$ is a reduced complete deterministic FSM with $n$ states

Fault domain $\Omega_n$ is the set of all complete deterministic FSMs with at most $n$ states

*Problem:* It is impossible to explicitly enumerate all FSMs with two inputs, two outputs and ten states; the number of such FSMs is approximately $20^{20}$

*Solution:* to derive a complete test suite based on the specification FSM only
Black box testing

- The structure of an FSM under test is unknown
- Only the upper bound on the number of states of an FSM under test is known
- We cannot observe states of an FSM under test

What to do?
Test suite derivation using only the specification FSM

Two states $s_j$ and $s_k$ of complete FSM $S$ are \textit{trace equivalent} if the FSM $S$ has the same output response at states $s_j$ and $s_k$ to each input sequence.

\textbf{Proposition.} If a complete initialized FSM $S$ is reduced and connected then initialized connected FSM $P$ is equivalent to $S$ iff $P$ is \textit{isomorphic} to $S$. 
Isomorphic FSMs

Two FSMs $S$ and $P$ are isomorphic iff

There exists one-to-one $h: P \rightarrow S$ between states, $h(p_1) = s_1$ s.t.

$h$ is preserved for transitions

$out_P(p, i) = out_S(h(p), i)$

and

$h(ns_P(p, i)) = ns_S(h(p), i)$

$S$ and implementation FSM $P$ have the same number of states
How to externally check if an implementation FSM is isomorphic to $\mathcal{S}$

We need:

- To insure that a given implementation $\mathcal{P}$ has $n$ states (need to reach each state of $\mathcal{P}$)
- To insure that for each transition of $\mathcal{S}$ there exists a corresponding transition in the FSM $\mathcal{P}$

Checking states and transitions of implementation FSM $\mathcal{P}$

$\mathcal{S}$: $s_1 \ldots s_n$

$\mathcal{P}$: $p_1 \ldots p_n$

$h: \uparrow \downarrow \ldots \uparrow \downarrow$
For each initially connected FSM there always exists a state cover set $SC$ of input sequences that reaches each state of $S$.

Sequences of a state cover $SC$:

- $\alpha(s1) = \varepsilon$
- $\alpha(s2) = x$
- $\alpha(s3) = z$

State cover $SC$ always contains the empty sequence.
Distinguishing sequences

$y$ distinguishes $s_1$ and $s_2$

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$yy$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>1</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>$s_2$</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>$s_4$</td>
<td>1</td>
<td>0</td>
<td>00</td>
</tr>
</tbody>
</table>
Distinguishability (characterization) set

For each complete reduced FSM there always exists a distinguishability (characterization) set $W$ of input sequences that distinguishes every two different states.

Distinguishing sequences:
\[ \gamma(s_1, s_2) = x \]
\[ \gamma(s_2, s_3) = y \]
\[ \gamma(s_1, s_3) = z \]

! Distinguishability set $W$ may not exist for a reduced partial FSM.
W-method (1)

1. For each two states $s_j$ and $s_k$ of the specification FSM $S$ derive a distinguishing sequence $\gamma_{jk}$

Derive a **distinguishability set** $W$ that has a distinguishing sequence for each pair of different states

2. For each state $s_j$ of the FSM $S$ derive an input sequence that takes the FSM $S$ from the initial state to state $s_j$

Derive a **state cover set** $SC$ that has a transfer sequence to each state of $S$

! The empty sequence is mandatory in $SC$
3. Concatenate the state cover set $SC$ with the distinguishability set $W$: $TS_1 = SC \cdot W$

**Proposition** If an implementation FSM $P$ passes $TS_1$ then
- $P$ has exactly $n$ states
- $SC$ is a state cover set of $P$
- there exists one-to-one mapping $h: P \rightarrow S$ s. t. $h(p) = s \Leftrightarrow p \cong_W s$
W-method (3)

4. Concatenate the state cover set $SC$ with the set $iW$ for each input $i$: $TS_2 = SC \cdot I \cdot W$

**Proposition** If an implementation FSM $\mathcal{P}$ that passed $TS_1$ passes also $TS_2$ then one-to-one mapping $h$ satisfies the property:

$$out_\mathcal{P}(p, i) = out_\mathcal{S}(h(p), i)$$

and

$$h(ns_\mathcal{P}(p, i)) = ns_\mathcal{S}(h(p), i)$$

Thus, if FSM $\mathcal{P}$ passes $TS_1 \cup TS_2$ then FSM $\mathcal{P}$ is isomorphic to $\mathcal{S}$, and thus, $\mathcal{P}$ is trace equivalent to $\mathcal{S}$
Test suite returned by W-method

All the sequences that are prefixes of other sequences can be deleted from a complete test suite without loss of its completeness.
W-method (5)

Theorem (Vasilevskiy, 1973) If a state cover \( SC \) is prefix closed and a distinguishability set \( W \) is suffix closed then the set

\[ SC \cdot I \cdot W \]

is a complete test suite for the case when faults do not increase the number of states of the specification

!Reliable reset is assumed in the specification and implementation FSMs
W-method when implementation FSM can have more states than the specification

Let the specification FSM $S$ have $n$ states and an implementation FSM $P$ have at most $m$ states, $m > n$

In order to completely test FSM $P$ when the transition relation of $P$ is unknown we need:

- to reach by test suite each state of $P$
- to check each transition of $P$

1. If $SC$ is a state cover set of $S$ $\Rightarrow$ $SC.I^{m-n}$ is a state cover set for each implementation FSM with at most $m$ states

2. If $W$ is a distinguishability set of $S$ then the set $SC.I^{m-n} \cup SC.I^{m-n+1} \cup W$ is a complete test suite under the assumption that an implementation FSM has at most $m$ states
# Experimental results ($m = n$)

<table>
<thead>
<tr>
<th>State num.</th>
<th>Input num.</th>
<th>Output num.</th>
<th>Trans. num.</th>
<th>Average length</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>6</td>
<td>6</td>
<td>180</td>
<td>2545</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>10</td>
<td>300</td>
<td>3393</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>6</td>
<td>300</td>
<td>5203</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>10</td>
<td>500</td>
<td>6773</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>10</td>
<td>1000</td>
<td>17204</td>
</tr>
</tbody>
</table>
Complexity of W-based methods

Theoretical complexity:
If \( m = n \) then \( TS = SC.I.W \)

\( TS \) length is \( O(kn^3) \) where

\( k \) – number of inputs
\( n \) - number of states

If \( m > n \) then \( TS = SC.I^{m-n+1}.W \)

\( TS \) length is \( O(k^{m-n+1}n^3) \)
Modifications of W-method

1. DS-method
2. UIO-method
3. Wp-method
4. UiOv-method
5. HSI-method
6. H-method
7. SPY method

Depending how a set of separating sequences is defined

! Theoretical complexity is the same

Experimental results can be found in:
Advantages and disadvantages of W-based methods

**Advantage:** Tests are derived based on the specification FSM and have high quality.

**Disadvantages:**
- Test suites are very long.
- Test derivation significantly depends on the integer $m$ that sometimes is difficult to determine.
- $S$ can be partial.
- $S$ can be non-deterministic.
Grey-box testing approach

The structure of an FSM under test is partially known

Where to apply
- User-driven faults
- Incremental testing
Incremental testing or testing user-driven faults

The specification is modified incrementally. An implementation also is incrementally modified.

Only some transitions can be faulty.
Fault domain for grey-box testing

We assume

1. Each possible implementation is complete and deterministic

2. Each possible implementation FSM is a submachine of a nondeterministic FSM that is called a *mutation* FSM
Straightforward solution for incremental testing:

To check only modified transitions using state identifiers

This is not enough

The reason is the correspondence between states of initial $S$ and $Imp$ is not always preserved for modified $S$ and $Imp$
## Experimental results

<table>
<thead>
<tr>
<th>s</th>
<th>i</th>
<th>HSI length</th>
<th>0-5% suspicious transitions</th>
<th>5-10% suspicious transitions</th>
<th>10-15% suspicious transitions</th>
<th>15-20% suspicious transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>10</td>
<td>2992</td>
<td>93</td>
<td>337</td>
<td>490</td>
<td>785</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>5818</td>
<td>148</td>
<td>477</td>
<td>999</td>
<td>1513</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>5333</td>
<td>135</td>
<td>518</td>
<td>957</td>
<td>1450</td>
</tr>
<tr>
<td>35</td>
<td>10</td>
<td>6588</td>
<td>148</td>
<td>539</td>
<td>1013</td>
<td>1537</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>3737</td>
<td>89</td>
<td>345</td>
<td>636</td>
<td>887</td>
</tr>
</tbody>
</table>
Nondeterministic FSM

Non-determinism

\[ s_0 \xrightarrow{i/o_1} s_1 \xrightarrow{\text{>>>}} \]

\[ s_0 \xrightarrow{i/o_2} s_2 \]
Testing nondeterministic FSMs

Nondeterministic behavior occurs in formal descriptions (formal models) according to:

- flexibility
- performance
- limited controllability and/or observability
- abstraction level
- ???
Complexity of test derivation when implementation FSMs are explicitly enumerated

Suppose that all possible implementation FSMs (faults) are explicitly enumerated

- length of a distinguishing sequence w.r.t. trace equivalence is polynomial (in the number of states of FSMs to be compared)
- length of a distinguishing sequence w.r.t. reduction is polynomial (in the number of states of FSMs to be compared)
- complexity of a distinguishing sequence w.r.t. $r$-distinguishability is polynomial (in the number of states of FSMs to be compared) if adaptive test execution is used
- length of a distinguishing sequence w.r.t. separability is exponential (in the number of states of FSMs to be compared)

There exist test derivation methods w.r.t. a number of fault models without explicit enumeration
‘All weather conditions’ assumption

Assumptions:

- All FSMs are complete and initialized (a reliable reset is assumed in the specification and implementation FSMs)
- When testing a nondeterministic FSM w.r.t. trace equivalence and reduction relations we assume that ‘all weather conditions” assumption holds, i.e., an FSM under test (IUT) shows all possible output responses to each applied input sequence (test case)
Testing nondeterministic FSMs

‘all weather conditions assumption’

- there exist test suite derivation methods
  - sometimes this assumption is not realistic
  - each test case is applied several times

+ realistic
  + each test case is applied only once

- existing test derivation methods do not guarantee the fault coverage

holds

does not hold
W-method for nondeterministic FSMs

Fault model: \(<S, \equiv, \Omega_n>\)

\(S\) is an initialized reduced complete nondeterministic observable FSM with \(n\) states

Fault domain \(\Omega_n\) is the set of all initialized complete nondeterministic observable FSMs with at most \(n\) states

Assumptions:
- The specification and each implementation FSM have a reliable reset
- ‘All weather conditions’ assumption holds, i.e., each input sequence (test case) of a complete test suite is applied to an implementation under test several times
W-method for nondeterministic FSMs (1)

1. For each two states $s_j$ and $s_k$ of the specification FSM $\Sigma$ derive a distinguishing sequence $\gamma_{jk}$
   Derive a distinguishability set $W$ that has a distinguishing sequence for each pair of different states

2. For each state $s_j$ of the FSM $\Sigma$ derive an input sequence that can take the FSM $\Sigma$ from the initial state to state $s_j$
   Derive a state cover set $SC$ that has such a sequence for each state of $\Sigma$

! The empty sequence is mandatory in $SC$
! The same input sequence can take $\Sigma$ to different states, i.e., $SC$ can be shorter than that for deterministic FSMs
**W-method (2)**

3. Concatenate the state cover set $SC$ with the distinguishability set $W$: $TS_1 = SC \cdot W$

4. Concatenate the state cover set $SC$ with the set $iW$ for each input $i$: $TS_2 = SC \cdot I \cdot W$

**Proposition.** If an implementation FSM $P$ passes $TS_1 \cup TS_2$ then there exists one-to-one mapping $h: P \rightarrow S$ s.t.

\[
out_p(p, i) = out_s(h(p), i)
\]

and

\[
h(ns_p(p, i)) = ns_s(h(p), i)
\]

Thus, if FSM $P$ passes $TS_1 \cup TS_2$ then FSM $P$ is isomorphic to $S$, and thus, is trace equivalent to $S$
W-method (3)

Test suite returned by W-method

State cover set $SC$

All the sequences that are prefixes of other sequences can be deleted from a complete test suite without loss of its completeness
**W-method when implementation FSM can have more states than the specification**

Let the specification FSM $S$ have $n$ states and an implementation FSM $P$ have at most $m$ states, $m > n$

Complete $TS$ is $SC.I^{m-n+1}.W$

- $SC$ – a state cover of $S$
- $W$ - a distinguishability set of $S$

The complexity is the same as for deterministic FSMs but according to the ‘all weather assumption’ each input sequence is applied several number of times
W-based test derivation methods for nondeterministic FSMs

- W-method is modified to deal with partial FSMs
  As in this case, a distinguishability set may not exist, such methods rely on state identifiers
  Adaptive test execution, since some input can be defined after one output and undefined after another output

- There are attempts to modify W-method to deal with non-observable FSMs

! Applicable only to observable FSMs
Modifications of W-method for nondeterministic FSMs

- HSI-method
- H-method

Depending on how a set of distinguishing sequences is derived
Deriving tests w.r.t. the reduction relation

- For trace equivalence relation two initialized observable reduced FSMs are equivalent iff they are isomorphic, i.e., an implementation FSM has to have the same number of states as the specification FSM.

- Two different states of the reduced specification FSM have to be implemented as different states of a conforming implementation FSM.

- Each state of the reduced specification FSM has to be implemented in a conforming implementation.

- For each complete initialized observable FSM there always exists a reduced complete initialized observable FSM.

- For reduction relation a conforming implementation of initialized observable reduced FSM can have less states than FSM $S$.

  Can be a submachine of $S$.

- Two different states of FSM $S$ can be implemented as a single state of a conforming implementation FSM that is a reduction of $S$.

  If two states of $S$ are $r$-distinguishable only then they have to be implemented as different states of each conforming implementation FSM that is a reduction of $S$.

- Not each state of the reduced specification FSM has to be implemented in a conforming implementation that is a reduction of $S$.

  Can be a submachine of $S$.

- Given a complete initialized observable FSM, there not necessarily exists a trace equivalent complete initialized observable FSM where each two states are $r$-distinguishable.
Deriving tests w.r.t. the reduction relation (example)

FSM $S$

FSM $P$ is a reduction of $S$ but state 2 of $S$ does not have a matching state in $P$
Reduction properties

An input sequence $\alpha$ is a \textit{deterministic transfer} ($d$-transfer) \textit{sequence} to state $s$ if for each possible output sequence $\beta$ the trace $\alpha/\beta$ takes the FSM $S$ from the initial state to state $s$.

State $s$ is a $d$-reachable state.

**Theorem 1.** If state $s$ is a $d$-reachable state of the specification FSM $S$ then state $s$ is implemented in any reduction $P$ of $S$, i.e., the intersection $S \cap P$ has a state $sp$ for some state $p \in P$.

**Theorem 2.** If states $s_1$ and $s_2$ are $r$-distinguishable states of the specification FSM $S$ then they can be distinguished by an $r$-distinguishing set $R(s_1, s_2)$. Moreover, for any reduction $P$ of $S$ it holds that if the intersection $S \cap P$ has a state $s_1p (s_2p)$ for some state $p \in P$ then the intersection $S \cap P$ has no state $s_2p (s_1p)$.
State counting method (SC-method)

Let $R$ be a subset of pair-wise $r$-distinguishable states of FSM $S$ and states of $R_d \subseteq R$ are $d$-reachable.

Let there be a path of a successor tree s.t. states of $R$ occur $(m - |R_d| + 1)$ times.

Implementation FSM has at most $m$ states and $|R_d|$ different states of the implementation FSM already correspond to $d$-reachable states of the FSM $S$.

Thus, two cases are possible:

1) One and the same state $p$ of the implementation FSM $P$ corresponds to two different states $s_1$ and $s_2$ of $R$ (this can be detected by a corresponding $r$-distinguishing set $R(s_1, s_2)$).

2) There are two states $s_1, p$ in the path (i.e. the sequence that labels the path is not a shortest $r$-distinguishing sequence).

Termination rules for the successor tree of the intersection $S \cap P$:

- $d$-reachable states
- $d$-SC
- States $s_1$ and $s_2$ are $r$-distinguishable
Example of deriving a complete test suite w.r.t. reduction relation

FSM $\mathcal{S}$

- States: 1, 2, 3
- Transitions:
  - 1 to 2: a/0
  - 1 to 3: a/1, b/0
  - 1 to 3: b/0,1
  - 2 to 3: a/0,2
  - 3 to 3: a,b/0,1

Implementation FSM $\mathcal{P}$

- States: x, y
- Transitions:
  - x to y: b/0
  - x to y: a/1

Derive a complete $\mathcal{T}S$ w.r.t. the reduction relation when an implementation FSM has at most 2 states

Verona University
Example of deriving a complete test suite w.r.t. reduction relation (cont-d)

FSM $S$

Each two states are $r$-compatible

States 1 and 3 are $d$-reachable
As $m = 2$, states 1 and 3 should occur twice through a path

State 2 should occur three times
Test derivation w.r.t. the reduction relation (references)


If ‘all weather condition’ assumption does not hold

A test suite with the guaranteed fault coverage can be derived w.r.t.

- non-separability relation
- \( r \)-compatibility relation
Separability relation

FSMs $P$ and $S$ are *separable* if

$$\forall \alpha \in I^* (\text{out}_P(t_1, \alpha) \cap \text{out}_S(s_1, \alpha) = \emptyset)$$

When we apply $\alpha$ we always know which FSM $P$ or $S$ is under test
GENERALIZED FAULT MODEL

Solution

to generalize the fault model to

\(< S, (\leq, \neq), FD >\)

Test suite \(TS\) is complete w.r.t. to

\(< S, (\leq, \neq), FD >\)

if each FSM \(p \in FD\)
such that \(p \neq s\) can be detected with \(TS\)

Differences

! Each test case is applied only once
! Some implementations which are non-separable from \(s\)
also are detected
EXPLICIT ENUMERATION

- $\text{Sub}_{n\alpha}(\mathcal{MM})$ – the set of all complete submachines of FSM $\mathcal{MM}$
- $\mathcal{R}_m$ - the set of all initialized complete FSMs with at most $m$ states

Fault models

$<S, (\leq, \neq), \text{Sub}_{n\alpha}(\mathcal{MM})>$ and $<S, (\leq, \neq), \mathcal{R}_m>$

have finite fault domains

A test suite can be derived by explicit enumeration
DERIVING A COMPLETE TEST SUITE W.R.T. $<S, (\leq, \neq), Sub_{nd}(MM)>$ and $<S, (\leq, \neq), \mathcal{R}_{m}>$

A complete test suite w.r.t. $<S, (\leq, \neq), Sub_{nd}(MM)>$ is derived based on a truncated successor tree of the intersection $S \cap MM$

A complete test suite w.r.t. $<S, (\leq, \neq), \mathcal{R}_{m}>$ is derived based on a truncated successor tree of the specification $S$
Testing w.r.t. the separability relation

More results can be found in:


Experimental results when deriving a complete test suite w.r.t. separability relation

- FSM $S$ is a complete nondeterministic FSM with 5 states
- For 20% of pairs $(s, i)$ there is more than one outgoing transition from state $s$ under input $i$
- Each FSM $MM$ is derived by adding up to 25% transitions to $S$

| $|I|$ | $|O|$ | Average test suite length |
|-----|-----|-------------------------|
| 2   | 2   | 2432                    |
| 3   | 3   | 15407                   |
| 3   | 4   | 6826                    |
Deriving adaptive test suites w.r.t. $r$-compatibility relation

Most such tests are only partially adaptive

For deriving adaptive tests we have got only preliminary results
- Testing deterministic implementations against a nondeterministic observable specification FSM
- Explicit enumeration when testing w.r.t. $r$-compatibility relation
Conclusions (1)

FSM based testing is widely used for testing discrete event systems

**Advantage:** FSM is a unique model for which there exist test generation methods for black box testing *without explicit mutant enumeration*

**Disadvantage:** Tests are too long

**Solutions:**
- usually are shortened losing the completeness
- test purposes are used instead of the fault coverage
- to use passive testing
Conclusions (cont-d)

Testing based on nondeterministic FSMs is a challenging problem
Not too much is known how to derive complete tests for nondeterministic FSMs
Adaptive testing needs more research

! There are other formal models and methods how to derive high-quality tests based on these models
! There are other test architectures which were not considered in this lecture
References

**FSM based testing**

References (cont-d)

**Testing non-deterministic FSMs**


Thanks for your attention!