COMMUNICATION PROTOCOLS

CHAPTER 6

Protocol Conformance Testing

Anandi Giridharan
Department of Electrical Communication Engineering
Indian Institute of Science
Bangalore – 560012, India
CONFORMANCE TESTING

Conformance testing is the process of testing the extent to which implementations adhere to the requirements stated in relevant standard or specification.

Distinction is made between functional testing and structural testing. Structural testing, also referred to as white-box testing, is based on the internal structure of a computer program.

Conformance Testing is functional black-box testing

Functional refers to the correct functional behavior of an Implementation Under Test (IUT). Black-box means that the internal structure of the IUT remains hidden.
Conformance Testing Objectives

Interoperability of products from different suppliers
Test the product only once
Acceptance of test results in different geographical regions
Meet regulatory or market driven requirements
Why Do We Need A Common Testing Methodology

- All actors must understand each other in all geographical regions and global markets
  - Equipment suppliers
  - Equipment buyers
  - Test laboratories
  - Accreditation organizations
  - Certification organizations
- Test results must have the same meaning in all global regions
- Test results must be accepted in all global regions
- Time to market - equipment must be tested only once without the need to retest for different markets
Why Conformance to Standards is important?

- Equipment from different vendors conforming to the same standards have a higher likelihood of interoperability.
- Different vendors can independently implement standards with higher assurance of product interoperability.
- Equipment buyers can buy products that will interoperate with previously purchased equipment from different suppliers.
Nature of Conformance Testing

· Testing to determine if the product does what the Recommendation says it is supposed to do.

· Each product is tested only once, against the standard (represented by the test suite).

![Diagram showing Product and Test Suite connected](image-url)
Overview of Conformance Testing

1. Static Review
2. Dynamic Tests
3. Test Report
4. Certification

Implementation with a formal declaration of which parts of the standard were implemented

STIMULUS

RESPONSE

Implementation Under Test (IUT)

Test Equipment and Test Suite

Certificate of Conformance

Test Report
Conformance Testing Framework

IUT – Implementation Under Test
UT – Upper Tester function
LT – Lower Tester function
PCO – Point of Control & Observation
TCP – Test Coordination Procedures
ASP – Abstract Service Primitive
UT plays the role of a user that makes use of the service provided by the IUT.

LT plays the role of the peer entity of the IUT, i.e. the LT and the IUT communicate in order to provide the service to the IUT.
UT plays the role of a user that makes use of the service provided by the IUT.

LT plays the role of the peer entity of the IUT, i.e. the LT and the IUT communicate in order to provide the service to the IUT.
PCO is a standardized interface.

Typically the lower interface of an IUT is accessible only from the remote.

Communication is always meant to be asynchronous => PCO is modeled as two FIFO queues.
TCP can be used to coordinate the actions of LT and UT.

LT and UT can be on separate testers.

IUT and LT communicates by means of ASP.

PDUs are encoded in ASPs of the underlying service.
Testing is the process of trying to find errors in a system implementation by means of experimentation. The experimentation is usually carried out in a special environment, where normal and exceptional use is simulated.

To conduct testing, experiments, or tests must be systematically devised. These tests are applied to an implementation, and the test outcomes are compared with the expected or calculated outcomes.

Based on the results of the comparison a verdict can be formulated about the correctness of the implementation, which, if negative, can be used for improving the implementation.
Conformance Testing

- Process of testing to determine whether a product or system meets some specified standard that has been developed for efficiency and interoperability.

The important factors which characterize conformance testing are as follows:

- The System Under Test (SUT) defines the boundaries for testing.
- The tests are executed by a dedicated test system that has full control and observes the SUT's behaviour.
- The tests are performed on open standardised interfaces.
- Because the test system controls the sequence and contents of the protocol messages sent to the IUT, it can explore a wide range of both expected and unexpected (invalid) behaviours.
Components of Conformance Testing specifications

- Test Purposes,
- Abstract Test Method/Architecture,
- Abstract Test Cases
- Abstract Test Suite
- Test Suite Structure
- Implementation Conformance Statements
A test case starts and ends in stable testing states.

Preamble to take the IUT from stable testing state to test state.

Test body is performed in order to check the test purpose.

Verification to check test body

Postamble to drive the IUT into a stable testing state again
Conformance Test Architectures

- Local test method,
- Distributed method,
- Remote method,
- Coordinated method, and
- distributed combined with local methods.
There are two PCOs. UT and LT both reside on the Test System. The upper boundary of the IUT is standardized hardware interface that plugs into the Test System.

**Figure 6.4: Local conformance test architectures**
Local conformance test architecture - Example

System under test (SUT)  
PCO → ASPs  
PCO  
IUT

Test system (TS)  
Upper Tester  
TCPs  
Lower Tester  
ASPs  
PCO

1) CR  
2) CONind  
3) CONresp  
4) CC

Underlying service

Example of local test method
There are two PCOs. The UT is located in the SUT. The LT is located in the Test System. Access to the upper boundary of the IUT is required to carry out testing either by human action or a programming interface.
Distributed conformance testing architecture - Example

System under test (SUT)

Upper Tester

2) CONind

3) CONresp

PCO

IUT

Test system (TS)

Lower Tester

ASPs

PCO

1) CR

4) CC

Underlying service

Example of distributed test method architecture
Coordinated Method

There is only one PCO and no UT. UT is integrated with TCP. The desired effects at the upper boundary of the IUT are realized by a special TCP called the standardized Test Management protocol. The method facilitates the highest degree of automation and security.
Coordinated Method - Example

- System under test (SUT)
  - 2) CONind
  - 3) CONresp
  - Upper Tester
  - IUT

- Test system (TS)
  - Lower Tester
  - ASPs
  - PCO

- Test management protocol

Example of coordinated test method

1) CR
2)...
3)...
4) CC

Underlying service
Remote Method

There is only one PCO and no UT or TCP.

The Tester has no access to the upper boundary of the IUT.

The desired effects at the upper boundary are informally described in the test suite and are carried out at the SUT by the test operator.

Figure 6.10: Remote conformance testing architecture
Remote Method - Example

Remote method example
What is a Test Suite?
- A test suite is a collection of test cases, one for each test purpose, specified in accordance to the test method used
- A test case verifies conformance/interoperability for a particular Requirement or Option according to the test purpose
To confirm if an implementation conforms to its standard, an external tester applies a sequence of inputs to the IUT and verifies its behavior.

**Issue1:** Preparation of conformance tests in coverage of IUT's all aspects.

**Issue2:** Time required to run the test should not be unacceptably long.

Two main limitations:

- **Controllability:** The IUT cannot be directly put into a desired state, usually requiring several additional state transitions.
- **Observability:** Prevents the external tester from directly observing the state of the IUT, which is critical for a test to detect errors.

Formal conformance testing techniques based on FSM generate a set of input sequences that will force the FSM implementation to undergo all specified transitions.

**Black box approach:** Only the outputs generated by the IUT (upon receipt of inputs) are observable to the external tester.
Fault Model for FSM

Output fault: the machine provides an output different from the one specified by the output function.

Transfer fault: the machine enters a different state than that specified by the transfer function.

Transfer faults with additional states: number of states of the system is increased by the presence of faults, additional states is used to model certain types of errors.

Additional or missing transitions: one basic assumption is that the FSM is deterministic and completely defined (fully specified). So the faults occur when it turns out to be non-deterministic and/or incompletely (partially) specified.
Test Sequence Generation Methods

- A conformance test sequence for a protocol is a sequence of input/output pairs derived from protocol specification.

![Transition diagram]

<table>
<thead>
<tr>
<th>Input State</th>
<th>A</th>
<th>B</th>
<th>Next-state A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>λ</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

e 6.12: A transition diagram and table for a machine M
For a given FSM S, a transition tour is a sequence which takes the FSM S from the initial state s0, traverses every transition at least once,

- Straightforward and simple scheme
- New state of the FSM is not checked

Fault detection power
- Detects all output errors
- There is no guarantee that all transfer errors can be detected

**Example**
Example 2

A minimum-cost transition tour of the FSM is (including reset edges), starting from state 1:

r, a,
r, c, a, b, b,
r, c,
r, c, a, b,
r, c, a, c, b, a, c, a,
r, c, b, a, a
T method

A transition tour sequence is:
0, 0, 3, 3, 4, 0, 3, 4, 2, 1, 4, 2, 1, 2

(not return to 0, but traverse all the transitions)
T-Method

T method is relatively simple compared to other methods. TS can be generated simply applying random inputs to fault free machine until machine has traversed every transition at least once. Redundant inputs can be removed using reduction method.

S0-S0-S3-S3-S4-S0
S3-S4-S2-S1-S4-S2-S1-S2

TS--BABABAAAAAAAAB

A transition diagram and table for a machine M
UIO Method

For testing it is sufficient to know an error has been detected
UIO sequence of a state of a FSM
  An I/O behavior that is not exhibited by any other state of the FSM
Advantages against DS & CS
  Cost is never more than DS and in practice is usually much less (shorter)
    Nearly all FSMs have UIO sequences for each state
    DS - same for all states; UIO sequence - normally different for each state
To check state s by using UIO sequence of s
  Apply input part of UIO, compare output sequence with the expected one
    If the same, then the FSM is in the state s; otherwise, not in the state s
    If not in state s, no information about the identity of the actual state s'
UIO Method

Example

The UIO sequences are:
- state 1: a/x, b/x
- state 2: a/x, a/x
- state 3: a/y

The $\beta$-sequences generated by U-Method are:
$$
\begin{align*}
  & r, a, b \\
  & r, a, a \\
  & r, b, a, a \\
  & r, b, a, b \\
  & r, b, b, a \\
  & r, a, a, a \\
  & r, a, b, a, b
\end{align*}
$$
Example

The UIO sequences are:
state 0: B/null  state 1: A/1,A/1
state 2: B/0    state 3: B/1,B/1
state 4: A/1,A/0

The β-sequences are:
r,A,B,B  r,B,B
r,A,A,A  r,A,B,B,B
r,A,A,A,B  r,A,A,B,B

An optimized test sequence is:
AAAAABAABrAAABBBBrAABBBrAB
BBBrBB
Unique Input/Output

Involves unique Input output sequence for each state. Unique I/O Is not exhibited by any other state.
S1-A/1, A/1
Input A, A machine transits to State S4 and later to S2 producing Output 1, 1 respectively. It repeats no where.

<table>
<thead>
<tr>
<th>State</th>
<th>UIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>B/λ</td>
</tr>
<tr>
<td>1</td>
<td>A/1 A/1</td>
</tr>
<tr>
<td>2</td>
<td>B/0</td>
</tr>
<tr>
<td>3</td>
<td>B/1 B/1</td>
</tr>
<tr>
<td>4</td>
<td>A/1 A/0</td>
</tr>
</tbody>
</table>

S0-S0------r B B-----------------------------
S0-S3------r A BB------>
### U - Method

<table>
<thead>
<tr>
<th>State</th>
<th>UIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>B/λ</td>
</tr>
<tr>
<td>1</td>
<td>A/1 A/1</td>
</tr>
<tr>
<td>2</td>
<td>B/0</td>
</tr>
<tr>
<td>3</td>
<td>B/1 B/1</td>
</tr>
<tr>
<td>4</td>
<td>A/1 A/0</td>
</tr>
</tbody>
</table>

**UIO sequences for M**

**Test subsequences for U method**

<table>
<thead>
<tr>
<th>Transitions</th>
<th>Test subsequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>state S0 to S0</td>
<td>r B B</td>
</tr>
<tr>
<td>state S0 to S3</td>
<td>r A BB</td>
</tr>
<tr>
<td>state S2 to S1</td>
<td>r AAA AA AA</td>
</tr>
<tr>
<td>state S2 to S2</td>
<td>r AAA AB B</td>
</tr>
<tr>
<td>state S2 to S1</td>
<td>r AAA A AA</td>
</tr>
<tr>
<td>state S4 to S3</td>
<td>r AA AB BB</td>
</tr>
<tr>
<td>state S3 to S4</td>
<td>r A A AA</td>
</tr>
<tr>
<td>state S3 to S2</td>
<td>r A AA B</td>
</tr>
<tr>
<td>state S3 to S3</td>
<td>r A B BB</td>
</tr>
<tr>
<td>state S4 to S2</td>
<td>r AA A B</td>
</tr>
<tr>
<td>state S3 to S0</td>
<td>r A AB B</td>
</tr>
</tbody>
</table>
DS method

A sequence of inputs is a distinguishing sequence (DS) for an FSM $S$, if the output sequence produced by the FSM $S$ in response to the input sequence is distinct for each initial state.

Fault detection power
- Detects all output errors
- Detects all transfer errors

Two severe drawbacks
- In practice, very few FSMs actually possess a DS
- Even if an FSM does have a DS, the upper bound on the length of the DS will be too large to be useful in general.
DS method

Example 1

The specification \( S \).
A distinguishing sequence is: \( b,b \)
If we apply it from:
- state 1, we obtain \( y,y \)
- state 2, we obtain \( y,x \)
- state 3, we obtain \( x,y \)

A test case which allow the detection of the transfer error is: \( a,b,b,b \)
If we apply it from the initial state of:
- the specification, we obtain \( x,x,y,y \)
- the implementation, we obtain \( x,x,x,x \)
Example 2

The test cases ($\beta$-sequences) are: (no reset)

state 1: $a, b, b$
  $b, b, b$
state 2: $b, a, b, b$
  $b, b, b, b$
State 3: $a, a, b, b$
  $a, b, b, b$

Test case structure corresponding to 3-step:
preamble, tested transition, state identification

Transfer sequence (Preamble): the minimum-cost (shortest path) input sequence taking FSM fro one state to another.
Example

DS = B, B

The $\beta$-sequences are:

- $r, A, B, B$
- $r, B, B, B$
- $r, A, A, A, A, A, B, B, B$
- $r, A, A, A, A, B, B, B$
- $r, A, A, A, B, B, B$
- $r, A, A, A, B, B, B$
- $r, A, A, B, B, B$
- $r, A, A, A, B, B, B$
- $r, A, A, B, B, B$

An optimized test sequence constructed from above is: $r$AAAABBB$r$AAAABBB

$r$AAAABBB$r$AAABB$r$AABB$r$ABB$r$BBB
D Method

- Distinguishing Sequence (DS): Represents Input string $X$ of machine if the output string produced by machine in response to $X$ is different for each starting state.

- Similar to U method, but at every occurrence of UIO sequence for state replaced with the DS.

$X=BB$ is shortest DS

- $S_1\rightarrow 1, 0$
- $S_2\rightarrow 0, 1$

$S_3\rightarrow S_3\rightarrow S_0\rightarrow S_3\rightarrow S_3\rightarrow r. A B BB(DS)$

$S_0\rightarrow S_0\rightarrow S_0\rightarrow r. B BB$
D Method

<table>
<thead>
<tr>
<th>State</th>
<th>DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>λλ</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>1λ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transitions</th>
<th>Test subsequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>state S3 to S3</td>
<td>r A BB</td>
</tr>
<tr>
<td>state S0 to S0</td>
<td>r B BB</td>
</tr>
<tr>
<td>state S1 to S4</td>
<td>r AAAAA A BB</td>
</tr>
<tr>
<td>state S1 to S2</td>
<td>r AAAAA B BB</td>
</tr>
<tr>
<td>state S2 to S1</td>
<td>r AAA A BB</td>
</tr>
<tr>
<td>state S4 to S3</td>
<td>r AA AB BB</td>
</tr>
<tr>
<td>state S3 to S4</td>
<td>r A A BB</td>
</tr>
<tr>
<td>state S3 to S2</td>
<td>r A AA BB</td>
</tr>
<tr>
<td>state S4 to S0</td>
<td>r AA B BB</td>
</tr>
<tr>
<td>state S4 to S2</td>
<td>r AA A BB</td>
</tr>
<tr>
<td>state S3 to S0</td>
<td>r A AB BB</td>
</tr>
</tbody>
</table>
D - Method

- **Algorithm**: Test subsequence generation using D-method

- **Begin**
  - apply reset input $r$ to $M$ so that $M$ is reset to initial state $0$;
  - If the transition starts at state $s_i$, then, find the shortest path $SP(s_i)$ from state $S_0$ to state $s_i$;
  - apply an input symbol such that $M$ makes a state transition from state $s_i$ to state $s_j$;
  - apply DS sequence for state $s_j$.

- **End**
W Method

For FSMs that do not possess a DS, W-Method defines partial DS each of which distinguishes a state $s_i$ from a subset of the remaining states instead of from every state of the FSM.

The states of the FSM are first partitioned into blocks which can be distinguished by observing the sequence of outputs produced by a sequence of inputs.

Each block is subsequently partitioned into distinguishable sub-blocks, and so on, until each block consists of exactly one state.

Main idea is to iteratively find a DS for each subset.

To identify a state (for step 3):
- Applying an input sequence
- Returning to the state via a transfer sequence
- Applying a second input sequence, and so on

The complete set of such input sequences for an FSM is called the characterizing set. Attach each CS in the set to the end of each transfer sequence.
W Method

Example

A characterizing set \( W = \{a, b\} \)
- for state 1: a/e, b/f
- for state 2: a/f, b/f
- for state 3: a/f, b/e

The \( \beta \)-sequences generated are:

\[
\begin{align*}
& r, a & r, b \\
& r, a, a & r, a, b \\
& r, b, a & r, b, b \\
& r, c, a & r, c, b \\
& r, b, a, a & r, b, a, b \\
& r, b, b, a & r, b, b, b \\
& r, b, c, a & r, b, c, b \\
& r, c, a, a & r, c, a, b \\
& r, c, b, a & r, c, b, b \\
& r, c, c, a & r, c, c, b \\
\end{align*}
\]
W Method

- Characterization Set based method. Use Character set instead of U and DS method.
- \( W = \{A, AA, B\} \)
- \( S_0 \rightarrow 0, 0, T \)
- \( S_1 \rightarrow 1,1,1 \)
- \( S_3-S_4 \rightarrow r. A A A; r A A AA, r A A B \)
- \( S_0-S_0 \rightarrow r. B A; r B AA ; r B B \)
**W - Method**

<table>
<thead>
<tr>
<th>State</th>
<th>$M_s(A)$</th>
<th>$M_s(AA)$</th>
<th>$M_s(B)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Last output symbols on W for M

<table>
<thead>
<tr>
<th>Transitions</th>
<th>Test subsequences with input strings A, AA and B</th>
</tr>
</thead>
<tbody>
<tr>
<td>state S3 to S3</td>
<td>r A A; r A AA; r A B</td>
</tr>
<tr>
<td>state S0 to S0</td>
<td>r B A; r B AA; r B B</td>
</tr>
<tr>
<td>state S1 to S4</td>
<td>r AAAA A A; r AAAA A AA; r AAAA A B</td>
</tr>
<tr>
<td>state S1 to S2</td>
<td>r AAAA B A; r AAAA B AA; r AAAA B B</td>
</tr>
<tr>
<td>state S2 to S1</td>
<td>r AAA A A; r AAA A AA; r AAA A B</td>
</tr>
<tr>
<td>state S4 to S3</td>
<td>r AA AB A; r AA AB AA; r AA AB B</td>
</tr>
<tr>
<td>state S3 to S4</td>
<td>r A A A; r A A AA; r A A B</td>
</tr>
<tr>
<td>state S3 to S2</td>
<td>r A AA A; r A AA AA; r A AA B</td>
</tr>
<tr>
<td>state S4 to S0</td>
<td>r AA B A; r AA B AA; r AA B B</td>
</tr>
<tr>
<td>state S4 to S2</td>
<td>r AA A A; r AA A AA; r AA A B</td>
</tr>
<tr>
<td>state S3 to S0</td>
<td>r A AB A; r A AB AA; r A AB B</td>
</tr>
</tbody>
</table>

Test subsequences for W-method
W - Method

- **Algorithm:** Test subsequence generation using D-method

- Begin

  - apply reset input $r$ to $M$ so that $M$ is reset to initial state $S_0$;
  - If the transition starts at state $s_i$, then, find the shortest path $SP(s_i)$ from state $S_0$ to state $s_i$;
  - apply an input symbol ($z$) such that $M$ makes the state transition from state $s_i$ to state $s_j$;
  - apply $W$ sequence for state $s_j$ with understanding that,
  - $S@W = S{@_{\alpha_1}; ....; \alpha_k}$
    - $= \{S@_{\alpha_1}; S@_{\alpha_2}; ......; S@_{\alpha_k}\}$
  - Where $S$ is concatenation of the inputs used in first three steps of algorithm, i.e.,
  - $S = r@SP(si)@z$.

- End
• Fault testing coverage
• Fault coverage for D, W and U is better than of T method
• Fault coverage for D, W and u method are the same
• All of these four methods assume minimal, strongly connected and fully specified Mealy FSM model of protocol entities
• On average, T-method produces the shortest test sequence, W-method the longest. D- and U- methods generate test sequence of comparable lengths
• T method
• T-method test sequences are able to detect output faults but not transition
• D-, W-, and U-methods are capable of detecting all kinds of faults and give the same performance.
• U-method attracts more and more attentions and there are several approaches based on the basic idea with some improvements