VI. Protocol Testing

Outline

- Introduction
  - Concepts
  - Fault models
  - Related definitions
  - General approach

- Methodologies based on FSM
  - T-Method (Transition Tour method)
  - D-Method (Distinguishing sequences)
  - W-Method (Characterizing sequences)
  - U-Method (Unique input/output sequences)

Protocol Conformance Testing

- To confirm if an implementation conforms to its standard
  - External tester applies a sequence of inputs to IUT and verifies its behavior
  - Issue 1: preparation of conformance tests in coverage of IUT’s all aspects
  - Issue 2: time required to run 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 Models

A fault model is a hypothetical model of what types of faults may occur in an implementation.
- Most fault models are "structural"; i.e., the model is a refinement of the specification formalism (or of an implementation model).
  - E.g. mutations of the specification or of a correct implementation.
- It may be used to construct the fault domain used for defining what "complete test coverage" means.
  - E.g. single fault hypothesis (or multiple faults).

A fault model is useful for the following problems:
- Test suite development for given coverage objective.
- Formalization of "test purpose".
- For existing test suite: coverage evaluation and optimization.
- Diagnostics.

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.

Fault Models for FIFO Queue and Petri Nets

FSM with several FIFO input queues
- Ordering fault: FIFO ordering is not preserved, or in case of multiple input queues, some input event enters a wrong input queue.
- Maximum length fault: the maximum length implemented is less than the one specified, or if an input event gets lost while queue is not overflow.
- Flow control fault: errors of ordering or of loss occur, in case the number of submitted input events overflows the maximum queue length specified.

Petri Nets
- Input or output arc fault: one of the input or output arcs is connected to the wrong place, missing, or exists in addition to those specified.
- Missing or additional transition: the number of transitions is not the same as in the specification.

FSM Related Definitions (1/2)

Directed graph \(G=(V, E)\) representing FSM \(M\)
- Set of vertices \(V = \{v_1, v_2, ..., v_n\}\) represents the set of states \(S\) in \(M\).
- Directed edge \((v_i, v_j) \in E\) represent a transition from state \(s_i\) to state \(s_j\) in \(M\).
- An edge in \(G\) is represented by a triple \((v_i, v_j, L)\), \(L=a_0/o_0\) is the input/output operation corresponding to the transition from \(s_i\) to \(s_j\) in \(M\).

Some other definitions & assumptions
- Deterministic FSM: predictable behavior in a given state for a given input.
- Strongly connected: for each state pair \((s_i, s_j)\) there is a transition path going from \(s_i\) to \(s_j\), i.e. each state can be reached from any other state.
- Fully specified: form each state it has a transition for each input symbol. Otherwise partially specified.
- Minimal: the number of states of M is less than or equal to the number of states of any equivalent machine.
FSM Related Definitions (2/2)

- **Start state** \( s_0 \in S \), usually the state when power-up
  - Often, there is a special input taking \( M \) to state \( s_0 \) from any other state with a single transition. In this case, \( M \) is said to have the **reset capability** and the input which performs the reset is denoted by “\( r \)”
- **Sequences for testing**
  - A **test subsequence** of \( M \) is a sequence of input symbols for testing either a state or a transition of \( M \)
  - A **\( \beta \)-sequence** for \( M \) is a concatenation of test subsequences for testing all transitions of \( M \)
  - A **test sequence** for \( M \) is a sequence of input symbols which can be used in testing conformance of implementations of \( M \) against the specification of \( M \)
  - An **optimize test sequence** is a test sequence such that no subsequence of it is completely contained in any other subsequence
- **So, the problem is how to obtain a “optimize test sequence” for \( M \)**

Outline

- **Introduction**
  - Concepts
  - Fault models
  - Related definitions
  - General approach
- **Methodologies based on FSM**
  - **T-Method** (Transition Tour method)
  - **D-Method** (Distinguishing sequences)
  - **W-Method** (Characterizing sequences)
  - **U-Method** (Unique input/output sequences)

Transition Level Approach

- **The methods for protocol conformance test sequence generation**
  - Produce a test sequence which checks the correctness of each transition of the FSM implementation
  - By no means exhaustive, i.e. no guarantee to exhibit correct behavior given every possible input sequence. The intent is to design a test sequence which guarantees “beyond a reasonable doubt”
- **Three basic steps for checking a transition** \( (s_i, s_j; L), L = a_i/o_r \)
  - **Step 1**: The FSM implementation is put into state \( s_i \) (e.g. reset+transfer)
    - Difficulty in realizing this is due to the limited controllability of the implementation
  - **Step 2**: Input \( a_i \) is applied and the output is checked to verify that it is \( o_r \) as expected;
  - **Step 3**: The new state of the FSM implementation is checked to verify that it is \( s_j \) as expected
    - Difficulty in verifying this is due to the limited observability of the implementation

T-Method: Transition Tour Method

- **For a given FSM \( S \), a transition tour \( L \) is a sequence which takes the FSM \( S \) from the initial state \( s_0 \), traverses every transition at least once, and returns to the initial state \( s_0 \).**
  - 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
- **The problem of generating a minimum-cost test sequence using the transition tour method is equivalent to the so-called “Chinese Postman” problem in graph theory**
  - First studied by Chinese mathematician Kuan Mei-Ko (管梅谷) in 1962
The implementation \( I_1 \) contains an output error. Our transition tour will detect it.

The implementation \( I_2 \) contains a transition error. Our transition tour will not detect it.

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

\[
1 \rightarrow r, a, r, c, a, b, b, r, c, a, b, b, r, c, a, b, b, r, c, a, b, b, r, c, a, b, b, r, c, a, b, b, r, c, a, b, b
\]

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. A DS is used as a state identification sequence. A DS is a useful tool for checking Step 3 in detecting output and transfer errors. Detects all output errors. Detects all transfer errors. 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. The requirement is too strong (leading to W- & U- methods…).
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: $a,b,b$
If we apply it from the initial state of:
- the specification, we obtain $x,x,y,y$
- the implementation, we obtain $x,x,x,x$

The test cases ($\beta$-sequences) are:

1. $r,a,c,a$
2. $r,c,a,b,a,a,c,a$
3. $r,c,a,b,a,a,c,a$
4. $r,c,a,b,a,b,c,a$
5. $r,c,a,b,a,c,a$

An optimized test sequence constructed from above is: $rAAAAABBrAAAABBBrABBBBrABBBBrABBB$
W-Method: Characterizing Sequences

- 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.
- 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 – 1

For the input sequence \( \text{Acs}1 = \text{A,A} \), the response is identical for states 2 and 3 (10), but is distinct from that for states 0(00), 1(11), 4(10).

Another input sequence \( \text{Acs}2 = \text{B} \) is distinct for states 2(0) and 3(1).
Therefore, \( \text{Acs}1 \) is required to identify states 0, 1, 4, and two input sequences \( \text{Acs}1 \) and \( \text{Acs}2 \), along with appropriate transfer sequences, are required to identify states 2 and 3. I.e. \( W = \{\text{AA, B}\} \).

W-Method Example – 2

No DS!
For the input sequence \( \text{Acs}1 = 0,1,0 \), the response is identical for states C and D (101), but is distinct from that for states A(000) and B(001).

Another input sequence \( \text{Acs}2 = 1,0 \) is distinct for states C(00) and D(01).
Therefore, \( \text{Acs}1 \) is required to identify states A and B, and two input sequences \( \text{Acs}1 \) and \( \text{Acs}2 \), along with appropriate transfer sequences, are required to identify states C and D. I.e. \( W = \{010, 10\} \).

W-Method Example – 3

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:
- \( a \) \( a \)
- \( b \) \( b \)
- \( r, a, a \) \( r, a, b \)
- \( r, b, a \) \( r, b, b \)
- \( r, c, a \) \( r, c, b \)
- \( r, b, a, a \) \( r, b, a, b \)
- \( r, b, b, a \) \( r, b, b, b \)
- \( r, c, a, a \) \( r, c, a, b \)
- \( r, c, b, a \) \( r, c, b, b \)
- \( r, c, c, a \) \( r, c, c, b \)
U-Method: Unique Input/Putout Sequences

- In DS and CS, requirement of state identification is too strong
  - Answer the question of "what is the current state of the implementation?"
  - 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
  - Answer the question of "is the implementation currently in state x?"

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'
Analysis

- Fault Testing Coverage
  - Fault coverage for D-, W-, and U-methods is better than T-method
  - Fault coverage for D-, W-, and U-methods are the same

- Summary
  - 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 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

References