EXPERIMENTAL EVALUATION OF FSM-BASED TESTING METHODS

Rita Dorofeeva 2Khaled El-Fakih 3Stephane Maag 3Ana R.Cavalli 1Nina Yevtushenko
1Tomsk State University, Russia, 2American University of Sharjah, UAE, 3Institut National des Télécommunications, France
drf@kitidis.tsu.ru, yevtushenko@elefot.tsu.ru,
kelfakih@aus.ac.ae {Ana.Cavalli,Stephane.Maag}@int-evry.fr

Abstract

The development of test cases is an important issue for testing software, communication protocols and other reactive systems. A number of methods are known for the development of a test suite based on a formal specification given in the form of a finite state machine. Well-known methods are called the W, Wp, UIO, UIOv, DS, H and HIS test derivation methods. These methods have been extensively used by research community in the last years; however no proper comparison has been made between them. In this paper, we experiment with these methods to assess their complexity, applicability, completeness, fault detection capability, length and derivation time of their test suites. The experiments are conducted on randomly generated specifications and on a realistic protocol called the Simple Connection Protocol.

1. Introduction

The development of test cases based on a formal model is an important issue for software testing including conformance testing of communication protocols and other reactive systems. A number of methods are known for the development of a test suite based on a specification given in the form of a finite state machine. Well known methods are called the W, Wp, UIO, UIOv, DS, H and HIS test derivation methods [4, 6, 15, 19, 22, 21, 17]. For related surveys the reader may refer to [18, 1, 20, 13]. In particular, in the last years an increasing research has been developed on the application of these methods to object oriented software [2].

In FSM-based testing, one usually assumes that not only the specification, but also an implementation can be modeled as a deterministic FSM. If the behavior of an implementation FSM is different than the specified behavior, the implementation contains a fault. Two types of implementation faults are usually considered, namely output and transfer faults. An implementation has an output fault if the output of one of its transitions is different from that of the specification and an implementation has a transfer fault if the next state of a transition is different from that of the specification. Moreover, an implementation has multiple faults if it has many output or transfer faults [1, 4, 6, 15, 18, 22, 21].

The above methods, except the UIO method, each provides the following fault coverage guarantee: If the specification can be modeled by an FSM with n states and if an implementation can be modeled by an FSM with at most m states, where m is larger or equal to n, then a test suite can be derived by the method (for this given m) and the implementation passes this test suite if and only if it conforms (i.e. is equivalent) to the specification (that is, the implementation does not contain output and transfer faults). Moreover, all of the above methods assume that a reliable reset is available for each implementation under test (written as ‘r’). This implies that the test suite can be composed of several individual test cases, each starting with the reset operation.

All of the above methods use certain state distinguishing input sequences (called state identifiers) in test derivation, and thus, can be only applied when all states of the specification FSM are pairwise distinguishable. Moreover, the length of a derived test suite essentially depends how state identifiers are selected. The DS method can be thought of as a particular case of the W method. However, the DS method is not always applicable even for complete reduced specifications [7]. The Wp method is an improvement to the W method and thus is expected to generate shorter test suites. Also, the UIOv method can be thought of as a particular case of the Wp and is expected to generate shorter test suites. The UIO method employs part of the UIOv method and thus generates shorter test suites. However, UIO test suites are not always complete [22]. In the so-called H method [12, 5], unlike all other methods, for
2. Finite State Machines

This section contains the definition of basic concepts that are used in the rest of the paper.

Definition 2.1 A deterministic finite state machine is an initialized complete deterministic Mealy machine that can formally be defined as a 6-tuple \( M = (S, X, Y, \delta_M, \lambda_M, s_1) \) [7], where \( S \) is a finite set of states, \( s_1 \) is the initial state, \( X \) is a finite set of input symbols, \( Y \) is a finite set of output symbols, \( \delta_M \) is a next state (or transition) function: \( \delta_M: S \times X \rightarrow S \), \( \lambda_M \) is an output function: \( \lambda_M: S \times X \rightarrow Y \). In usual way, functions \( \delta_M \) and \( \lambda_M \) are extended to input sequences.

Definition 2.2 A FSM \( A \) is called connected if for each state \( s \in S \) there exists an input sequence \( \alpha_s \) that takes FSM \( A \) from the initial state to state \( s \). The sequence \( \alpha_s \) is called a transfer sequence for the state \( s \).

Definition 2.3 A set \( Q \) of input sequences is called a state cover set of FSM \( M \) if for each state \( s_i \) of \( S \), there is an input sequence \( \alpha_i \in Q \) that takes FSM from the initial state to state \( s_i \). If FSM is connected, i.e. if each state is reachable from the initial state, then a state cover set always exists. We further assume that the specification FSM \( M \) is a connected FSM\(^1\) and consider only prefix-closed state cover sets, i.e. a state cover set contains all prefixes of each sequence.

Definition 2.4 Let \( M = (S, X, Y, \delta_M, \lambda_M, s_1) \) and \( I = (T, X, Y, \delta_I, \lambda_I, t_1) \) be two FSMs. In the following sections \( M \) usually represents a specification while \( I \) denoting an implementation. We say that two states \( s_j \) of \( M \) and \( t_i \) of \( I \) are equivalent [7], written \( s_j \equiv t_i \), if for each input sequence \( \alpha \in X^* \) it holds that \( \lambda_M(s_j, \alpha) = \lambda_I(t_i, \alpha) \). Otherwise, we say that states \( s_j \) and \( t_i \) are distinguishable, written \( s_j \not\equiv t_i \).

Definition 2.5 An input sequence \( \alpha \in X^* \) such that \( \lambda_M(s_j, \alpha) \neq \lambda_I(t_i, \alpha) \) is said to distinguish the states \( s_j \) and \( t_i \). FSMs \( M \) and \( I \) are equivalent, written \( M \cong I \), (distinguishable, written \( M \not\cong I \)) if their initial state are equivalent (distinguishable).

Definition 2.6 An FSM is said to be reduced if its states are pair-wise distinguishable.

Definition 2.7 We say that \( I \) conforms to \( M \) if and only if FSMs \( I \) and \( M \) are equivalent. In other words, for each input sequence the output responses of \( M \) and \( I \) coincide [7, 17, 18].

Definition 2.8 Given a specification FSM \( M \), the fault domain \( J(X) \) of \( M \) is the set of all possible implementations of \( M \) defined over the input alphabet \( X \) of \( M \). Similar to [16] we let \( J_m(X) \) denote the set of all complete FSMs defined over the input alphabet \( X \) with up to \( m \) states. A test suite \( TS \) is a finite set of finite input sequences of the specification FSM \( M \).

Definition 2.9 A test suite \( TS \) is \( m \)-complete if for each implementation \( I \in J_m(X) \) that is distinguishable from \( M \), there exists a sequence in \( TS \) that distinguishes \( M \) and \( I \).

\(^1\) We consider only connected FSMs without loss of generality, since any state of an FSM that is unreachable from the initial state, does not influence the behavior of the FSM.
3. Overview of Test Derivation Methods

This section includes an overview of test derivation methods. All these methods, except the UIO, have two phases. Tests derived for the first phase check that each state presented in the specification also exists in the implementation, while tests derived for the second phase check all (remaining) transitions of the implementation for correct output and ending state as defined by the specification. For identifying the state during the first phase and for checking the ending states of the transitions in the second phase, certain state distinguishing input sequences are used.

The only difference between the above methods is how such distinguishing sequences are selected. In the original W method, a so-called characterization set $W$ (or simply $W$ set) is used to distinguish the different states of the specification. The Wp method uses the $W$ set during the state identification phase (the first phase) while only an appropriate subset, namely a corresponding state identifier, is used when checking the ending state of a transition. In the HIS method a family of state identifiers is used for state identification as well as for transition checking. In the UIOv method, which is a proper sub-case of the Wp method, the state identifier of each state has a single UIO method, which is a proper sub-case of the Wp method, the state identifier of each state has a single UIO sequence. Such an UIO method, the state identifier of each state has a single UIO sequence and then the set $\{\alpha_1, \alpha_2, ..., \alpha_n\}$ is known to be a $W$ set. We note that a UIO sequence may not exist for some states of a reduced FSM.

Let $\alpha$ be an input sequence such that for any two states $s_i, s_j, i \neq j$, $\lambda_M(s_i, \alpha) = \lambda_M(s_j, \alpha)$. Then $\alpha/\beta_i$ where $\beta_i = \lambda_M(s_i, \alpha)$, is said to be a Simple Input/Output Sequence [9] or a Unique Input/Output (UIO) sequence [17], for state $s_i$. Let each state of the specification FSM have an UIO $\alpha_i/\beta_i$ and $\{\alpha_1, \alpha_2, ..., \alpha_n\}$ be the set of input parts of all these UIOs. Then the set $\{\alpha_1, \alpha_2, ..., \alpha_n\}$ is known to be a $W$ set. In order to obtain a separating family from these identifiers, we

Given state $s_j \in S$ of FSM $M$, a set $W_j$ of input sequences is called a state identifier (or a separating set) of state $s_j$ if for any other state $s_i$ there exists $\alpha \in W_j$ such that $\lambda_M(s_i, \alpha) \neq \lambda_M(s_j, \alpha)$. A separating family [20] (or a family of harmonized identifiers [14, 15]) is a collection of state identifiers $W_j, s_j \in S$, which satisfy the condition: For any two states $s_j$ and $s_i, i \neq j$, there exist $\beta \in W_j$ and $\gamma \in W_i$ which have common prefix $\alpha$ such that $\lambda_M(s_i, \alpha) \neq \lambda_M(s_j, \alpha)$. A separating family always exists for a reduced FSM.

In subsection 3.2 we briefly describe the test derivation methods [4, 6, 15, 22, 12]. Meanwhile, we describe the state identification utilities used by these methods.

3.1. State identification facilities

In order to check that each state and each transition defined in the specification also exists in the implementation, the methods use certain input/output behaviors that can distinguish the states of an FSM. Consider a reduced specification FSM $M = (S, X, Y, \delta_M, \lambda_M, S_1)$.

A characterization set of the FSM $M$, often simply called a $W$ set, is a set of input sequences defined over the input alphabet of $M$ that satisfies the condition: For any two states $s_i$ and $s_j, i \neq j$, $\exists \beta \in W$ such that $\lambda_M(s_i, \beta) \neq \lambda_M(s_j, \beta)$. That is for any two states of $M$, the $W$ set includes a sequence that distinguishes these states. A $W$ set always exists for a reduced FSM.

The specification $M$ admits the set of sequences $\{x, y, xy\}$ as a $W$ set. From the above, we get the following state identifiers, $W_1 = \{y\}, W_2 = \{y\}, W_3 = \{x\}$, and $W_4 = \{x, yy\}$. In order to obtain a separating family from these identifiers, we

![Figure 1. Specification FSM](image-url)
harmonize $W_2$ and $W_3$ by adding the input $x$ into $W_1$ and $W_2$ and $y$ into $W_3$, and obtain as a separating family the set $F = \{H_1, H_2, H_3, H_4\}$, where $H_1 = \{x, y, y\}$, $H_2 = \{x, y\}$, $H_3 = \{x\}$, and $H_4 = \{x, y\}$.

The FSM $M$ has the sequence $ywy$ as a distinguishing sequence. For states $s_1$, $s_2$, $s_3$, and $s_4$ of $M$, we have, in response to $ywy$, the output sequences $010, 100, 001$, and $000$, respectively.

3.2 Test Derivation Methods

Given a reduced complete specification FSM $M = (S, X, Y, \delta_M, \lambda_M, s_1), |S| = n$, let $W$ be a characterization set of $M$ and $F = \{W_1, \ldots, W_n\}$ be a separating family of $M$.

The test derivation methods have two phases in order to test the equivalence of $I$ and $M$.

Given a reduced complete specification FSM $M = (S, X, Y, \delta_M, \lambda_M, s_1), |S| = n$, let $W$ be a characterization set of $M$ and $F = \{W_1, \ldots, W_n\}$ be a separating family of $M$.

The test derivation methods have two phases in order to test the equivalence of $I$ and $M$.

**State identification phase:**

This phase checks that each state specified by $M$ also exists in $I$ using a characterization set $W$ ($W$, $W_p$, UIO, and DS methods), or a separating family $F$ (HIS method). We note that for the UIOv method, the $W$ set consists of the input parts of the UIOs of $M$ and for the DS method the $W$ set consists of a single sequence, namely a DS of $M$.

Given a prefix-closed state cover set $Q = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ of the specification FSM, for each state $\alpha_j \in S$, the state identification phase comprises the sequences:

- $r.\alpha_j.H_j$ (in the HIS method) or

Each test sequence starts from the initial state, after the application of the reset input $r$. In this case, in the $W$, $W_p$, DS and UIO methods, to identify the ending state $\alpha_j$ after applying an input sequence $\alpha_j$, all the sequences contained in $W$ are applied to $I$, separately. However, in the HIS it is enough to apply the sequences of the state identifier set $H_j$ of $\alpha_j$.

**Transition testing phase:**

This phase assures that for each transition of $M$ there exists a corresponding transition in $I$. For this purpose, for each sequence $\alpha_j \in Q$ that takes the specification FSM to appropriate state $s_j$, and each $x \in X$ that takes the $M$ from state $s_j$ to state $s_k$, the transition testing phase includes the following set of test sequences:

- $r.\alpha_j.x.H_k$ in the HIS method,
- $r.\alpha_j.x.W_k$ in the UIOv and $W_p$ methods, where $W_k \subseteq W$ is a state identifier or a corresponding UIO of the state $s_k$ or
- $r.\alpha_j.x.W$ in the $W$ method.

If FSM $I = (T, X, Y, \Delta_I, A_I, t_i)$ has at most $n$ states and passes the test sequences of both testing phases, then $I$ is equivalent to the specification FSM, i.e. $I$ is a conforming implementation.

The UIO method [17] is based on UIO sequences and it employs only the second testing phase. It was originally claimed [17] that a test suite derived using the UIO method is complete w.r.t. all implementations with up to $n$ states. However, a counter-example was later given in [22] that proves that the UIO method does not guarantee complete fault coverage.

The $H$ method [12, 5] can be regarded as an improvement to the HIS method. The main idea of the $H$ method is not to use a priori derived state identifiers. State identifiers are constructed, based on already derived test cases, in order to distinguish the ending states of transitions. Thus, different state identifiers can be used when testing transitions with the same ending state.

**Example:** As an application example of the above methods, consider the FSM $M$ of Figure 1. We recall that $M$ admits as a characterization set the set $W = \{x, y, yy\}$, as state identifiers the sets $W_1 \equiv \{yy\}$, $W_2 = \{y\}$, $W_3 = \{x\}$, and $W_4 = \{x, yy\}$, and the set $F = \{H_1, H_2, H_3, H_4\}$ as a separating family of harmonized state identifiers, where $H_1 = \{x, yy\}$, $H_2 = \{x, y\}$, $H_3 = \{x\}$, and $H_4 = \{x, yy\}$. Moreover, $M$ has the state cover set $Q = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$, where $\alpha_1 = e, \alpha_2 = y, \alpha_3 = x$, and $\alpha_4 = yy$.

**W method application:**

Based on the above sets, in the $W$ method, the state identification phase yields the test sequences $TS_W: r.\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}.W$ and the transition testing phase yields the test sequences: $r.\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}\{x, y\}.W$.

We replace the $e$’s and $W$’s in the above sequences by their corresponding values and then remove from the obtained set those sequences that are proper prefixes of other sequences and obtain $TS_{W'} = \{xxxx, rxxx, rxxxy, rxxyy, rxy, rxyy, ryxx, ryyxx, ryyxy, ryyyy\}$ of total length 49.
Wp method application:
In the Wp method, in addition to the state identification sequences, the transition testing phase yields the sequences \( TSWp \): \( r.\alpha_1.x.W_3 + r.\alpha_1.y.W_2 + r.\alpha_2.x.W_2 + r.\alpha_2.y.W_4 + r.\alpha_3.x.W_2 + r.\alpha_3.y.W_1 + r.\alpha_4.x.W_3 + r.\alpha_4.y.W_4 \). We replace the \( \alpha \)'s and \( W \)'s by their corresponding values and obtain \( TSWp = \{rxxy, rxyyy, rxxy, ryyx, ryyxy, ryyyyy\} \) of total length 29. Similarly, we apply the HIS method and obtain the test suite \( TSHIS = \{rxxx, rxx, rxx, rxxx, ryy, ryyx, ryyxx, ryyyyy\} \) of total length 41. We note here that for this example, the first parts for the state identification phases in Wp and HIS methods coincide and for this reason the Wp returns a shorter test suite knowing that we do not need to harmonize, as in the HIS, the state identifiers.

H method application:
In the H method, in the state identification phase, we obtain the sequences \( \{rxx, rxy, rxx, ryy\} \) using the same state identifiers \( W_1 = \{y\}, W_2 = \{y\}, W_3 = \{x\}, \) and \( W_4 = \{x, y\} \). In the transition testing phase, consider the transition from state \( s_3 \) under input \( x \) with the final state \( s_2 \). In order to test this transition, we apply the input sequence \( r.\alpha_3.x.y = r.xx.y \) instead of applying the sequences \( r.\alpha_3.x.H_2 = r.xx.x\{x, y\} \) as in the HIS method. This is done since the input \( y \) is already applied, in the state identification phase, at each state of the implementation (if the implementation passes the sequences of the state identification phase) and thus we can use \( y \) in the transition testing phase instead of using as in the HIS method the harmonized state identifier \( H_2 = \{x, y\} \). Similarly, we derive identifiers to check all other transitions. The obtained test suite \( TSH = \{rxxy, ryy, rxx, ryyyyy\} \) is of total length 25.

DS method application:
When constructing a characterization set we derive a shortest distinguishing sequence for each pair of different states of the specification FSM. However, sometimes such sequences do not yield the shortest test suite. By direct inspection, one can assure that the FSM \( M \) of our working example has a distinguishing sequence \( yyy \). Thus, we can select as harmonized state identifiers the sets \( Z_1 = \{yy\}, Z_2 = \{y\}, Z_3 = \{yyy\}, \) and \( Z_4 = \{yyy\} \). In this case, the HIS method returns a test suite \( TS_{DS} = \{rxxy, rxyy, rxx, ryyxx, ryyyyy\} \) with total length 27.

4. Experimental Results
In this section we experiment with the above described methods with the following objectives: (i) determine and compare the length and derivation time of their test suites, (ii) determine how often the UIO and DS methods are not applicable and how often the UIO method (when applicable) generates incomplete test suites, (iii) determine the fault detection capability of some UIO-based incomplete test suites, and (iv) compare length of the obtained test suites with the theoretical upper-bound.

For every pair of (different) states of \( M \), we generate a shortest input sequence that distinguishes this pair \([7, 19]\). The set of all obtained distinguishing sequences is a W set of the specification FSM \( M \). The subset of all obtained sequences that distinguish a state \( s_i \) from all other states of \( M \) is a state identifier \( (W_i) \) of \( s_i \). The set of all state identifiers is a separating family \( (F) \) of \( M \). An algorithm for deriving a distinguishing sequence (if exists) for a given FSM is given in \([11]\). We derive UIO sequences using the distinguishing tree algorithm given in \([7]\).

Table 1 provides a comparison between the length of test suites obtained by these methods and their derivation time in seconds. The comparison is based on randomly generated completely specified reduced specifications with a varying number of inputs/outputs and states \( \{n\} \).

Each row of Table 1 corresponds to a group of 50 randomly generated completely specified reduced specifications. For each of these specifications we use the W, Wp, HIS, UIOV and H methods to derive corresponding test suites. Moreover, we also derive test suites using the UIO and DS methods when the specifications have UIO and DS sequences. Then, we calculate the average length and derivation time (in seconds) of test suites generated for each group using each of these methods as shown in Columns VI to XII, respectively. We also calculate for each group how many times (out of 50) the UIO and DS methods were applicable and the average length for their test suites as shown in Columns XIII and XIV, respectively.
<table>
<thead>
<tr>
<th>Group</th>
<th>State, n</th>
<th>Inputs, k</th>
<th>Outputs</th>
<th>Transitions</th>
<th>WT Suites</th>
<th>WP Suites</th>
<th>UHS Suites</th>
<th>UIOV Suites</th>
<th>H Suites</th>
<th>UIO Suites</th>
<th>DS Applicable</th>
<th>Average Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td>6</td>
<td>6</td>
<td>180</td>
<td>2545</td>
<td>1626</td>
<td>1649</td>
<td>1406</td>
<td>1105</td>
<td>783</td>
<td>934</td>
<td>0.63 (0.06)</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>8</td>
<td>8</td>
<td>240</td>
<td>2645</td>
<td>1919</td>
<td>1946</td>
<td>1406</td>
<td>1320</td>
<td>1012</td>
<td>1281</td>
<td>1.09 (1.10)</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>10</td>
<td>10</td>
<td>300</td>
<td>2893</td>
<td>2175</td>
<td>2243</td>
<td>1477</td>
<td>1568</td>
<td>1196</td>
<td>1493</td>
<td>1.48 (1.54)</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>4</td>
<td>4</td>
<td>200</td>
<td>3393</td>
<td>2635</td>
<td>4077</td>
<td>1711</td>
<td>996</td>
<td>0</td>
<td>46</td>
<td>1.15 (2.12)</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>6</td>
<td>6</td>
<td>300</td>
<td>5203</td>
<td>3261</td>
<td>3447</td>
<td>2142</td>
<td>1383</td>
<td>1777</td>
<td>50</td>
<td>2.98 (3.06)</td>
</tr>
<tr>
<td>6</td>
<td>50</td>
<td>8</td>
<td>8</td>
<td>400</td>
<td>6049</td>
<td>3828</td>
<td>3167</td>
<td>2534</td>
<td>1820</td>
<td>2177</td>
<td>6</td>
<td>4.49 (2.54)</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>10</td>
<td>10</td>
<td>500</td>
<td>6773</td>
<td>4305</td>
<td>3126</td>
<td>2852</td>
<td>2233</td>
<td>2710</td>
<td>50</td>
<td>4.93 (3.61)</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>6</td>
<td>6</td>
<td>360</td>
<td>7814</td>
<td>4844</td>
<td>4250</td>
<td>3185</td>
<td>2229</td>
<td>2660</td>
<td>50</td>
<td>1.63 (3.14)</td>
</tr>
<tr>
<td>9</td>
<td>60</td>
<td>8</td>
<td>8</td>
<td>480</td>
<td>8978</td>
<td>5443</td>
<td>4178</td>
<td>3656</td>
<td>2758</td>
<td>3269</td>
<td>12</td>
<td>7.69 (6.20)</td>
</tr>
<tr>
<td>10</td>
<td>60</td>
<td>10</td>
<td>10</td>
<td>600</td>
<td>9277</td>
<td>5899</td>
<td>5389</td>
<td>4200</td>
<td>3584</td>
<td>4365</td>
<td>15</td>
<td>11.05 (9.18)</td>
</tr>
<tr>
<td>11</td>
<td>70</td>
<td>6</td>
<td>6</td>
<td>420</td>
<td>8245</td>
<td>5047</td>
<td>4047</td>
<td>3274</td>
<td>2901</td>
<td>3625</td>
<td>15</td>
<td>6.76 (5.32)</td>
</tr>
<tr>
<td>12</td>
<td>70</td>
<td>8</td>
<td>8</td>
<td>560</td>
<td>11012</td>
<td>6673</td>
<td>4933</td>
<td>2697</td>
<td>2250</td>
<td>3262</td>
<td>2</td>
<td>7.80 (5.32)</td>
</tr>
<tr>
<td>13</td>
<td>70</td>
<td>10</td>
<td>10</td>
<td>700</td>
<td>3254</td>
<td>1813</td>
<td>1674</td>
<td>1344</td>
<td>1192</td>
<td>1892</td>
<td>13</td>
<td>14.65 (13.6)</td>
</tr>
<tr>
<td>14</td>
<td>80</td>
<td>6</td>
<td>6</td>
<td>480</td>
<td>9890</td>
<td>6608</td>
<td>6078</td>
<td>5882</td>
<td>3358</td>
<td>4433</td>
<td>50</td>
<td>10.25 (12.5)</td>
</tr>
<tr>
<td>15</td>
<td>80</td>
<td>8</td>
<td>8</td>
<td>640</td>
<td>11243</td>
<td>6980</td>
<td>7132</td>
<td>7132</td>
<td>5000</td>
<td>4200</td>
<td>50</td>
<td>15.58 (11.5)</td>
</tr>
<tr>
<td>16</td>
<td>80</td>
<td>10</td>
<td>10</td>
<td>800</td>
<td>13144</td>
<td>7982</td>
<td>6604</td>
<td>5054</td>
<td>3771</td>
<td>4466</td>
<td>50</td>
<td>20.93 (17.7)</td>
</tr>
<tr>
<td>17</td>
<td>90</td>
<td>6</td>
<td>6</td>
<td>540</td>
<td>11522</td>
<td>7005</td>
<td>7045</td>
<td>6004</td>
<td>4426</td>
<td>4716</td>
<td>50</td>
<td>14.09 (18.4)</td>
</tr>
<tr>
<td>18</td>
<td>90</td>
<td>8</td>
<td>8</td>
<td>720</td>
<td>13254</td>
<td>8138</td>
<td>8159</td>
<td>8280</td>
<td>3262</td>
<td>3466</td>
<td>50</td>
<td>20.36 (15.5)</td>
</tr>
<tr>
<td>19</td>
<td>90</td>
<td>10</td>
<td>10</td>
<td>900</td>
<td>15129</td>
<td>9194</td>
<td>9194</td>
<td>9194</td>
<td>6078</td>
<td>4293</td>
<td>50</td>
<td>28.59 (25.1)</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>6</td>
<td>6</td>
<td>600</td>
<td>13261</td>
<td>8031</td>
<td>8031</td>
<td>8031</td>
<td>4976</td>
<td>3012</td>
<td>50</td>
<td>18.35 (23.9)</td>
</tr>
<tr>
<td>21</td>
<td>100</td>
<td>8</td>
<td>8</td>
<td>800</td>
<td>15091</td>
<td>9332</td>
<td>9332</td>
<td>9332</td>
<td>6020</td>
<td>3892</td>
<td>50</td>
<td>26.61 (22.1)</td>
</tr>
<tr>
<td>22</td>
<td>100</td>
<td>10</td>
<td>10</td>
<td>1000</td>
<td>17204</td>
<td>10503</td>
<td>10503</td>
<td>10503</td>
<td>6880</td>
<td>4810</td>
<td>50</td>
<td>41.99 (33.7)</td>
</tr>
</tbody>
</table>
Figure 2. Average length of the W, Wp, HIS, UIOv, H, UIO and DS Test Suites.

Figure 2 depicts the average length of test suites and sorted according to Column V (number of transitions), derived using the testing methods. The UIO method generates shorter test suites than all other methods. However, on average, most of the UIO test suites are incomplete (more analysis on the completeness of the UIO test suites is given below). The DS and H methods generate test suites of comparable length. However, unlike the H method, the DS method is not always applicable and the experiments show that it becomes less applicable as the ratio of the number of outputs to the number of transitions decreases (more detailed analysis is given below). The HIS and Wp test suites are of comparable length. The reason for that could be that state identifiers used by Wp method do not need to be harmonized and thus, are shorter than those used by HIS method. The HIS/Wp methods generate shorter suites than those of the W method, as expected. The UIOv did not perform better than the HIS and Wp method; that also is expected as UIOv is a particular case of the Wp-method. Moreover, differently from Wp-method only UIO sequences are used as state identifiers in the transition checking phase.

Figure 3 depicts the ratios of length of the test suites of the HIS/Wp, UIOv, H and UIO methods over the length of the W-based test suites for the (groups of) experiments depicted in rows 1 to 22 of Table 1. The HIS/Wp (UIOv, H, UIO, DS) test suites are on average 0.61 (0.65, 0.4, 0.27, 0.36) percent of those of the W method. According to these experiments these ratios, for all except the UIOv, are almost independent of the size of the specification. For the UIOv, in some cases, for large machines, when the number of inputs/outputs is small in comparison to the number of states (see for example rows 17 and 20 of Table 1), the UIOv produces test suites that are longer than those of the W method. The reason can be that the W set that contains all UIO is worse for test derivation than a distinguishability set that contains a shortest distinguishing sequence for each pair of states. On average, the ratio of the length of UIOv test suites over the length of the W test suites is 0.65.

For a given reduced FSM $M$ with $n$ states and $k$ input symbols, the worst-case length of the test suite generated using the W, Wp, HIS, H, UIOv and UIO methods is of the order $kn^2$ for a reduced completely specified FSM [4, 20]. In practice, according to the conducted experiments, the test suites derived by these methods have length of the order $cn^2$, where the constant $c$ is 5 for the W method and 4 for the Wp, HIS, UIOv and H methods. We also experimented how long are UIO sequences when those exist. According to the obtained results, the average length of UIO sequences equals to two and is independent of the size of the specification.

Columns XIII and XIV of Table 1 show how many times (out of the 50) the UIO and DS methods are applicable. On average, the UIO method is applicable to 99% of all conducted experiments and it seems that its applicability is independent of the size of the specification. On average, the DS method is applicable only to 19% of all conducted experiments and its applicability significantly decreased as the ratio of the number of outputs to the number of states of the specifications decreases. For example, on average, the DS was applicable to 65% of all conducted experiments when the ratio of the number of outputs to the number of states is more than 0.2 (rows 1, 2 and 3 of Table 1). However, this applicability drops to 14% when the ratio is between 0.1 and 0.2.

Knowing that the UIO method generates (when applicable) shorter test suites than all other methods, we conducted experiments to have an idea about the fault detection capability of these suites. Table 2
contains the details of these experiments. Each row of Table 2 corresponds to a group of 50 randomly generated completely specified reduced specifications. For each of these specifications, we derive a test suite using the UIO method (when applicable) and explicitly derive all possible faulty implementations of the specification. Then, we determine how many of these implementations are killed (detected) by the derived test suite. We consider in our experiments small size specifications with two to six states ($n = 2, 3, 4, 5, 6$), two inputs ($k = 2$) and outputs, since explicit enumeration is only possible for small size implementations.

As shown in Table 2, on average, when applicable, 41% of the UIO test suites are incomplete. Moreover, when the number of states compared with the number of outputs increases the possibility having incomplete test suites increases.

### Table 2. Fault Coverage of UIO Test Suites

<table>
<thead>
<tr>
<th>I. No. of States, n</th>
<th>II. No. of Inputs, k</th>
<th>III. No. of Outputs</th>
<th>IV. No. of Experiments (out of 50) where UIO is Applicable</th>
<th>V. No. of Complete Test Suites</th>
<th>VI. % of Complete UIO Test Suites</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>50</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>50</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>38</td>
<td>27</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>43</td>
<td>11</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>59</strong></td>
</tr>
</tbody>
</table>
test suites are significantly longer than those derived by all other methods. The length of the Wp and HIS test suites almost coincide and are larger than those of the suites derived by DS, UIO, and H methods. Test suites derived by the UIO method are the shortest and the UIO method is applicable to 99% of the conducted experiments. However, when applicable, experiments with small size specifications show that 41% of the UIO test suites are incomplete. Test suites derived by the DS and H methods are comparable and significantly shorter than those derived by the W, Wp, HIS, and UIOv methods. However, the DS method is applicable only to 19% of all conducted experiments and its applicability significantly decreases as the ratio of the number of outputs to the number of transitions decreases. The order of all derived test suites is $O(cn^2)$ which is lower than the theoretical worst-case order $O(kn^3)$, where $n$ is the number of states of the specification/implementation machines, $k$ is the number of inputs, and $c$ is a constant less than or equal to 5. The experiments are conducted on randomly generated specifications and on a realistic protocol called the Simple Connection Protocol.

6. References


### Appendix: The SCP Protocol Messages

![Diagram of SCP Protocol Messages](image)

CONreq(qos) with qos∈[0,3], connect(ReqQos) with ReqQos∈[0,3], accept(qos) with qos∈[0,3], CONcnf(+,FinQos) with FinQos∈[0,3] and data(FinQos) with FinQos∈[0,3].