Basing on Model Checking Techniques to Generate Testcases

Nguyen Tam Thi Minh (0710023)

School of Information School,
Japan Advanced Institute of Science and Technology

February 5, 2009

Keywords: Spin, Promela, Testcase, W-method, Conformance.

1 Introduction

In software development, besides following processes in right order such as requirement analysis, design, implementation, and so on; developers are also interested in guaranteeing the quality of those processes. Although designers accomplished their jobs well, it is not guaranteed that the implementation process is right, i.e. following designed specifications. In practice, testing using Testcases is one of the effective methods for this problem.

In this research, we will propose an algorithm to automatically generate Testcases from design model for testing the conformity of implementation with design model. So far we verified the correctness of a design model by model checking tool Spin. To conform to Spin[2], the design model is described in Promela language. From the model in Promela (Promela description), we will propose a method to automatically generate Testcase.

2 Related Work

In related research, we will present W-method, a method of conformance testing, and two approaches which automatically generate Testcases from formal specifications, e.g. model checking and Model Based Testing (MBT).

W-method is a method of testing the conformity of Implementation with its Specification based on automata theory. This method assumes that Specification and Implementation may be modeled as Finite State Machine (FSM). So, testing the conformity of Implementation with its specification means that testing the conformance of two FSMs. Furthermore, this method assumes that the FSM is 1) completely specified, 2) minimal, 3) every state is reachable in practice. These assumptions themselves limit the application of W-method on in fact.

MBT builds a model from requirement to derive Testcases for the system. The model is built by designers, and its correctness is unverified. As a result, different models may derive from the same requirement, and all are not correct. If valuation of test result is not good, it is necessary to consider rebuilding the model and Testcase generation algorithm.

Model checker generates Testcases by applying mutation analysis. The mutation analysis is a method for building the inconsistencies between model and the temporal logic constraints. When model checker processes an inconsistency, it generates a counterexample - a sequence of reachable states beginning in a valid initial state and ending with the property violation state. The counterexample is used as a Testcase. However, it is hard to build the set of counterexamples coveraging all paths of model. Thus, this method is incomplete.

3 Approach

In this research, we adatp W-method[1] to propose a frame for testing conformance between Implementation and Promela description. Promela description and Implementation are modeled as FSM. Then, testing conformance between Implementation and Promela description becomes testing conformance between two FSMs. Moreover, we propose a method which automatically generate Testcases from Promela description by using the search function in Spin. Hence, we redefine FSM based on Promela description.

To evaluate the proposal method, we carry out real experiment on Kitchen Timer[3]. The design model in Promela of Kitchen Timer is available, and its correctness has been verified by Spin.

4 Proposal Method

4.1 Frame for Conformance Testing

The frame for testing conformity of Implementation and Promela description consists of three major steps:

Step 1. model Promela description as FSM. We offer some regulations for modeling Promela description as FSM. In this research, we focus on the Promela description which has only one internal process.

Step 2. model Implementation as FSM. We assume that the Implementation is C program, and its internal state is observable. And the Impelementation is model as FSM based on the observable states. To derive unique FSM from the Implementation, we normalize the way of drawing data describing the observable states.

Step 3. test the conformance of two FSMs, D and M. The conformance of D and M means that $M = D$. In this research, we assume that $I_M = I_D$ and $S(M) = S(D)$, where I_M is the set of input symbols in M, I_D is the set of input symbols in D, $S(M)$ is the set of states in M, and $S(D)$ is the set of states in D. Basing on these assumptions, we propose a method for testing $M = D$ by adapting W-method.

4.2 Automatic Testcase Generation based on Promela Description

The process of automatically generating Testcase from Promela description consists of three steps:

1) label the branchs of search tree in Spin with a tuple (input symbol, output symbol, next state). In this step, we embed C code into Promela description to print the tuple (input symbol, output symbol, next state) at each branch.

2) compile the Promela description which contains embbeded C code. We compile this Promela description with `-DCHECK` option. This compilation prints what the verifier does at each step during the verification process in more detail. Also, it contains what the search tree traverses at each step during the verification process in more detail. In addition, the result of compilation is stored into file `dcheck.out`.

3) automatically generate Testcase from file `dcheck.out`. We build a program `TestcaseGenerator` as a Testcase generation tool in this research. `TestcaseGenerator` processes file `dcheck.out`, and generates the set of Testcases.

5 Experiment

By using the process of automatically generating Testcase from Promela description, we generated the set of Testcase in Kitchen Timer. The set contains 1155 Testcases.

After generating the set of Testcases, we implemented Kitchen Timer based on its Promela description. To have the assumption $I_M = I_D$, C program of Kitchen Timer handles only three events corresponding to three messages in Promela description. However, it is more complicated to have the assumption $S(M) = S(D)$. Therefore, we implemented Kitchen Timer on three Implementation Patterns to consider this assumption.

1) Pattern 1 ($L_{Imp} = L_{Promela}$, $V_{Imp} = V_{Promela}$) is the simplest case. In this pattern, Kitchen Timer was implemented as Promela description. Thus, it is easy to have the assumption $S(M) = S(D)$.

2) Pattern 2 ($L_{Imp} > L_{Promela}$, $V_{Imp} = V_{Promela}$) is Implementation Pattern of real Kitchen Timer. In this pattern, the strings describing redundant labels were replaced by the labels corresponding to Promela description. Then we had the assumption $S(M) = S(D)$.

3) Pattern 3 ($L_{Imp} = L_{Promela}$, $V_{Imp} > V_{Promela}$) is the most complicated case. In this pattern, one state in Promela description is implemented by several states. We only considered these states in series. To have the assumption $S(M) = S(D)$, we defined special symbols and reprensetations, and offer their correspondence relation. When a Testcase is run on the Implementation, if these symbols or representations are found, their correspondence relation is applied to have the assumption $S(M) = S(D)$.

In this research, we considered the above Implementation patterns, and offered solutions corresponding to each pattern to have the assumption $S(M) = S(D)$. However, there are many Implementation patterns in fact. In the future, we will consider other Implementation patterns.

6 Conclusion

This research adapted W-method to propose a method for automatically generating Testcases from design model, and offered a frame for testing conformance of the model and its Implementation by using generated Testcases. Because the design model is described in Promela, this method is also called a method for automatically generating Testcases from Promela description. Furthermore, we carried out the real experiment on Kitchen Timer to valuate the proposal method.

This research contributed a testcase generation method to software testing. This method removed the problem as the problem in MBT, and overcome obstacles in W-method. The method automatically generates testcase from Promela description by using search function in Spin, so that it is practicable.

References

[1] T. Chow, " Testing software design modeled by finite state machines ", *IEEE Trans. Software Eng.*, vol. SE-4, pp. 178-187, Mar, 1978.

[2] Holzmann, G. J.(2004). *The SPIN Model Checker: Primer and Reference Manual*. Boston: Addison-Wesley.

[3] **啓樹穴田** (2007, December 3). **モデルベース開発とは**. **日経エレクトロニクス**: **組み込みアカデミー**, pp. 133-140.