A Test-case Design Method based on Feature Trees

Do Thi Bich Ngoc, Takashi KITAMURA, Hitoshi OHSAKI,
Ling FANG, Shunsuke YATABE

Collaborative Facilities for Verification and Specification
National Institute of Advanced Industrial Science and Technology
do.ngoc@aist.go.jp,t.kitamura@aist.go.jp, ohsaki@ni.aist.go.jp,
fang-ling@aist.go.jp, shunsuke.yatabe@aist.go.jp

Abstract. This paper proposes a test-case design method for black-box testing. First, a language for test-case design based on "*feature tree*" is provided. Next, a formal semantics of the language is defined as for reliability of generated test-cases. Last, we propose and implement a SAT based algorithm for automated test-case generation, whose correctness is proved w.r.t. the above-mentioned semantics. Our preliminary experiments show that the method is effective for test-case design in practice in both aspects of the language design and scalability of the algorithm.

Background

In black-box testing (BBT), test-cases are designed by analyzing the input domain of the system under test (SUT) according to its specification. There are several techniques for designing test-case in BBT, such as analysis techniques of input parameter (e.g., equivalent partitioning, boundary value analysis) and combinatorial techniques (e.g., orthogonal arrays, all-pair testing). However, it lacks techniques to analyze the input domain of SUT in a top-down manner to achieve an exhaustive analysis.

We are developing a test-case design method for BBT with the three advantages: (1) providing a test-case design language to represent the input domain of the system as a feature tree, (2) generating test-cases from the feature tree automatically, (3) analyzing test-case and providing a visualization of test-cases with the tree diagram.

Test-case design method

Basically, our method provides a diagrammatic language for test-case design based on feature modeling method [1]. To design test-case, the input domain is recursively split according to different features, and then represented as an and-xor tree diagram. A set of test-cases is derived by using a combinatorial technique based on an interpretation of feature tree. For example, Fig. 1 shows a test-case design for a computer vision system that recognizes size and shape of various blocks. To design test-cases for the system, we analyze the input domain of the system, i.e., "block". "block" is first split into the two features "size" and "shape" and labeled *and*. Next, "size" is split into the two features "large" and "small" and labeled *xor*... The test-cases, e.g. ("large", "scalene"), are derived through the combination of features such that for *xor*-feature, only one sub-feature is considered.

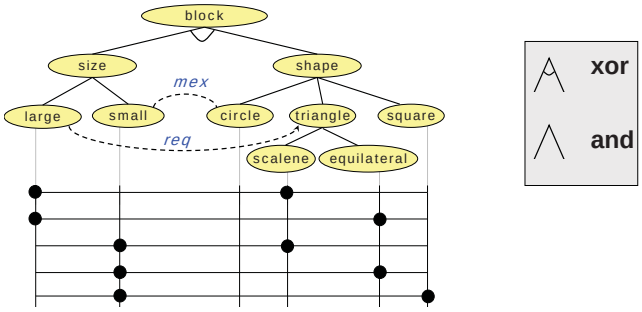

Fig. 1. Example of feature tree and test-cases

To get rid of non-sense test-cases, and-xor tree is extended by adding two kinds of cross-tree operators, i.e., *req*(require) and *mex*(mutual exclude). For example, assume that "there is no block whose size is small and shape is circle" is given as a specification of SUT. The *mex* operator drawn between "small" and "circle" in Fig. 1 expresses the situation, excluding their combinations.

We develop a formal semantics of language as a basis for the reliability of the method. The semantics is defined in terms of a set of test-cases. It defines a set of test-cases derived from a feature tree in a unique way, preventing its misinterpretation and ambiguity which often cause faulty development.

We design and implement a fully-automated test-case generation algorithm. It encodes a feature tree into propositional formulas, and a set of test-cases is generated by finding all the satisfying assignments of the formulas using a SAT solver. The advantage of the algorithm is that propositional formulas can be solved efficiently by SAT solvers recently. Besides, SAT-encoding of feature trees is straightforward regarding the developed semantics, and hence correctness (i.e., soundness and completeness) of the algorithm w.r.t. the semantics can be easily proved.

Our preliminary experiments on designing test-cases of real-time OS show that the compact language is expressive enough for the test-case design purpose in practice.

Conclusion

A test-case design method is proposed based on feature modeling method. In practical point of view, the method has several advantages. First, visualization of test-case design with tree diagrams functions as evidence media for showing the correctness of SUT. Second, a top-down analysis of input-domain with tree structure can often help us design exhaustive set of test-cases. Third, automated test-cases analysis function can reduce the cost of test-cases preparing and managing. Last, by developing a formal semantics and proving correctness of the algorithm, we guarantee the reliability which is essential to such a quality assurance method.

References

1. K. C. Kang et. al. Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon University Software Engineering Institute, 1990.