Research Article

A Combining Method for Wireless Protocol Conformance Testing: A Empirical Case

Lin Wei-Wei ^{1,2} **Zeng Hong-Wei**¹ and **Jung Yoon Kim** ³

¹School of Computer Engineering and Science, Shanghai University, Shanghai, China

²Shanghai Key Laboratory of Computer Software Evaluating & Testing, Shanghai, China

³Graduate School of Game, Gachon University, Seongnam, Republic of Korea

Correspondence should be addressed to Lin Wei-Wei; newstart@shu.edu.cn and Jung Yoon Kim; kjyoon@gachon.ac.kr

Received 29 May 2021; Revised 6 June 2021; Accepted 18 June 2021; Published 15 July 2021

Academic Editor: Xiaoxian Yang

Copyright © 2021 Lin Wei-Wei et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Ensuring the consistency of protocol implementation and protocol specification is the basic premise of wireless communication. With the application of wireless communication in more and more fields, the wireless communication environment becomes more and more complex, and the fault coverage of wireless protocol conformance testing is also facing more and more challenges. To solve this problem, this paper uses Finite State Machine (FSM) as a formal description tool for wireless protocols and presents a combining test method based on two test methods with complementary characteristics in the test technologies based on structural coverage and state identification. Then, the paper evaluates the effectiveness of the method based on 14 empirical cases. The experimental results show that the fault coverage of each empirical case can be effectively improved to 100% when the average test cost is only increased by 17.99%.

1. Introduction

Wireless protocol is the rule that must be obeyed by the communication entities and processes in wireless communication [1]. It is also the basic guarantee to realize the stability of wireless communication and the competitive advantage in the low-power wireless communication market [2, 3]. Conformance testing, performance testing, interoperability testing, and robustness testing are all essential to verify the correctness of wireless protocol implementation [4]. Conformance testing is the basis of the other three kinds of testing, which is used to ensure the consistency of protocol implementation and protocol specification [1, 5, 6]. Finite State Machine (FSM) [7–10] is an effective formal modeling tool [11, 12] in wireless protocol conformance testing since the operations implemented on the entities in wireless protocols (such as button click or window input in GUI and message request in Web service) can be mapped to the inputs in FSMs, and the responses to the operations according to the wireless protocols (such as message response in GUI or web service) can be mapped to the outputs in FSMs [13–16]. To

facilitate the description, we abstract these events into alphabetic or numeric symbols in the examples and experiments of this paper. ZigBee is an emerging short-range, low-rate, and low-power wireless network technology. Taking the connection and release process of MAC layer in ZigBee protocol as an example [17], the interaction process between communication entities is shown in Figure 1(a) and the corresponding FSM is shown in Figure 1(b). The relationships between specific events in ZigBee protocol and abstract symbol definitions in FSM are described in Tables 1, 2, and 3, respectively. In this way, the problem of judging whether the implementation of ZigBee protocol is consistent with the ZigBee protocol specification can be transformed into the problem of judging whether the implementation of ZigBee protocol is consistent with the specification FSM of ZigBee protocol. Specifically, a test set is generated based on the specification FSM and the test set is applied to the wireless protocol implementation. When the actual output is consistent with the expected output, the implementation is consistent with the protocol. Otherwise, it means that there is a fault in the implementation.

FIGURE 1: Connection and release process in ZigBee.

TABLE 1: Abstract definition of state.

State signal	State
s_1	Unconnected state (initial state)
s_2	Received connection request data frame and waiting for data request command frame
s_3	Agree to the connection request, and wait for the ACK
s_4	Connection status

TABLE 2: Abstract definition of input event.

Input signal	Input event
i_1	Connection request frame
i_2	Data request frame
i_3	Normal network status and network resources are not exhausted
i_4	Malicious connection request frames, exhausting network resources
i_5	ACK
i_6	Release request frame
i_7	No input data, waiting timeout

TABLE 3: Abstract definition of output event.

Output signal	Output event
o_1	ACK
o_2	Response data frame of allowing nodes to join the network
o_3	Response data frame of refusing nodes to join the network

less protocol conformance testing, this paper presents a combined method based on the complementary characteristics of the TPC method and UIO method. And the combined method is studied by 14 established empirical cases including ZigBee protocol and the WM2RP protocol both of which are widely used in wireless communication. The contribution of this paper is mainly as follows:

- (1) The fault coverage of wireless protocol conformance testing is effectively improved
- (2) Since the experimental objects are all FSM empirical cases designed and generated by testers according to professional experience and domain knowledge, the experimental data and results have more practical significance, which can also promote the application of FSM conformance testing in industry

Both the transition-pair coverage (TPC) method [18–20] based on structure coverage and the unique input output (UIO) method [7, 13] based on state identification can be used in wireless protocol conformance testing. However, neither TPC method nor UIO method can achieve 100% fault coverage. To meet the high reliability requirements of wire-

The remainder of the paper is organized as follows. Section 2 introduces the concepts and terms about FSM. Section 3 presents a combined method for wireless protocol conformance testing. Section 4 describes the case study and evaluates the experimental results. Section 5 overviews the

FIGURE 2: FSM M_1 .

related work. Finally, Section 6 summarizes the full text and prospects of the future work.

2. Preliminaries

An FSM is formally defined as a 6-tuple $M = (I, O, S, s_0, \delta, \lambda)$ [21, 22] where

- (i) I and O are finite and nonempty sets of input symbols and output symbols, respectively
- (ii) S is a finite and nonempty set of states
- (iii) $s_0 \in S$ represents the initial state of M
- (iv) $\delta : S \times I \longrightarrow S$ denotes the state transition function
- (v) $\lambda : S \times I \longrightarrow O$ is the output function

According to this definition, when an input symbol i is delivered to the current state s , M moves to state $s' = \delta(s, i)$ with an output produced by $\lambda(s, i)$. A transition t is defined by a tuple $(s, s', i/o)$ where s is the start state of t , i is an input of s , $o = \lambda(s, i)$ is the associated output, and $s' = \delta(s, i)$ is the end state of t .

An FSM M is initially connected if for any state s there is at least one input sequence from s_0 to s , i.e., $\forall s \in S, \exists \alpha \in I^* \cdot (\delta(s_0, \alpha) = s)$ (where I^* is the set of finite sequences of input symbols).

If $s \neq s', \alpha \in I^* \cdot (\lambda(s, \alpha) \neq \lambda(s', \alpha))$, then s and s' are a pair of distinguishable states, and α is called a separating sequence of s and s' . If all the states in FSM M are pairwise distinguishable, i.e., $\forall s, s' \in S, s \neq s', \exists \alpha \in I^* \cdot (\lambda(s, \alpha) \neq \lambda(s', \alpha))$, then M is said to be minimal or reduced.

An FSM M is deterministic if at any state s and for any input i , there is at most one transition leading to the next state. Otherwise, M is nondeterministic.

An FSM is not initially connected means there are some states that cannot be reached from the initial state. In other words, the states that cannot be reached from the initial state are meaningless to the FSM. Therefore, an FSM that is not initially connected can be converted to an initially connected FSM by deleting the states that cannot be reached from the initial state. An FSM with equivalent states can be transformed into a reduced FSM by joining equivalent states. A nondeterministic FSM can be converted to a deterministic FSM by combining states, i.e., raising the level of abstraction.

Since all the above definitions can be satisfied, only initially connected, reduced and deterministic FSMs are considered in this paper. In Figure 2, M_1 is an initially connected, reduced, and deterministic FSM where $S = \{s_1, s_2, s_3\}$, $I = \{x, y\}$, $O = \{0, 1\}$ and s_1 highlighted in bold defaults to the initial state.

The connection of input sequences in FSM is represented by the juxtaposition of input sequences; that is, if $\alpha_1, \alpha_2 \in I^*$, then $\alpha_1\alpha_2$ represents the connection of α_1 and α_2 . If $\beta = \alpha_1\alpha_2$, then α_1 is the prefix of β , expressed as $\alpha_1 \leq \beta$. If $\alpha_1 \neq \epsilon$ (ϵ is an empty string), then α_1 is the true prefix of β , expressed as $\alpha_1 < \beta$. Given a set of input sequences T , $\text{pref}(T)$ represents the set of all prefixes of all sequences in T ; that is, $\text{pref}(T) = \{\alpha \mid \exists \beta \in T \text{ and } \alpha \leq \beta\}$; if $T = \text{pref}(T)$ then, T is a set of prefix closures.

In this work, we assume that FSM M has a reset operation r which can bring M to its initial state from any state without any output. A string $x_1x_2, x_k \in I^*$ is said to be a defined input sequence at $s \in S$ if there exists $s_1, \dots, s_k, s_{k+1}$, where $s_1 = s$, such that $(s_i, x_i) \in (S \times I)$ and $\delta(s_i, x_i) = s_{i+1}$ for all $i = 1, \dots, k$. We use $\Omega_M(s)$ to denote the set of all defined input sequences at state s and Ω_M as an abbreviation for $\Omega_M(s_0)$, i.e., the input sequences defined at the initial state of M , and hence for M itself. A test case of M is an input sequence $\alpha \in \Omega_M$ starting with r . A test suite of M is a finite set of test cases, and there are not two test cases α and β such that $\alpha < \beta$. The number of symbols of a sequence α is represented by $|\alpha|$. For instance, given a test case $ryyy$ for M_1 , the length of $ryyy$ is 4, since it contains four symbols, that is, $|ryyy| = 4$. This notation is extended to a test suite T ; i.e., $|T|$ means the sum of all the length of sequences in T . For instance, given a test suite $T = \{ryx, ryyy, ryyx, rx\}$ for M_1 , the length of T is 13, i.e., $|T| = |\{ryx, ryyy, ryyx, rx\}| = 13$.

An initialization fault occurs when FSM starts from an incorrect initial state $s_i (s_i \neq s_0)$. An output fault occurs when the correct output of a transition $(s, s', i/o)$ is changed, and a transition $(s, s', i/o')$ in which the output is different from the expected one is generated. An end state fault occurs when the end state of a transition $(s, s', i/o)$ is changed, and a transition $(s, s'', i/o)$ in which the end state is different from the expected one is generated. Figure 3 demonstrates the three types of faults based on M_1 . There is an initialization fault starting from an incorrect initial state s_2 in Figure 3(a). There is an output fault in Figure 3(b). The transition $(s_3, s_1, y/0)$ is changed to the transition $(s_3, s_1, y/1)$ with a wrong output. There is an end state fault in Figure 3(c). The transition $(s_1, s_1, x/0)$ is changed to the transition $(s_1, s_2, x/0)$ with a wrong end state.

3. TPC-UIO Test Method

3.1. Motivation Example. Figures 4 and 5 are two examples to present the complementarity between the TPC method and UIO method.

The FSM in Figure 4(a) is a wireless protocol specification, and the FSM in Figure 4(b) is an implementation, where the dotted line represents a transition with an end state fault. $\{rcbbdb, rcbbecda, rcbbecbbac\}$ is the test set of TPC method

FIGURE 3: Three types of faults in M_1 .

FIGURE 4: Example 1 of relationship between the TPC method and UIO method.

FIGURE 5: Example 2 of relationship between the TPC method and UIO method.

generated based on Figure 4(a), and $\{rcbba, rcbdb, rcbecb-bacb, rcbecda\}$ is the test set of UIO method generated based on Figure 4(a).

The FSM in Figure 5(a) is a wireless protocol specification, and the FSM in Figure 5(b) is an implementation, where the dotted line also represents a transition with an end state fault. $\{rba, rbb, raaa, raab, rabaa, rabab, rabba, rabbb\}$ is the test set of the TPC method generated based on Figure 5(a), and $\{raaa, rbaa, rabba, rabaaa\}$ is the test set of the UIO method generated based on Figure 5(a).

Test results indicate the following:

- (1) The test set of the TPC method can test the fault in Figure 5(b) but cannot test the fault in Figure 4(b)
- (2) The test set of the UIO method can test the fault in Figure 4(b) but cannot test the fault in Figure 5(b)

It can be concluded from the above examples that there are obvious differences in the performance of TPC method

and UIO method in different examples. Specifically, the two methods show strong complementarity. When the number of transitions outgoing from the correct end state and the wrong end state is equal, furthermore, the input/output of the transitions outgoing from the correct end state and the wrong end state are all the same, fault coverage ability of UIO method is superior to TPC method. When there are differences about the number of transitions outgoing from the correct end state and the wrong end state or there are differences about the input/output of transitions outgoing from the correct end state and the wrong end state and the UIO sequence of the correct end state is also available in the wrong end state, fault coverage ability of the TPC method is superior to that of the UIO method.

3.2. Test Generation of TPC-UIO Method. Based on the above analysis, we introduce a combined TPC-UIO test method to realize the complementarity of the TPC method and UIO method.

Input: FSM M with n states $\{s_1, s_2, \dots, s_n\}$ and s_1 is the initial state, transition tree T and $\text{Root}(T) = s_1$
Output: State covering set $\text{SC}(M)$

```

1: Initialize  $\text{SC}(M) = \emptyset$ 
2: for each transition starting with  $s_1$ 
3: Add a branch in  $T$  and mark the branch with the input/output of the transition
4: end for
5: do
6: for each leaf node appears for the first time in  $T$ 
7: Add a branch in  $T$  for each transition starting with the leaf node and mark the branch with the input/output of the transition
8: end for
9: until leaf node appears for the first time cannot be found in  $T$ 
10: for ( $i=1; i \leq n; i++$ )
11: Get an input/output sequence  $\beta_i$  from  $s_1$  to  $s_i$  that appears for the first time in  $T$ 
12:  $\text{SC}(M) = \text{SC}(M) \cup \beta_i$ 
13: end for

```

ALGORITHM 1: State covering set generation

FIGURE 6: Transition tree of M_1 .

3.2.1. State Covering Set Generation

Definition 1 (State covering set). Given an FSM M and an input sequence set $Q = \{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$, if for any state s_i , $\exists \alpha_i \in Q$ ($0 \leq i \leq n-1$) satisfies $\delta(s_0, \alpha_i) = s_i$, where s_0 is the initial state, then Q is a state covering set of M .

There must be a state covering set in initially connected FSM, and an empty sequence ϵ is included to reach the initial state s_0 . In addition, only state covering set of prefix closure is considered in this paper. As described in Algorithm 1, given an FSM with n states $\{s_1, s_2, \dots, s_n\}$ in which s_1 is the initial state and a transition tree T with only a root node s_1 , state covering set $\text{SC}(M)$ is first initialized to $\emptyset$. Then, for each transition starting with s_1 , a branch annotated with the input/output of the transition is added in T . Next, for each transition starting with a leaf node appears for the first time in T , a branch annotated with the input/output of the transition is added in T iteratively until there is no leaf node appearing for the first time that can be found. So far, the complete transition tree is constructed. Finally, for each state s_i ($1 \leq i \leq n$), an input/output sequence β_i from s_1 to s_i that appears for the first time in T is obtained and β_i is thrown to $\text{SC}(M)$.

According to Algorithm 1, the transition tree of M_1 is shown in Figure 6, and $\{\epsilon, y/1, (y/1)(y/1)\}$ is a state covering set obtained from the transition tree.

3.2.2. TPC Test Set Generation

Definition 2 (adjacent transition). If the end state of a transition $t_1 = (s_1, s, i_1/o_1)$ is the start state of a transition $t_2 = (s, s_2, i_2/o_2)$, t_1 and t_2 are a pair of adjacent transition.

The test set of TPC method covers each pair of adjacent transitions in FSM at least once. As shown in Algorithm 2, given an FSM with n states $\{s_1, s_2, \dots, s_n\}$ in which s_1 is the initial state and a state covering set $\text{SC}(M) = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$, TPC test set $\text{TPC}(M)$ is first initialized to $\emptyset$. Then, for each state s_i ($1 \leq i \leq n$), each transition sequence β of length 2 starting from s_i is obtained and $\alpha_i\beta$ is thrown to $\text{TPC}(M)$.

Consider M_1 in Figure 2, $\{rxx, rxy, ryy, ryx, ryxx, ryxy, ryyx, ryyy, ryyxx, ry, ryyxy, ryyyx, ryyyy\}$ is the test set of M_1 under the TPC method.

3.2.3. UIO Generation

Definition 3 (UIO sequence). An input/output sequence β is a UIO sequence of state s_i if and only if for any state $s_j \in S$ ($s_j \neq s_i$), $\lambda(s_i, \beta_{\text{in}}) \neq \lambda(s_j, \beta_{\text{in}})$, where β_{in} is the input part of β .

Definition 4 (uncertainty of the start states). Given an FSM M and an input sequence α , a partition $\pi(\alpha)$ of the state set S can be obtained according to the output generated by each state for α . For any state s_i and s_j , they will be in the same partition block if and only if they cannot be distinguished by α , i.e., $\lambda(s_i, \alpha) = \lambda(s_j, \alpha)$, and this partition is called the uncertainty of the start states.

Definition 5 (uncertainty of the current states). The current states of M after delivering α can be expressed as $\delta(B, \alpha) \mid B$

Input: FSM M with n states $\{s_1, s_2, \dots, s_n\}$ and s_1 is the initial state, state covering set $SC(M) = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$
Output: TPC test set $TPC(M)$
 1: Initialize $TPC(M) = \emptyset$
 2: **for** $(i=1; i \leq n; i++)$
 3: **for** each transition sequence β of length 2 starting from s_i
 4: $TPC(M) = TPC(M) \cup \alpha_i \beta$
 5: **end for**
 6: **end for**

ALGORITHM 2: TPC test set generation

FIGURE 7: Successor tree of M_1 .

$\in \pi(\alpha)$, and this partition is called the uncertainty of the current states.

In this way, the scope of the state before and after the input α can be learned according to the output corresponding to α . The idea of state partition can be applied to the construction of successor tree.

Definition 6 (successor tree). The successor tree of an FSM is a tree showing the behavior of the FSM at any state and upon any input sequence.

For each possible input sequence, there is a path starting from the root node in the successor tree and each node in the successor tree is annotated with the uncertainty of start states as well as the uncertainty of current states.

The successor tree of M_1 is shown in Figure 7. When x is delivered to the set of states $\{s_1, s_2, s_3\}$, according to the different output, the partition blocks of start state uncertainty are $0\{s_1, s_3\}$ and $1\{s_2\}$, respectively, where 0 and 1 represent the output generated by the state in the partition block. The corresponding partition blocks of current state uncertainty are $\{s_1, s_3\}$ and $\{s_2\}$, respectively. When x is again delivered to the current state uncertainty $\{s_1, s_3\}$ and $\{s_2\}$ according to the different output, the partition blocks of start state uncertainty are $00\{s_1, s_3\}$ and $11\{s_2\}$, respectively. The corresponding partition blocks of current state uncertainty are $\{s_1, s_3\}$ and $\{s_2\}$. Since the set $\{s_1, s_3\}$ can never be further divided by x , this branch can terminate the decomposition about input x . Next, y is delivered to the partition blocks of the current state uncertainty $\{s_1, s_3\}$ and $\{s_2\}$; according to the differ-

ent output, the partition blocks of start state uncertainty are $01\{s_1\}$, $00\{s_3\}$, and $11\{s_2\}$, respectively. The corresponding current state uncertainty partition blocks are $\{s_2\}$, $\{s_1\}$, and $\{s_3\}$, respectively. Since there is only one element in each partition block, the division of this branch ends here. In the same way, we can construct other branches in the successor tree until a successor tree meeting the requirement is generated.

In this paper, UIO sequence is obtained by successor tree. Given an FSM and its successor tree, if there is only one state s_i in a partition block of the start state uncertainty, we will get a UIO sequence of s_i . Here, we only consider the shortest UIO sequence; that is, the UIO sequence we first obtained in the top-down search of the successor tree. As shown in Figure 7, the UIO sequence of s_1 is $(x/0)(y/1)$, $(y/1)(x/1)$, or $(y/1)(y/1)$, the UIO sequence of s_2 is $x/1$, and the UIO sequence of s_3 is $y/0$.

3.2.4. Test Set Generation of TPC-UIO Method

Definition 7 (TPC-UIO method). The test set of the TPC-UIO method should satisfy two conditions. (1) Cover each pair of adjacent transition in an FSM at least once. (2) Identify the end state of each pair of adjacent transition using the shortest UIO sequence.

As shown in Algorithm 3, given TPC test set $TPC(M)$ of an FSM and UIO sequence of each state, for each test case in $TPC(M)$, identify the end state of the test case by the corresponding UIO sequence. Then, the test cases which are true prefixes of others are deleted and TPC-UIO test set $TPC\text{-}UIO(M)$ is constructed.

Taking M_1 for example, $\{rxxxxy, rxyx, ryxxx, ryxyy, ryyyxy, ryyxyxy, ryyyxyx, ryyyyx\}$ is the final test set of the TPC-UIO method.

4. Case Study

4.1. Experimental Subject. In existing literatures on FSM conformance testing, most of researchers conduct experiment using randomly generated FSMs. Generate FSMs randomly can obtain a large amount of FSMs in a short time, however, whether the experimental conclusions obtained from randomly generated FSMs are completely applicable on FSM empirical cases are still uncertain. This paper selects 14 established FSM empirical cases which are designed and generated by testers with professional experience and domain knowledge [17, 22–29]. The information of all FSM empirical cases

Input: TPC test set TPC(M), UIO sequence of each state
Output: TPC-UIO test set TPC-UIO(M)
 1: **for** each test case in TPC(M)
 2: Identify the end state of the test case by the corresponding UIO sequence
 3: **end for**
 4: delete the test cases which are true prefixes of others

ALGORITHM 3: Test set generation of TPC-UIO method

TABLE 4: Experimental subject.

FSM	S	I	T
Connection and release process of MAC layer in ZigBee protocol	4	7	7
Base station model in WM2RP protocol	3	3	3
Intermediate node model in WM2RP protocol	4	4	4
Leaf node model in WM2RP protocol	3	3	3
Railway signal system communication protocol	9	15	20
Conference protocol	5	11	28
INRES communication protocol	4	5	9
Transfer Protocol	4	10	18
ISDN BRI network layer protocol	8	13	25
NBS TP4 protocol	8	12	26
Alternating Bit Protocol	8	5	10
RPL host protocol	5	6	10
RPL router protocol	8	14	29
LLN Border Router Protocol	4	5	6

is shown in Table 4. $|S|$, $|I|$, and $|T|$ denote the number of states, the number of inputs, and the number of transitions, respectively.

4.2. Mutation Generation. In this paper, we use mutations of the specification to model faults in the implementation. Mutation operators [30] are introduced to obtain the three most typical faults in FSM conformance testing.

Definition 8 (change initial state (CIS)). This operator represents the initialization fault: change initial state s_0 to s_k ($s_k \neq s_0$) such that the implementation starts from an incorrect initial state, that is, $CIS(M) = \{N = (I, O, S, s_k, \delta, \lambda) \mid s_k \in S \cdot s_k \neq s_0\}$.

Definition 9 (change output (CO)). This operator represents the output fault such that an unexpected output is produced in an implementation. Formally, $CO(M) = \{N = (I, O, S, s_0, \delta, \lambda) \mid \exists x \in I, s_i \in S \cdot \lambda(s_i, x) \neq \delta(s_i, x)\}$.

Definition 10 (change end state (CES)). This operator represents the end state fault such that the implementation moves to an incorrect state. Formally, $CES(M) = \{N = (I, O, S, s_0, \Delta, \lambda) \mid \exists x \in I, s_i \in S \cdot \Delta(s_i, x) \neq \delta(s_i, x)\}$.

4.3. Experimental Setup. In order to improve the credibility of the experimental results, experimental replication is adopted in this paper, and the number of replications in each round is

not the same. For every FSM empirical case, each type of mutation is randomly operated 10 times, 100 times, 1000 times, 10000 times, 20000 times, 30000 times, 40000 times, 50000 times, 60000 times, 70000 times, 80000 times, 90000 times, and 100000 times; that is, a total of three groups including 551110 mutations in each group are obtained.

4.4. Result Analysis

4.4.1. Comparison of Test Cost. The test set information of all FSM empirical cases is shown in Table 5. L represents the length of the test set and N represents the number of the test cases in the test set, that is, the number of reset operations.

- (i) Only consider the length of the test set. For all FSM empirical cases, when considering the average length of the test sets of each test method, the average length of the UIO method is 59.64, the average length of the TPC method is 186.57, and the average length of the TPC-UIO method is 227.93. Figure 8 indicates that the length of the test set for TPC-UIO method is larger than that for the TPC method and UIO method for each empirical case. Compared with that of the TPC method, the average increase of the length of the TPC-UIO method is 24.32%
- (ii) Only consider the number of the test cases. For all FSM empirical cases, when considering the average number of the test cases of each test method, the

TABLE 5: Collection of test set information.

FSM	UIO		TPC		TPC-UIO	
	<i>L</i>	<i>N</i>	<i>L</i>	<i>N</i>	<i>L</i>	<i>N</i>
Connection and release process of MAC layer in ZigBee protocol	19	4	19	4	23	4
Base station model in WM2RP protocol	5	1	5	1	6	1
Intermediate node model in WM2RP protocol	6	1	6	1	7	1
Leaf node model in WM2RP protocol	5	1	5	1	6	1
Railway signal system communication protocol	66	12	260	48	308	48
Conference protocol	135	25	762	138	922	142
INRES communication protocol	32	6	72	13	85	13
Transfer Protocol	70	16	204	45	249	45
ISDN BRI network layer protocol	98	18	223	41	264	41
NBS TP4 protocol	153	22	218	40	318	45
Alternating Bit Protocol	36	5	24	3	40	5
RPL host protocol	37	6	95	15	110	15
RPL router protocol	156	22	691	96	820	98
LLN Border Router Protocol	17	3	28	5	33	5

FIGURE 8: Relationship among the length of test sets for TPC, UIO, and TPC-UIO.

average number of the test cases for the UIO method is 9.93, the average number of the test cases for the TCP method is 32.07, and the average number of the test cases for the TPC-UIO method is 33.14. Figure 9 indicates that the number of the test cases

for TPC-UIO method is more than or equal to that for the TPC method and UIO method for each empirical case. Compared with the TPC method, the average increase of the number of the test cases of the TPC-UIO method is 6.43%

FIGURE 9: Relationship among the number of test cases for TPC, UIO, and TPC-UIO.

- (iii) Consider both the length of the test set and the number of the test cases. In order to quantitatively analyze the cost considering both the length of the test set and the number of the test cases, L is used to denote the length of the test set and the cost generated by the length of the test sets is expressed as a function $C_1 = f(L)$, and N is used to denote the number of the test cases and the cost generated by the number of the test cases is expressed as a function $C_2 = g(N)$. The total test cost considering both the length of the test set and the number of the test cases can be expressed as $C = C_1 + C_2$. Assuming that the test cost of a test set containing only one test case of length 1 is 1, then $f(1) + g(1) = 1$ ($f(1) < 1, g(1) < 1$). Assuming $\alpha = f(1)$, then $g(1) = 1 - \alpha$, $f(L) = \alpha \times L$, $g(N) = (1 - \alpha) \times N$, and $C = \alpha \times \sum_{i=1}^N L_i + (1 - \alpha) \times N$ where L_i represents the length of the i th test case, so that the cost of each test method can be quantified by assigning a value to α . Generally, the impact of the length of the test set is greater than the impact of the number of the test cases, so it is usually $\alpha > 0.5$. Figure 10 shows the relationship among the comprehensive test costs of each empirical case under different test methods when α is assigned 0.6, 0.7, 0.8, and 0.9, respectively. From the examples of different values of α , it can be seen that when α takes different values greater than 0.5, the comprehensive test cost of each

empirical case under different test methods changes, and the larger the value of α , the greater the test cost. However, the relative relationship among test costs of different empirical cases has not changed. The comprehensive test cost of the TPC-UIO method is always larger than that of the TPC method and UIO method for each empirical case. Compared with the TPC method, the average increase of the comprehensive test cost of the TPC-UIO method is 23.22%.

4.4.2. Comparison of Fault Coverage. As shown in Table 6, the TPC method, UIO method, and TPC-UIO method can achieve 100% fault coverage for initialization fault and output fault. Neither the TPC method nor the UIO method can reach 100% fault coverage for end state fault. The TPC-UIO method can effectively improve the fault coverage rate and achieve 100% fault coverage.

Although the differences of data in the experiment seem to be subtle, these differences are still worthy of attention. (1) When the fault domain is very large, even a small increase of fault coverage represents a large number of faults. (2) In a safety-critical environment, there is a tiny optimization even if an optimization of only one fault has high practical value. It is noted that the end state faults in the experiment are generated randomly. In fact, the higher the proportion of the analyzed end state faults in Section 4, the better the fault coverage of the combined method.

(a) $\alpha = 0.6$ (b) $\alpha = 0.7$

FIGURE 10: Continued.

(c) $\alpha = 0.8$ (d) $\alpha = 0.9$

FIGURE 10: Relationship among TPC, UIO, and TPC-UIO considering both the length of test sets and the number of test cases.

TABLE 6: Fault coverage ranking of TPC, UIO, and TPC-UIO.

Ranking	Initialization fault	Output fault	End state fault
1	TPC (100%) UIO (100%) TPC-UIO (100%)	TPC (100%) UIO (100%) TPC-UIO (100%)	TPC-UIO (100%)
2			UIO (99.8%)
3			TPC (99.06%)

5. Related Work

Conformance testing based on FSM has always been an active research direction in software testing [31–39]. In the existing literature, there are precedents for improving fault coverage in conformance testing by combining different testing methods. Mouchawrab et al. [40] investigated the complementarity of the test method of round trip path based on UML state machine and the test method of edge coverage based on source code and then extended round-trip paths testing with edge coverage testing. Results show that the two test strategies are significantly more effective when combined by augmenting state machine testing with structural testing. Hutchins et al. [41] combined the all-use criterion in a white-box test and the edge coverage criterion in a black-box test to improve fault coverage with higher test cost. Results show that significant improvements in the effectiveness of coverage-based tests usually occurred as coverage increased from 90% to 100% and the complementarity of the two methods in their effectiveness was proved. Briand et al. [42] extended the round trip path criterion in the white-box test with the classification partition in the black-box test and obtained better fault detection performance.

This paper also analyzes the complementary characteristic of different test methods and improves the fault coverage at the expense of an acceptable increase of test cost, so as to meet the high requirements of fault coverage in wireless communications.

6. Conclusion and Future Work

There are some differences about the test requirements of the TPC method and UIO method in wireless protocol conformance testing. The TPC method focuses on covering all consecutive operations of length two between entities or processes in wireless communication. UIO method focuses on identifying each state of entities or processes and confirming the input and output information between any two states in wireless communication. In order to improve the effectiveness of wireless protocol testing, a combined method is presented and the method is evaluated by 14 established empirical cases. A unified method based on state covering set is used to generate test sets for each test method and three types of classic faults are generated by mutating the specification. Fault coverage has been effectively improved by the combining method. The experimental results show that the fault coverage of each empirical case can be improved to 100% when the average test cost is only increased by 17.99%. Since what we considered is the specifications modeled by FSMs and the faults generated based on FSMs, although the fault coverage has been effectively improved,

these results cannot be popularized to the specifications modeled by other formal models or faults generated based on code.

In a future work, we will try to enrich the experimental data with more publicly available FSM empirical cases. To better describe the behavior characteristics of wireless protocol testing, the future work will also consider extended semantic FSMs to improve the expression ability of the model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

None of the authors have any conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 61572306.

References

- [1] W. W. Lin, H. W. Zeng, H. H. Gao, H. K. Miao, and X. L. Wang, "Test sequence reduction of wireless protocol conformance testing to Internet of Things," *Security and Communication Networks*, vol. 2018, Article ID 3723691, 13 pages, 2018.
- [2] H. H. Gao, W. Q. Huang, and X. X. Yang, "Applying probabilistic model checking to path planning in an intelligent transportation system using mobility trajectories and their statistical data," *Intelligent Automation and Soft Computing*, vol. 25, no. 3, pp. 547–559, 2019.
- [3] X. L. Wang, H. W. Zeng, H. H. Gao, H. K. Miao, and W. W. Lin, "Location-based test case prioritization for software embedded in mobile devices using the law of gravitation," *Mobile Information Systems*, vol. 2019, Article ID 9083956, 14 pages, 2019.
- [4] H. H. Gao, W. Q. Huang, X. X. Yang, Y. C. Duan, and Y. Y. Yin, "Toward service selection for workflow reconfiguration: an interface-based computing solution," *Future Generation Computer Systems*, vol. 87, pp. 298–311, 2018.
- [5] H. L. Yang, K. Ma, C. Deng, H. S. Liao, J. Yan, and J. Zhang, "Towards conformance testing of choreography based on scenario2013 International Symposium on Theoretical Aspects of Software Engineering," in pp. 59–62, Birmingham, UK, 2013.
- [6] M. S. Ramada, T. W. De Lima, A. Simao, and A. D. Soares, "Generating reduced tests for FSMs using a search-based testing approach," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 400–407, Portland, OR, USA, November 4–6, 2019.

- [7] P. Liu, H. K. Miao, H. W. Zeng, and Y. Liu, "FSM-based testing: theory, method and evaluation," *Chinese Journal of Computers*, vol. 34, no. 6, pp. 965–984, 2011.
- [8] A. Jaffari, C. Yoo, and J. Lee, "Automatic test data generation using the activity diagram and search-based technique," *Applied Sciences*, vol. 10, no. 10, p. 3397, 2020.
- [9] G. Z. Lu and H. K. Miao, "Optimized test cases generation for EFSM model combining abstraction refinement and satisfiability," *Chinese Journal of Computers*, vol. 39, no. 11, pp. 2236–2252, 2016.
- [10] A. Saeed, S. H. Ab Hamid, and A. A. Sani, "Cost and effectiveness of search-based techniques for model-based testing: an empirical analysis," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 4, pp. 601–622, 2017.
- [11] H. H. Gao, S. Y. Mao, W. Q. Huang, and X. X. Yang, "Applying probabilistic model checking to financial production risk evaluation and control: a case study of Alibaba's Yu'e Bao," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 3, pp. 785–795, 2018.
- [12] H. H. Gao, H. K. Miao, L. L. Liu, J. Y. Kai, and K. Zhao, "Automated quantitative verification for service-based system design: a visualization transform tool perspective," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 10, pp. 1369–1397, 2018.
- [13] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines—a survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090–1123, 1996.
- [14] F. R. Porto, A. T. Endo, and A. Simao, "Generation of checking sequences using identification sets," in *In 15th International Conference on Formal Engineering Methods (ICFEM)*, pp. 115–130, Berlin, Heidelberg, October 29–November 1, 2013.
- [15] K. Hoda and L. Yvan, "On FSM-based testing: An empirical study: complete round-trip versus transition trees," in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 305–315, Toulouse, France, October 23–26, 2017.
- [16] M. C. Gaudel, "Formal methods for software testing," in *In International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pp. 1–3, Sophia Antipolis, France, September 13–15, 2017.
- [17] S. R. Liang, *Fuzzy Test Algorithm of ZigBee Protocol Based on FSM [M.S. Thesis]*, BUPT, Beijing, China, 2014.
- [18] A. Simão, A. Petrenko, and J. C. Maldonado, "Experimental evaluation of coverage criteria for FSM-based testing," in *In the 21st Simpósio Brasileiro de Engenharia de Software (SBES)*, pp. 359–374, Joao Pessoa, Brazil, January 13–15, 2007.
- [19] P. Ammann and J. Offutt, *Introduction to Software Testing*, China Machine Press, Beijing, 2018.
- [20] J. Lee, S. Kang, and P. Jung, "Test coverage criteria for software product-line testing: systematic literature review," *Information and Software Technology*, vol. 122, no. 2020, article 106272, 2020.
- [21] K. El-Fakih, N. Yevtushenko, and G. Bochmann, "FSM-based incremental conformance testing methods," *IEEE Transactions on Software Engineering*, vol. 30, no. 7, pp. 425–436, 2004.
- [22] J. F. Cutigi, A. Simao, and S. R. Souza, "Reducing FSM-based test suites with guaranteed fault coverage," *The Computer Journal*, vol. 59, no. 8, pp. 1129–1143, 2016.
- [23] C. C. Liu and H. L. Yang, "Testing of data gathering protocol for wireless sensor networks based on FSM model," *Software Guide*, vol. 16, no. 9, pp. 14–18, 2017.
- [24] J. D. Lee, J. I. Jung, J. H. Lee, J. G. Hwang, J. H. Hwang, and S. U. Kim, "Verification and conformance test generation of communication protocol for railway signaling systems," *Computer Standards & Interfaces*, vol. 29, no. 2, pp. 143–151, 2007.
- [25] B. Sarikaya and G. V. Bochmann, "Synchronization and specification issues in protocol testing," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 389–395, 1984.
- [26] Z. B. Wang, H. Zhou, and B. H. Zhao, "Path overlapped method for protocol conformance test generation," *Computer Systems & Applications*, vol. 20, no. 7, pp. 47–52, 2011.
- [27] R. E. Miller and S. Paul, "On the generation of minimal-length conformance tests for communication protocols," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 116–129, 1993.
- [28] K. Sabnani and A. Dahbura, "A new technique for generating protocol test," *ACM SIGCOMM Computer Communication Review*, vol. 15, no. 4, pp. 36–43, 1985.
- [29] J. Tang, X. Huang, J. Qian, and C. Viho, "A FSM-based test sequence generation method for RPL conformance testing," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pp. 591–597, Beijing, China, August 20–23, 2013.
- [30] X. Y. Dang, X. J. Yao, D. W. Gong, and T. Tian, "Efficiently generating test data to kill stubborn mutants by dynamically reducing the search domain," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 334–348, 2020.
- [31] V. H. Fragal, A. Simao, A. T. Endo, and M. R. Mousavi, "Reducing the concretization effort in FSM-based testing of software product lines," in *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 329–336, Tokyo, Japan, March 13–17, 2017.
- [32] P. Liu, Y. D. Li, and Z. J. Li, "Some thoughts on model-based test optimization," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 268–274, Sofia, Bulgaria, July 22–26, 2019.
- [33] S. Pradhan, M. Ray, and S. K. Swain, "Transition coverage based test case generation from state chart diagram," *Journal of King Saud University-Computer and Information Sciences*, vol. 5, no. 5, pp. 1–10, 2019.
- [34] T. L. Zhou, H. Y. Sun, J. Liu, X. H. Chen, and D. H. Du, "Improving testing coverage for safety-critical system by mutated specification," in *2014 21st Asia-Pacific Software Engineering Conference*, pp. 43–46, Jeju, South Korea, 2014.
- [35] A. Denise, M. C. Gaudel, S. D. Gouraud, R. Lassaigne, J. Oudinet, and S. Peyronnet, "Coverage-biased random exploration of large models and application to testing," *International Journal on Software Tools for Technology Transfer*, vol. 14, no. 1, pp. 73–93, 2012.
- [36] R. M. Hierons and H. Ural, "Optimizing the length of checking sequences," *IEEE Transactions on Computers*, vol. 55, no. 5, pp. 618–629, 2006.
- [37] H. Ural and F. Zhang, "Reducing the lengths of checking sequences by overlapping," in *In International Conference on Testing of Communicating Systems*, pp. 274–288, New York, NY, USA, May 16–18, 2006.
- [38] L. H. Duan and J. Chen, "Reducing test sequence length using invertible sequences," in *In 9th International Conference on*

- Formal Engineering Methods (ICFEM)*, pp. 171–190, Boca Raton, FL, USA, November 14–15, 2007.
- [39] G. V. Jourdan, H. Ural, and H. Yenigün, “Reduced checking sequences using unreliable reset,” *Information Processing Letters*, vol. 115, no. 5, pp. 532–535, 2015.
 - [40] S. Mouchawrab, L. C. Briand, Y. Labiche, and M. Di Penta, “Assessing, comparing, and combining state machine-based testing and structural testing: a series of experiments,” *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 161–187, 2011.
 - [41] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand, “Experiments on the effectiveness of data-flow and control-flow-based test adequacy criteria,” in *In Proceedings of 16th International conference on Software engineering (ICSE)*, pp. 191–200, Sorrento, Italy, 1994.
 - [42] L. C. Briand, M. Di Penta, and Y. Labiche, “Assessing and improving state-based class testing: a series of experiments,” *IEEE Transactions on Software Engineering*, vol. 30, no. 11, pp. 770–783, 2004.