기능 블록 다이어그램에 대한 자동 뮤턴트 생성

Lingjun Liu ○, 지은경, 배두환
한국과학기술원
{riensha, ekjee, bae}@se.kaist.ac.kr

Automated mutant generation for function block diagram programs

Lingjun Liu ○, Eunkyoung Jee, Doo-Hwan Bae
School of Computing, Korea Advanced Institute of Science and Technology (KAIST)

Abstract

Since function block diagram (FBD) programs are widely used to implement safety-critical systems, effective testing for FBD programs has become important. Mutation testing is an error-based technique. It is highly effective but computationally expensive. To support developers for FBD testing, we propose an automated mutant generator for FBD programs. We designed this tool with the cost and equivalent mutant issues in consideration. We conducted experiments on real industrial examples to present the performance of this tool. The results show that this tool can generate mutants for FBD programs automatically with low probability of equivalent mutants and low cost. This tool can effectively support mutation analysis and mutation-adequate test generation for FBD programs.

1. Introduction

The testing for Programmatic Logic Controller (PLC) programs has become an important issue since the PLCs have been used to implement safety-critical systems. As Function Block Diagram (FBD) is one of the standard PLC programming languages defined in IEC 61131-3 [1], the effective testing of FBD programs is also necessary. Mutation testing is an effective technique to measure fault detection capability of test data and also a way to achieve the high quality required in critical software [2].

We propose a tool called MuGenFBD to automatically generate mutants for FBD programs based on the defined mutation operator [3]. We consider the cost of mutation testing and equivalent mutant raising issues in our approach. This tool can considerably ease the mutation analysis and the generation of mutation adequate test suite for FBD programs.

2. Related work

For FBD testing, some existing studies evaluated effectiveness of test suites [4] and generated mutation adequate test suites [5] by mutation testing method. Shin et al. [4] conducted mutation analysis to investigate the fault detection capability of test suite and defined five mutation operators: Constant Value Replacement (CVR), Inverter Insertion or Deletion (IID), Arithmetic Block Replacement (ABR), Logic Block Replacement (LBR), and Comparison Block replacement (CBR). Eniou et al. [5] proposed mutation-based test suite generation by model checking and defined six mutation operators. The difference between these two mutation operator sets is the additional timer block replacement operator defined in Eniou et al.'s work.

*This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1I1A1A01062946) and the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2015-0-00250, (SW Star Lab) Software R&D for Model based Analysis and Verification of Higher-order Large Complex System).

Jee et al. [3] extended Shin et al.'s work and defined 13 mutation operators which includes all the mutation operators defined in the previous work. This mutation operator set covers most of defined functions and function blocks. Considering the misplacement of inputs, this mutation operator set also includes SWitched Inputs (SWI) operator.

3. Mutant generator for FBD programs

3.1. Overall process

The overall process of MuGenFBD is shown in Figure 1. MuGenFBD takes subject FBD programs in XML format and mutation operator selection as input. For each block, MuGenFBD applies the corresponding block replacement operator with number of inputs in consideration if the block replacement operator is selected; MuGenFBD applies the SWI operator with equivalent mutant issues in consideration if the

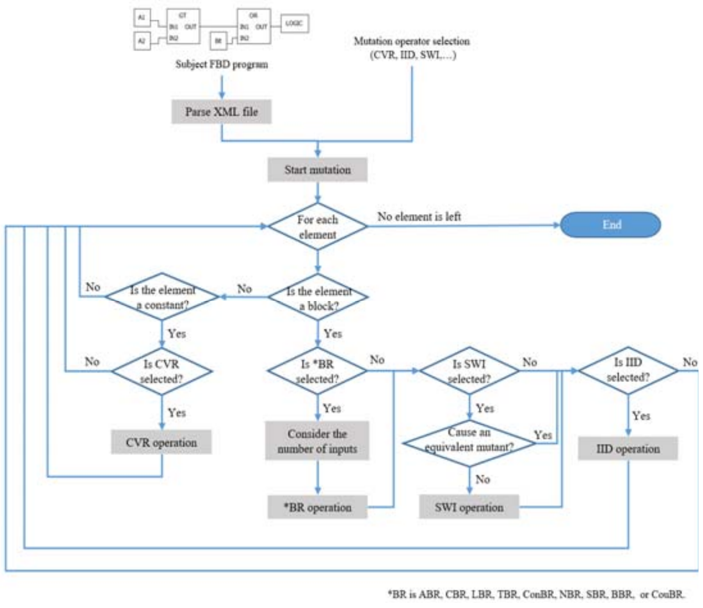

Figure 1. Overall process of MuGenFBD

Table 1. Probabilities of generating equivalent mutants

Subject	#blocks	#inputs	#outputs	#total mutants	Probability
simTRIP	3	3	2	14	0.000 (0/14)
simGRAVEL	3	3	4	8	0.000 (0/8)
LAUNCHER	4	2	1	13	0.077 (1/13)
FFTD	29	12	8	129	0.000 (0/129)
FRTD	29	12	8	129	0.000 (0/129)
VFTD	44	16	8	198	0.015 (3/198)
VRTD	44	17	8	199	0.015 (3/199)
MFTD	47	21	8	211	0.014 (3/211)
HB	19	6	1	114	0.000 (0/114)
combinedTD	437	221	92	1948	0.005 (9/1948)
Average	65.9	31.3	14	296.3	0.013

SWI operator is selected; MuGenFBD applies the IID operator if the IID operator is selected. For each constant, MuGenFBD applies the CVR operator when the operator is selected.

3.2. Issues of block replacement operators

Among all functions, there are some extensible functions. The number of inputs in extensible functions can be increased. The number of inputs is fixed in non-extensible functions. In the same group, there might be some extensible functions and some non-extensible functions, and there might be some different numbers of input or output between blocks. Thus, when we designed the block replacement mutation operators, we also considered whether the block is extensible or not and different number of inputs and outputs between blocks.

3.3. Equivalent mutant raising issues

An equivalent mutant is functionally equal to the original program. Calculating mutation score should exclude equivalent mutants. Thus, the chance for generating equivalent mutants should be limited. We found the SWI operator can possibly generate equivalent mutants. For instance, if we apply this mutation operator to the *AND* block, there's no influence on the logic (behavior). Hence, when applying the SWI operator, we carefully exclude functions that produce equivalent mutants, such as *ADD* (addition), *AND*, *OR*, etc.

4. Empirical evaluation

4.1. Subject programs

We chose our subject programs from the Korean Nuclear Instrumentation and Control System (KNICS) project's BP system [6]. The BP system includes about 20 modules that can be categorized into heartbeat (HB) monitoring modules and five types of trip decision modules: fix-falling (FFTD), fix-rising (FRTD), variable-rate-falling (VFTD), variable-rate-rising (VRTD), and manual-reset-falling (MFTD). To test scalability of the proposed approach, the combinedTD module is developed by combining several modules in the BP system. We designed three more subject programs, which are simTRIP, simGRAVEL, and LAUNCHER, to cover all function block groups. Table 1 shows the size information of each subject program.

4.2. Experiment

To demonstrate the performance of our tool, we applied our tool to subject programs. There are three aspects that we want to show the performance: (1) probability of producing equivalent mutants, (2) mutation operator selection, and (3) time efficiency.

Probability of producing equivalent mutants:

While selecting all the mutation operators, we executed MuGenFBD on all subject programs. As shown in Table 1, in half of cases, no equivalent mutants were found. In average, there is only 1.3 percent of probability of generating equivalent mutants. We utilized the SMT solver to identify equivalent mutants by finding a solution to distinguish the mutant from the original program. Without the automated mutant generation, we cannot evaluate the quality of mutation operator set.

Mutation operator selection:

MuGenFBD can support users freely select their desired mutation operators. First, we selected the original mutation operator set: CVR, IID, ABR, LBR, and CBR [4]. Second, we selected all the implemented mutation operators. In average, the extended mutation operator set generates over 44% more mutants than the original mutation operator set.

Time Efficiency:

To present the efficiency, we selected all the mutation operators and executed MuGenFBD on the large scale program called combinedTD. MuGenFBD took around three minutes to generate up to 1948 mutants. MuGenFBD is considered to provide practically usable performance.

5. Conclusion

This paper proposed an automated mutant generator called MuGenFBD for FBD programs by considering the cost and equivalent mutant issues. MuGenFBD achieved significantly low chance for producing equivalent mutants by only 1.3 percentage. For large scale program, MuGenFBD generated 1948 mutants in around three minutes. According to results, MuGenFBD can ease the mutation analysis and the automated generation of mutation-based test suites for FBD programs. In future work, we plan to develop a tool, which can automatically generate mutation-based test suite, with the help of MuGenFBD.

6. References

[1] International Electrotechnical Commission (IEC), "IEC61131-3: International Standard for Programmable Controllers - Part 3: Programming Languages," 2013

[2] M. R. Woodward, "Mutation testing—its origin and evolution," Information and Software Technology, Vol. 35, No. 3, pp.163-169, 1993

[3] E. Jee, J. Song, and D. H. Bae, "Definition and Applicatio of Mutation Operator Extensions for FBD Programs," Journal of KIISE: Transactions on Computing Practices, Vol. 24, No. 11, pp. 589-595, 2018 (in Korean)

[4] D. Shin, E. Jee E, D. H. Bae, "Empirical evaluation on FBD model-based test coverage criteria using mutation analysis," International Conference on Model Driven Engineering Languages and Systems, pp. 465-479, 2012

[5] E. P. Enoiu, D. Sundmark, A. Causevic, R. Feldt, andP. Pettersson, "Mutation-Based Test Generation for PLC Embedded Software using Model Checking," Proc. of the 28th International Conference on Testing Software and Systems, Lecture Notes in Computer Science, Vol. 9976, pp. 155-171, 2016.

[6] Doosan Heavy Industry & Construction, "Software design specification for the bistable processor of the reactor protection system," KNICS.RPS.SDS231-01, Rev.01, 2006 (In Korean)