20

[+]A Conformance Testing Framework for Applying Test Purposes

Jae Hong Park[*] Jai Yong Lee[**] Il Young Jung[***] and Jin Pyo Hong[***]
[*]Computer Science Dept., POSTECH
San 31 Hyoja Dong, Pohang, Kyungbuk, KOREA, 790-764
[**]Electronics Eng. Dept., YON-SEI University
134 Shin-Cheon Dong, SeoDaeMoon Ku, Seoul, KOREA,120-749
[***]Protocol Engineering Center, ETRI
E-mail : jhpark@einstein.postech.ac.kr, jyl@bubble.yonsei.ac.kr
Phone : +82-562-279-2243
FAX : +82-562-279-2299

Abstract

Since the generating test cases from the specification is a difficult work, formal methods of generating test cases are being discussed in the Formal Methods in Conformance Testing(FMCT) groups. One of the real problems in generating test cases automatically is that the problem size is too big to be applied to the actual protocol. In this paper, we suggest a framework for applying test purposes in various stages of formal specification represesentation, which eventually reduces the problem size of a protocol. For this framework, we have presented three stages : extraction, formalization, and application. A GID-based method is applied to this framework.

Keyword Codes : C.2.2, D.2.5, F.4.3
Keywords : Network Protocols, Testing and Debugging, Formal Languages

1. Introduction

Generating test suites in testing the conformance of an IUT(Implementation Under Test) with respect to reference specification is the most time-consuming process in the whole process of conformance testing. Many efforts to reduce the cost of generating test suites have been done. FMCT(Formal Methods in Conformance Testing) group, which was organized in 1989 within ISO SC21/WG1, is one of the groups which performs such efforts by formalizing the process of conformance testing based on Conformance Testing Methodology and Framework of IS 9646[1]. For this purpose, FMCT group has translated informal concepts described in Conformance Testing Methodology and Framework into formal ones and has defined new concepts. For example, using several relations and symbols, it formally defines the meaning of 'conformance', 'conformance relation', and etc. Though many parts are considered in formal aspects, test purpose, which takes an important role in reducing the problem space of generating test suites, has not been much considered in the process of FMCT.

Test purpose can be briefly defined as " the intention of tester or what the tester are interested in". Based on this definition, we can conclude that each test case has its own corresponding test purpose explicitly or implicitly. This relation is represented in the document, "TSS & TP(Test Suite Structure and Test Purpose)", which describes the test suite in tree-shaped form. Its basic principle is that

- The higher level(i.e. nearer to the root) it is in tree, the more global objective it has.
- The objective of anyone is some parts of its ancestor's objective.
- Each test case in leaf has its corresponding 'primitive test objective', which is test purpose.

In general, it is impossible to generate a test suite without considering test purpose, which is equal to consider all test purposes. There are several reasons for that, that is,

- It is impossible to consider all the behaviors of a protocol(including data variation)

[+]This research was financially supported by ETRI PEC(Protocol Engineering Center) of KOREA

T. Mizuno et al. (eds.), *Protocol Test Systems*

- Even though we can predict all behaviors of a protocol, we may finally face practical problems of insufficient of memory and too long processing time.

Therefore, we should apply test purposes to the course of test case generation process to reduce the problem size. The advantages that we can get by applying test purposes may be classified as follows :

- Because we can consider only small parts of a protocol, it is possible to perfectly test that parts.
- If we can identifiy the characteristics of a target protocol(i.e what the core behavior of the protocol is or where the errors are easily occur), we can perform tests more efficiently by giving weights to those parts using test purpose.

There have been several research works referring test purposes. It is considered that [2] and [3] are the first works to exploit the area of applying test purposes to automatic generation of test cases. They suggested the possibility of appilying test purposes with their own formalization method in generating test cases but did not show a general framework for applying test purposes. In this paper, we show and analyze the framework for applying test purposes in the FMCT procedures and apply the suggested framework to our test case generation process, which is GID(Global Instantaneous Description) based method.

The rest of this paper is organized as follows : in section 2, we introduce the framework of applying test purposes and its related works. Sections 3 explains GID(Global Instaneous Description)-based method, which is to apply test purpose formally to the Control Flow Graph (whose node is GID). For the validation of our method, several examples are given in section 4. Finally, conclusions follow in section 5.

2. A Framework for applying Test Purposes

The definition of test purpose is described in [1] and [4].

In [1] : Test purpose is a prose description of a narrowly defined objective of testing, focusing on a single conformance requirement as specified in the appropriate OSI international standard or CCITT recommendation verifying the concept of specific parameter.

In [4] : The set of test purpose is the subset of conformance requirement.

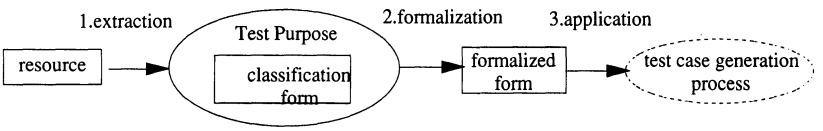

Figure 2-1 : Test purposes applying process

In order to formally apply the defined test purposes in generating test cases, we have to consider three steps as shown in Figure 2-1: extraction, formalization and application.

2.1. Extraction

In order to apply test purposes to the generation of test suites, as a first step, we have to define the resources from which we can extract test purposes. The resource can be viewed in several aspects. For example, in [1] the resources for extracting test purposes are related specification, PDU encoding rules and state tables. In [4], the resource is a set of conformance requirements and [6] views the resources as functions, timers, parameters, states and transitions. From the above viewpoints, we can generally define that resource is the relevant protocol specification or its equivalent form(conformance requirements may be the representatively equivalent form of protocol specification).

The extracted test purposes should be classified to represent their relationships after extracting test purposes from the resources. Though [5] and [6] show different viewpoints on this classification form, the relationship among test purposes is generally represented with tree-shaped structure. This is because test purposes are tightly related with Test Suite Structure(TSS). [7] is an example for this structure.

2.2. Formalization

(1) General concepts

After defining the resource from which we can extract test purposes and its classification form, the next

step for applying test purposes to the generation of test cases is how to formally represent test purposes. Formal representation is needed for combining test purpose with the relevant protocol specification(or its equivalent form), because informal test purpose(written in natural language) itself is difficult to be algorithmically implemented. Figure 2-2 shows the combining process between test purpose and relevant protocol specification.

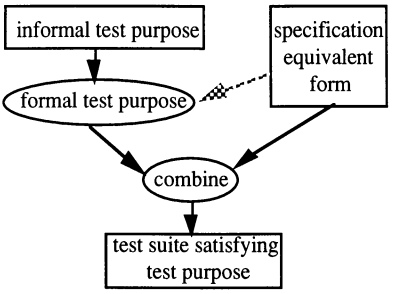

Figure 2-2. Combining process

In Figure 2-2, the step from 'informal test purpose' to 'formal test purpose' may be difficult to be implemented automatically. Therefore, human expert should intervene in this step. This step can be done manually or semi-automatically, depending on the kind of test purpose. For example, the mandatory test purposes related with 'behavior testing'[7] such as connection set-up, connection release and data transfer can be semi-automatically formalized because they are independent of protocols. Meanwhile, test purposes mainly concerned with 'capability testing'[7] are difficult to formailze automatically because they are dependent on protocols.

The combining step in Figure 2-2 is the most important part in applying test purpose. In order to implement this step automatically, we should translate test purpose into the formal one as previously mentioned. Once the formal test purpose is given, it is easy to combine test purpose with protocol specification algorithmically.

(2) Combining procedures : combined form and timing

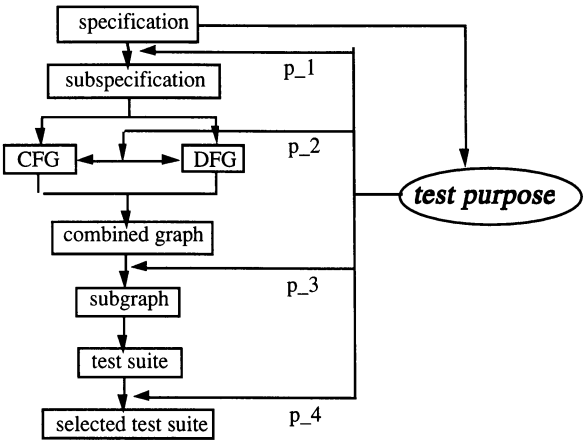

Figure 2-3. Combining points of test purposes and formal spec.

Now, let's consider the combining procedure of Figure 2-2. Note that, in this combining procedure, the formal representation may be dependent on the specification equivalent form, which is represented as a dotted arrow in Figure 2-2. Let's take some examples to show this combining procedure in detail.

As a first example, let's take test case when the test purpose is related with the structure of a protocol[8]. In this case, test purpose should be combined with the protocol specification to be applied in generating test cases. As a second example, test purpose should be applied to the DFG(Data Flow Graph), if test purpose is to test the validity of variables. From these examples, we can conclude that the formalized object to be applied to the generation test suites should be a form of a combined object of formal test purpose object and formal protocol specification. Moreover, we can conclude that the form of combined object, which can be directly used for generating test suites, can be obtained according to the combining time of formal test purpose and formal protocol specification. Figure 2-3 shows four combining points p_1, p_2, p_3 and p_4 according to the combining time of formal test purpose objects and formal protocol specification.

The characteristics, pros(+) and cons(-) for each points are summarize in the followings :

p_1. At this point, we can generate the subspecification satisfying test purposes directly from the specification. [3] shows this case, because it does directly apply test purpose to the protocol specification written in prolog. The cons and pros of this point;
(+) It is effective because the problem size can be reduced in the beginning of the process.
(+) When the information related with the structure of protocol is selected as a test purpose, it can not be applied to except at this point.[8]
(-) It is very difficult to select the part satisfying test purpose in protocol specification because specification is too abstract.

p_2. When the specification is represented with CFG(Control Flow Graph) and DFG(Data Flow Graph), it is generally known that generating DFG considering all data is almost impossile. If test purpose can be combined with the formal specification at this point, we need not generate DFG for all data but only generate for the data related with test purposes.
(+) This method is easy to use and may be used practically. Therefore, if we use it for the test purpose related with data, it would be effective.
(-) This method can be applied only to test purpose related with data.

p_3. At this point, we can generate subgraph from the combined graph (combined graph is the combination of CFG and DFG). [2] shows this case.
(+) Most existing test case generation method such as UIO, D and W methods assume graphical form of protocol specification. This method can be generally applied.

p_4. This method is the simplest and have been used widely. [10] is the case of this method, in which test case generation is considered with determining value assigning functions.
(+) It is suitable for small protocol.
(-) For large protocols, it may be impossible to generate all test suites. Because this method can be used after generating test suite, it is useless when test suite can not be generated.
(-) Since test purpose is related with conformance requirement which is required in the beginning of test suite generating process and test suite can be obtained at the end of test suite generating process, it is difficult to find the relationship between these two objects.

As we see in the above, each combining points for test purpose and formal specification representation has its own pros and cons. From the above observation, we can generally conclude that the extracted test purposes should be classified and then applicable points should be searched according to the purpose of test and the representation of formal specification. Note that the problem size of generating test cases can be reduced more as the combining point is earlier.

2.3. Application

So far, we have shown how to extract test purpose by defining the resource and when to combine the extracted test purpose with the various stages of formal protocol specification representation.

In this application process, now we can apply test case generation algorithms such as UIO[12], D[13] and GUIO[14] in the reduced problem space to generate test cases.

3. GID-based method

In section 2, we have shown a general framework for applying test purposes in various stages of formal specification to generate test cases automatically. In this section, we show how to apply the suggested framework for FMCT when the formal specification is based on GID(Global Instantaneous Description). Here, GID is a node of CFG for the specification written in Estelle.(We can generate the CFG of any protocol written in Estelle using the tool PECT(Postech Estelle to Cfg Translator)[16]). Note that the test purpose should be formally represented as GID form since the formal specification has the form of GID. So, this method is called 'GID-based method'.

3.1. GID-based CFG[11]

GID represents the whole status of a protocol at any instance. For that reason, it has many informations, such as modules, edges, queues, and etc. GID-based CFG is composed of a set of nodes and a set of edges, where the node is GID and the edge connects two GIDs.

3.2. Formalization

In this method, the combining point(application point) for test purpose is p_3 in Figure 2-3. Since the formal specification is a form of CFG, the combined object in this method is also a form of CFG(whose size is reduced) and test purpose should be formalized with the form being compatible to CFG. For this purpose, we develop GID table sequence(GID sequence, in short) as a compatible form to CFG. Definition 1 shows its definition.

Definition 1 : GID sequence(or GID table sequence)

Assuming that GID(i) is the i-th node in a graph and its information is composed of several fields(Figure 3-2). GID sequence is represented as follows :

GID sequence = $GID(0)GID(i_1)GID(i_2) \ldots GID(i_j)$

, where i_k is the index of a GID table and GID(0) is the initial GID table.

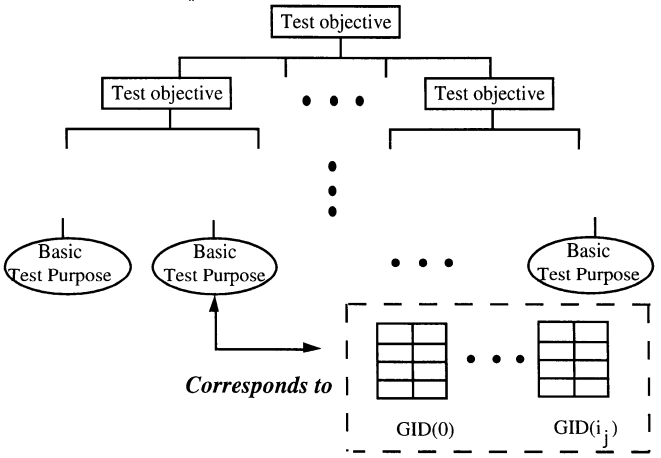

Figure 3-1. Relation between test purpose and GID sequence

When the node of CFG is GID, test purpose represented by GID sequence can be directly used for selecting subgraph from the graph. Figure 3-1 supports this explanation. As seen in Figure 3-1, the leaf(basic) node corresponds to the GID sequence of a test purpose on the assumption that test purpose has tree-structure. In general, test purpose is the sequence of nodes and GID is equivalent to node, so it is reasonable that test purpose corresponds to the GID sequence. Figure 3-2 shows the simplified version of GID table, which consists of four fields; module, event, link and state. By the definition of GID, all test purpose can be represented by GID sequence because full version of GID table may have all the possible information. But, in most cases, simplified

version will be sufficient. For example, [3] assumed that test purpose was represented by only event sequence. This kind of test purpose can also be represented by GID sequence using fields, state and event in Figure 3-2. The detailed usage will be explained in the next section.

module	
event	
link	
state	

Figure 3-2. Structure of a GID-table

3.3. Application

As will be shown in the example of section 4, we can consider that test purpose plays a role of extractor. That is, it extracts subgraph from the given CFG. Note that, the resultant object may not be a graph. But, it is allowed to consider it to be a graph because that the ID(IDentification) sequence, like UIO sequence[12], Distinguishing sequence[13], GUIO sequence[14] and etc, translates it into a graph. Genarally, in generating test sequence, we should consider the sequence for transition error check and the one for transfer error check. ID sequence is for the latter case. The translated graph is described in definition 4.

Definition 2: element_of_set

If sequence $A = a_1.a_2...a_n$, element_of_set(A) = $(a_1, a_2, a_3, .. a_n)$.

Definition 3: end

If edge 'e' is from state 'S ' to state 'T', end(e) = T.

Definition 4: translated graph

P(N, E) is translated into G(N∪Nc, E∪Ec) ,

Ec = {element_of_set(ID(n)) for all n∈N}

Nc = {end(e) for all e∈Ec}

where, P(N, E) is the result of combining test purpose and CFG

and G(N∪Nc, E∪Ec) is the graph.

When generating test sequence for translated graph G(N∪Nc, E∪Ec), we should generate test sequence checking transition error only for E and transfer error only for N. Strictly speaking, neither Ec nor Nc are our interest, so it is not needed to test them.

Based on this explanation, application algorithm is given in Algorithm 1. Two terms should be defined to explain this algorithm.

Definition 5: start

If edge 'e' is from state 'S ' to state 'T', start(e) = S.

Definition 6: LD(i)

LD(i) is the transfer sequence from initial state to state i.

Algorithm1. Application algorithm

Input : CFG and GID sequence(i.e. formalized test purpose)

output : test sequence satisfying test purpose

step 1. By comparing GID sequence and CFG, mark the node N and edge E satisfying test purpose

step 2. test_sequence(0) = LD(0).reset /* transfer error check for initial state */

```
i = 1;
for (e∈ E) { /* transition error and transfer error check */
   test_sequence[i] = LD(start(e)).e.ID(end(e)).reset;
   test_sequence = test_sequence.test_sequence[i];
   i++;
}
```

4. Example

In this section, the effectiveness of GID-based method in applying the framework for FMCT is shown by taking several practical examples.

4.1. Basic example

In [3], five exmaple TPs are introduced. They are relatively simple but important TPs, so we also select one of them as an example. In order to compare GID-based method with the one of [3], we present representation form in both method of [3] and GID-based method.

"Test IUT's ability to accept a connection"

This is to test that when a Connection Request(CR) PDU arrives at the IUT, whether the IUT replies by sending a Connection Confirm(CC) PDU. The formal representation of [3] is as follows :

[tpe(incoming, cr, -), tpe(outgoing, cc, -)]

This can be represented by GID-based method as shown in Figure 4-1.

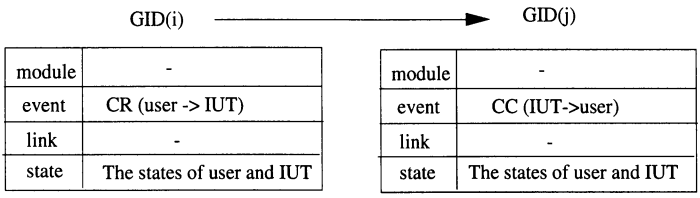

Figure 4-1. GID sequence for Example 1

4.2. Inres service[15]

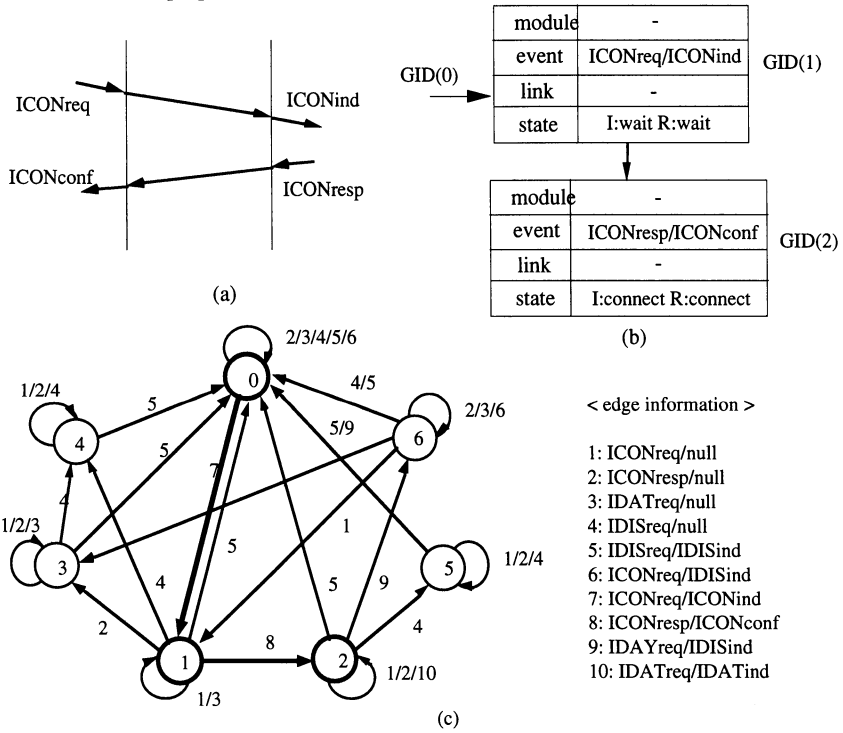

Figure 4-2. GID sequence for Example 2

Inres protocol has been approved to be a common testing protocol by the FMCT(Formal Methods in Conformance Testing) group of ISO/SC21/WG1 for comparison, evaluation and coordination of every works related with conformance testing. Though it is not a real protocol, it presents many OSI concepts and mechanisms, such as connection, disconnection, sequence numbers, acknowledgements and retransmissions, etc. It is in fact simplified version of the Abracadabra protocol.

For the CFG of Inres service, the result of applying a simple example TP is as follows :

Example TP : Test IUT's ability of succesful connection establishment, that is, after applying 'ICONreq', confirm whether 'ICONconf' is received or not.

Figure 4-2.(a) shows PDU-exchange diagram and Figure 4-2.(b) shows its GID sequence. And Figure 4-2.(c) shows the results. Using algorithm 1, we can derive test sequence satisfying test purpose,

test_sequence = reset.test_sequence[0].test_sequence[1].test_sequence[2]

where,

test_sequence[0] = ID(0).reset = GUIO(0).reset = 7.path(0).(1)'.reset

test_sequence[1] = LD(0).7.ID(1).reset = 7.UIO(1).reset = 7.2.reset

test_sequence[2] = LD(1).8.ID(2).reset = 7.8.UIO(2).reset = 7.8.4.9.reset

In this sequence, the notation and ID sequence of GUIO[14]is used. And we do not yet try to reduce the length of test sequence using some optimality algorithms[17].

5. Conclusions and Further study

In this paper, we have suggested a framework for applying test purposes in FMCT method to generate test cases automatically and showed a GID-based method to which the suggested framework is applied. Since problem size of a actual protocol is large, test purpose should be applied to the right stages of various formal protocol specification representation. The suggested framework will be eventually helpful in reducing the problem size of generating test cases.

Representing test purposes according to the formal specification, extracting methodology for test purpose from various resources, and extended work of GID-based method remains as further study.

Reference

1. ISO "Open Systems Interconnection - Conformance Testing Methodology and Framework", IS 9646.
2. Tomas Robeles, "Specification and derivation of OSI conformance test suite", IWPTS '92
3. S.J.Ashford,"Automatic Test Case Generation Using Prolog", Jan. 12, 1993. NPL technical report.
4. Formal Methods in Conformance Testing, Nov 1992.
5. Sarikaya, "A Test design methodology for protocol testing", IEEE tr. on. SE-13 No.5 May 1987.
6. S.T.Chanson, Sijian Zhang and Qin Li, " A Test case management system", IWPTS '92.
7. PTT-NL, " CMIP Test Suite Structure and Test Purpose",June '93.
8. G.J.Tretmans, "A formal approach to Conformance testing", Ph.d. thesis, 1993.
9. Hassan Ural, "Test sequence selection based on static data flow analysis", Computer Communications 10(5) 1987.
10. Ed Brinksma, "A Framwork for Test selection", PSTV IX(1991).
11. D.Y.Lee and J.Y.Lee, "A well-defined Estelle specification for the automatic test generation" , IEEE tr.on Computers, April 1991.
12. K.Sabnani and A.Dahabura, "A new technique for generating protocol tests", in proc 9th Data comm. Symp, pp 173~187, Sep 1985.
13. G.Gonenc, "A method for the design of fault detection experiments," IEEE tr.on. Computer, vol C-19, pp 551~558, June 1970.
14. J.H.Park and J.Y.Lee, "A Test Sequence Generation method with Minimum Restrictions", JWCC '93.
15. OSI formal specification case study : the Inres protocol and service.
16. POSTECH technical report: "Postech Estelle to Cfg Translator", 1993.
17. R.E.Miller and Sanjoy Paul., "On the generation of Minimal-Length Conformance Tests for Communication Protocols", IEEE/ACM tr. on Networking, vol 1, No 1, Feb. 1993.