Automated Exhaustive Test Case Generation for FBD Program

Sang Hun Lee [a*], Sung Min Shin[a], Hyun Gook Kang [b], Jong-Gyun Choi[a]
[a]*Korea Atomic Energy Research Institute, 111 Daedeok-daero, 989beon-gil, Yuseong-gu, Daejeon, Republic of Korea*
[b]*Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY, USA*
[*]*Corresponding author: lees@kaeri.re.kr*

1. Introduction

For the function block diagram (FBD) programs that run on programmable logic controllers (PLCs) [1, 2], thorough testing of software is crucial for ensuring the safety of nuclear power plants (NPPs). By its nature, the NPP software is a logical matter and determines the function of hardware in the digitalized environment. The space that digitalized input and internal variables construct can be considered as the domain that the software may encounter during system operation, which may be large but not infinite. If we can perform the software testing over the whole of this space, the limitations related to the state-of-the-art test-based software reliability quantification method, such as the uncertainty in input selection and model parameter estimation [3, 4], can be resolved.

In this study, the automated exhaustive test case generation framework for NPP safety-critical software testing is proposed which formally translates the FBD program into Satisfiability Modulo Theories (SMT) formula and generates exhaustive test cases by iteratively solving the translated SMT formula using an SMT solver.

2. Methods

2.1 Satisfiability Modulo Theory

For the static analysis and program verification, Boolean satisfiability (SAT) and SMT have received considerable attention during the last decade to derive the test cases from various models automatically [5]. The SMT problem is the problem of determining whether such a formula in first-order logic is satisfiable where some function and predicate symbols have additional interpretations such as linear inequalities or function symbols.

As the test case generation is the process of identifying the software variables' states that satisfy the given test requirement, the exhaustive test case generation problem can be formulated as the problem of finding all possible combinations of software variables' states that generate the desired output, as shown in Equation (1).

$$\underset{x_i \in S}{arg}\, f(x_i) = true \quad \cdots\cdots\cdots\cdots\cdots\cdots (1)$$

In Equation (1), the formula (f) represents the FBD program that defines the logic between the software safety output ($f(x_i)$) and the software input and internal variables, or the literals (x_i). Space (S) is defined as the domain of all possible software input and internal variables' states that may encounter during software operation. For NPP safety software, the test requirement can be set to the safety signal initiation by the software (i.e., $f(x_i)=true$). In this case, the derived test cases will reflect all possible safety signal demand situations. In this study, the proposed approach utilizes Z3 [6], to identify the states of the software variables' states that generate the safety signal for FBD program.

2.2 Translation from FBD into SMT

In order to generate exhaustive test cases for FBD program, we first formally defined the FBDs based on the ideas discussed in previous studies [7, 8] and FBD-to-SMT translation rules were developed based on the formal definition of FBDs. As shown in Fig. 1, the translation starts with generating SMT formulas for all individual function blocks (FBs), and continues for component FBD and system FBD.

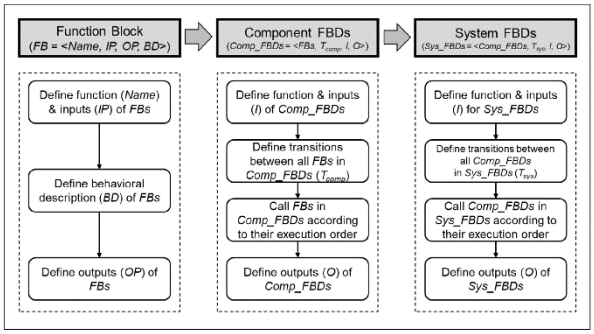

Fig. 1. FBD-to-SMT translation procedure.

2.3 Exhaustive test case generation for FBD program

Given the SMT formulas translated from FBD program, a test case can be generated by finding the states of software input and internal variables that satisfy the selected test requirement. The procedure for generating exhaustive test cases for FBD program consists of three major parts: 1) defining the software variables and FBD program under test, 2) defining the test requirements, and 3) retrieving the model for software variables that satisfies the test requirements. Fig. 2 shows a flowchart of the exhaustive test case generation procedure.

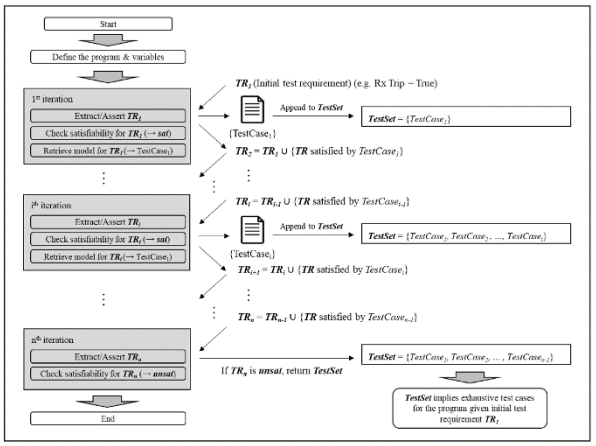

Fig. 2. Flowchart of exhaustive test case generation procedure.

In the first part, the program information required for test case generation, such as program variables and the FBD programs, are defined. Here, the names and possible ranges of all variables under test and constants as well as the SMT formula translated based on FBD-to-SMT translation rules are declared. In the second part, the initial test requirement (TR_1) is defined and added as a set of constraints of the formula to be solved. In the third part, the algorithm at least one interpretation for the defined variables that evaluate a given formula to satisfy test requirement, the SMT solver returns sat as a satisfiability check result. If it is unsatisfiable, it returns unsat. If the formula is satisfiable, the model that makes the formula true is retrieved and saved as a single test case for the FBD program. In order to derive another model, the negation of the derived model at each iteration ($TestCase_i$ at i-th iteration) is added as a new constraint to the test requirement which will be used for next iteration (TR_{i+1}). This requires the SMT solver to find another solution at the next iteration ($TestCase_{i+1}$ at $(i+1)$-th iteration) for the formula except the ones found in previous iterations. At each iteration, the derived model is added to the *TestSet*. The process is repeated until the solver returns unsatisfiable (unsat) as a satisfiability check result for the formula given the test requirement. In result, the derived *TestSet* represent the exhaustive test cases for the initial test requirement where each test case is exclusive to each other

3. Case Study

3.1 Exhaustive test case generation of target software

As a case study, the exhaustive test cases for pressurizer pressure-low reactor trip logic of KNICS IDiPS-RPS bistable processor (BP) trip logic software [9] were derived. The FBD program of BP software was loaded to the FETCG which generates the test module and the translated SMT formula, as shown in Fig. 3. The test module generates exhaustive test cases from the defined test requirement for the FBD program based on the three levels of translated SMT files (i.e., FBs, component FBDs, and system FBD). While the BP trip logic software generates various external outputs including heartbeat and diagnostic signals, the system output variable considered in this study for test case generation purpose was selected as trip signal output.

The FETCG then generates the SMT formula for the software output which the tester provided and derives the software input and internal variables. In the case study, the FETCG derived 35 variables among a total of 612 variables defined in BP software as the program input and internal variables that contribute to generating the pressurizer-pressure-low trip signal output.

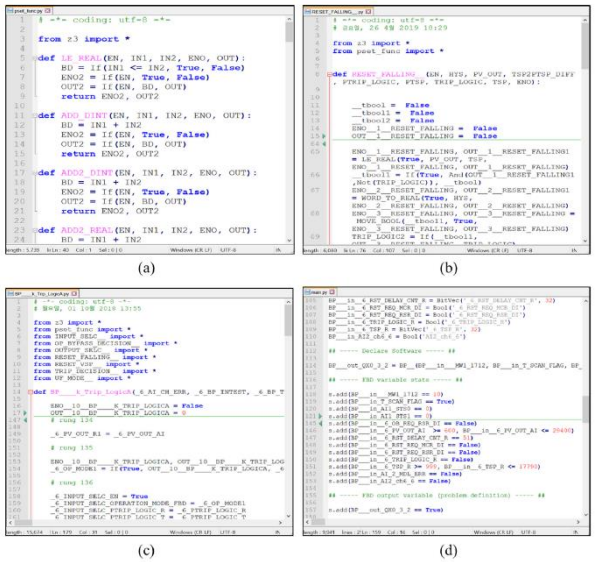

Fig. 3. Generated files of the FETCG for IDiPS-RPS BP software: (a) Function block, (b) Component FBDs, (c) System FBD, (d) Test module.

3.2 Test execution and result analysis

The generated exhaustive test cases were post-processed into the format of software test-bed to verify whether the machine code of BP software generates the trip signal given the derived test cases. Fig. 4 shows the test results for the exhaustive test cases using a simulation-based software test-bed developed in authors' previous study [10]. The expected output for all test cases is set as the value of the memory address of output variable to be true (0×1). In result, all the 147,694,036 test cases generated the trip signal, and the test was conducted in 3.45 hrs using 64 units of 3-GHz logical processors ($\sim$ 5.38 msec per test case). As all test cases generated correct output, BP software was proven to be error-free in terms of its safety function (pressurizer-pressure-low trip signal generation) for the target scenario.

Fig. 4. Test results of the case study for BP pressurizer-pressure-low trip logic: (a) Test case file, (b) Expected output file, (c) Program file, (d) Screenshot of simulation-based test-bed execution, (e) Test result file, (f) Test summary file.

4. Conclusion

This study proposes an automated exhaustive test case generation method for FBD programs. Testing the NPP software is important in demonstrating that the software generates its dedicated safety function when demand comes. The proposed framework formally translates the FBD program to SMT formula and the FBDET algorithm is developed for the exhaustive test case generation. As an application of the proposed software test method, the exhaustive test cases for the pressurizer-pressure-low trip logic of KNICS IDiPS-RPS BP software were derived and tested using the software test-bed.

Although the proposed framework focused on the exhaustive test case generation method for the FBD program used in NPP digital safety systems, the verification and the performance of the SMT solver used in the framework should be further investigated for its ability to generate a correct model and test cases.

While this study only considered the normal range of the software variables that can occur during its operation; however, the abnormal inputs that can occur due to the faults in the PLC module or memory will be also investigated in the future work.

ACKNOWLEDGEMENT

This work was supported by the Nuclear Safety Research Program through the Korea Foundation Of Nuclear Safety(KoFONS) using the financial resource granted by the Nuclear Safety and Security Commission(NSSC) of the Republic of Korea. (No. 2106027)

REFERENCES

[1] IEC, Programmable controllers - Part 3: Programming languages, International Electrotechnical Commission, Geneva, Switzerland, IEC 61131-3:2013, 2013.
[2] P. S. Acharyulu and P. Seetharamaiah, A framework for safety automation of safety-critical systems operations, Safety Science, Vol. 77, pp. 133-142, 2015.
[3] T. L. Chu et al., Development of a statistical testing approach for quantifying safety-related digital system on demand failure probability, U.S. NRC, Washington, DC, USA, NUREG/CR-7234, 2017.
[4] H. G. Kang, H. G. Lim, H. J. Lee, M. C. Kim and S. C. Jang, Input-profile-based software failure probability quantification for safety signal generation systems, Reliability Engineering System Safety, Vol. 94, No. 10, pp. 1542-1546, 2009.
[5] C. Barrett and C. Tinelli, Satisfiability modulo theories, in Handbook of Model Checking, Cham, Switzerland, Springer, pp. 305-343, 2018.
[6] L. d. Moura and N. Bjørner, Z3: An efficient SMT solver, The 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Budapest, Hungary, Mar. 29 - Apr. 6, 2008.
[7] J. Yoo, E.-S. Kim, J.-S. Lee, A behavior-preserving translation from FBD design to C implementation for reactor protection system software, Nuclear Engineering Technology, Vol 45, No. 4, pp. 489–504, 2013.
[8] D. A. Lee, J. Yoo, J. S. Lee, A systematic verification of behavioral consistency between FBD design and ansi-c implementation using HW-CBMC, Reliability Engineering System Safety, No. 120, pp. 139-149, 2013.
[9] K. C. Kwon and M. S. Lee, Technical review on the localized digital instrumentation and control systems, Nuclear Engineering Technology., Vol. 41, No. 4, pp. 447-454, 2009.
[10] S. H. Lee, S. J. Lee, J. Park, E. C. Lee, H. G. Kang. Development of simulation-based testing environment for safety-critical software. Nuclear Engineering Technology, Vol 50, No. 4, pp. 570-581, 2018.