Automated Random Simulation for Checking a Behavioral Model of Systems Based on Extended Place/Transition Net with Attributed Tokens

Sho Matsumoto
Department of Engineering and Design, Faculty of Engineering and Design, Kagawa University
2217-20 Hayashi-cho, Takamatsu-shi, Kagawa 761-0396, Japan

Tetsuro Katayama
Department of Computer Science and Systems Engineering, Faculty of Engineering, University of Miyazaki
1-1 Gakuen-kibanadai nishi, Miyazaki 889-2192, Japan

Tomohiko Takagi
Department of Engineering and Design, Faculty of Engineering and Design, Kagawa University
2217-20 Hayashi-cho, Takamatsu-shi, Kagawa 761-0396, Japan
E-mail: s19t332@kagawa-u.ac.jp, kat@cs.miyazaki-u.ac.jp, takagi@eng.kagawa-u.ac.jp

Abstract

Extended Place/transition Net with Attributed Tokens (EPNAT) is one of formal modeling languages, and it enables system engineers to construct an executable and abstracted behavioral model of multiple software systems. In this paper, we propose an automated random simulation technique of an EPNAT model in order to detect failures in the model. In the simulation, input is randomly selected for model execution. When a constraint given for each system or multiple systems is violated through the model execution, a failure is revealed. The simulation is terminated by the detection of a failure or the satisfaction of a criterion focusing on the combination of marking, data writing and reading between different systems. A prototype tool of the simulation technique was developed and applied to a trial model to discuss its effectiveness. Two failures inserted into the trial model were successfully detected, and a few challenges were found in the experiment.

Keywords: formal modeling, place/transition net, VDM, simulation, failure detection

1. Introduction

The collaborative behavior of multiple software systems can provide advanced services and functions, but also can be a contributory factor in the occurrence of failures. Systematic quality control techniques need to be constructed so that system engineers can detect such failures and improve the reliability effectively. Therefore, in this paper, we propose an automated random simulation technique for checking an abstracted collaborative behavior of multiple systems based on an Extended Place/transition Net with Attributed Tokens (EPNAT).

EPNAT [1] is a relatively new formal modeling language developed from PN (Place/transition Net), and includes the following additional elements written in VDM++ [2]:

- Attributes (variables for each token to characterize the behavior of systems or subsystems)
- Actions (procedures of data processing in each transition of systems or subsystems)
- Basic constraints (conditions that should be satisfied in each variable and transition, that is, pre-conditions, post-conditions, and invariants)

By using EPNAT, engineers will be able to construct an executable and unambiguous model for the abstracted collaborative behavior of multiple systems, which is

called EPNAT model or simply model in this paper. In our previous study [3], a semi-automated simulation technique was proposed to visualize the behavior of a given EPNAT model and arbitrarily search its common failures. In the semi-automated simulation, engineers need to manually select input to execute a model, and need to manually evaluate some constraints. Note that basic constraints can be automatically evaluated, but there are complex requirements among systems that are not so easy to be defined as basic constraints. On the other hand, in the automated random simulation we propose in this paper, the selection of input and the evaluation of non-basic constraints are automatically executed to reduce engineers' effort. Another study [4] shows a test stopping criterion to cover successive markings and transitions based on coverability trees of a PN model, but it is not suitable for the automated random simulation of an EPNAT model. Therefore, a new stopping criterion for the model execution is introduced to concentrate on the detection of failures on the collaborative behavior. It focuses on the combination of marking, data writing and reading between different systems. This idea is related to data flow testing [5].

This paper is organized as follows. In section 2, we propose the automated random simulation technique that consists of a model execution algorithm and a stopping criterion of the model execution. Section 3 shows the results of a preliminary experiment using our prototype tool and a trial model, and gives discussion about the effectiveness of the proposed technique. Finally, section 4 describes conclusion and our future work.

2. Automated Random Simulation Technique

In this section, we propose the automated random simulation technique that consists of a model execution algorithm and a stopping criterion of the model execution.

2.1. *Model Execution Algorithm*

The input to the algorithm is a model to be checked, non-basic constraints, and the initial state of the model. The non-basic constraints are based on the engineers' aim of checking, and are defined by the engineers for markings, variables, and fire of transitions. They are classified into type A and B according to the timing of evaluation. Type A means that the constraints should be satisfied among systems at all times or at the timing of specific execution, and they can often be replaced with basic constraints. Type B means that sometime the constraints should be satisfied among systems. A state of a model is represented by a marking, and values of variables in this study. The initial state is defined by the engineers according to their aim of checking.

The output from the algorithm is a failure of the model. If the stopping criterion has been satisfied without any failures, the output is null. Also, the process of model execution can be outputted as animation, since EPNAT is a visual language.

The algorithm consists of the following six steps:

[Step 1] The given model is initialized to the given initial state.

[Step 2] A set of transitions that may be fireable is created according to only the current marking. Hereinafter, the set is referred to as *T*. Note that the fireability is determined by not only the current marking but also pre-conditions of each transition that are addressed in a later step.

[Step 3] If *T* is empty and the current state is not any of final states that were defined as a part of the given model, or if *T* is not empty and the current state is one of the final states, then this situation is reported as a failure and this algorithm is terminated. If *T* is empty and the current state is one of the final states, then this algorithm returns to Step 1. Otherwise, a transition is randomly selected from *T*. Hereinafter, the selected transition is referred to as *t*.

[Step 4] The pre-condition of *t* is evaluated. If the pre-condition includes any arguments of *t*, their values are randomly selected from a set of valid values that makes their invariants and the pre-condition true. If the set of valid values is empty or if the result of the evaluation is false, then *t* is removed from *T* and this algorithm returns to Step 3.

[Step 5] *t* is fired with valid values that are randomly selected for its arguments. The action of *t* is executed, and then the current state is changed. All the related invariants, the post-condition of *t*, and the given non-basic constraints (type A) are evaluated to detect failures. If there are any constraints whose evaluation results are false, then this situation is reported as a failure and this algorithm is terminated.

[Step 6] If the stopping criterion discussed in the next subsection is not satisfied, this algorithm returns to Step 2. Otherwise, the given non-basic constraints

(type B) are evaluated to detect failures. If there are any constraints whose evaluation results are false, this situation is reported as a failure. This algorithm is terminated.

2.2. *Stopping Criterion*

In EPNAT models, glue transitions play an important part in the collaborative behavior of multiple systems, and thus we focus on the glue transitions in order to construct the stopping criterion.

One model consists of multiple sub-models that correspond to multiple systems. The sub-models are connected with each other by glue transitions. When a glue transition is fired, the writing and/or reading of attributes among connected sub-models is executed in its action. A result of the reading of an attribute depends on the preceding writing of the attribute. Also, the results of the writing and reading are related to markings in which they are executed.

According to the above, the combinations of markings, writing and reading that are related to glue transitions should be covered in the model execution. A set of the combinations, which is referred to as C, is defined as follows:

$$C = \bigcup_{a \in A} \left(\bigcup_{w \in W_a} (M_w \times \{w\}) \times \bigcup_{r \in R_a} (M_r \times \{r\}) \right) \quad (1)$$

where A expresses a set of attributes. W_a and R_a express sets of the writing and reading of attribute a ($a \in A$), respectively. One of or both of w ($w \in W_a$) and r ($r \in R_a$) in each element of C should be included in action of a glue transition. M_x ($x \in W_a \cup R_a$) expresses a set of markings in which x is executed.

Each element of C is called check object in this study. Not all check objects defined by equation (1) are usually feasible, and the strict evaluation of feasibility on software is known as a hard problem to solve. Therefore, when the number of check objects that have been actually reached through the model execution converges, Step 6 concludes that the stopping criterion is satisfied.

3. Discussion

This section shows the results of a preliminary experiment using our prototype tool and a trial model.

In this study, we are developing a prototype of a tool that has the functions to edit an EPNAT model and to execute the automated random simulation of the model. A screenshot of the prototype tool is shown in Fig. 1. Also, we created a trial model that represents the collaborative behavior of a simple student management system and book management system. The model satisfies the following two non-basic constraints:

[C1] Each student cannot borrow multiple books.

[C2] Sometime each student can log out.

C1 and C2 are type A and B, respectively.

We executed the automated random simulation by applying the trial model, C1, C2 and an initial state of the trial model to the prototype tool. The overview of its result is shown in Fig. 2. No failures had been detected, and it met our expectations. 64 check objects had been successfully reached on the trial model. The number of the reached check objects converged when transitions had been fired about 2,500 times. It grows rapidly in the earlier stage of the model execution, since the algorithm can easily find new execution paths on the given model. There are check objects that the algorithm failed to reach, which will be due to its random search approach.

Additionally, we created two faulty models F1 and F2 that do not satisfy C1 and C2 respectively, and then similarly executed the automated random simulation of

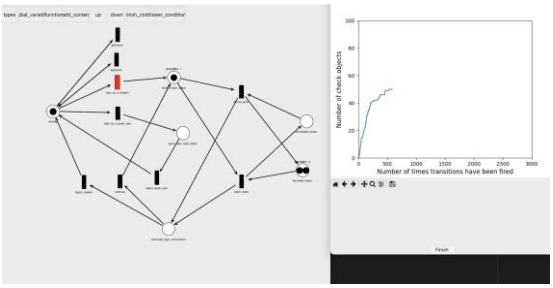


Fig. 1. A screenshot of our prototype tool.

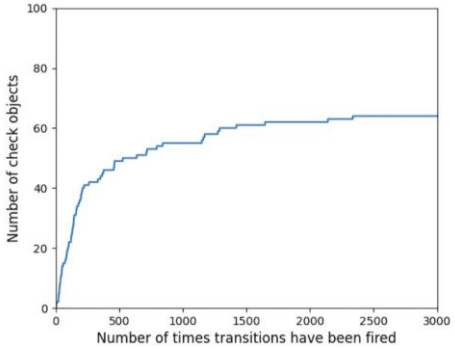


Fig. 2. Overview of an experimental result.

them five times. As their results, when transitions had been fired an average of 17.6 times in F1, the simulation was automatically terminated, and successfully reported a failure that is the violation of C1. The simulation of F2 was terminated after the number of reached check objects had been converged. It reported that C2 had not been satisfied, and implied the existence of a failure. Note that type B constraints cannot reveal failures as established facts. It is expected that the proposed technique will be able to detect failures to some extent. However, the faulty models and non-basic constraints used in this experiment are relatively small and simple, and thus additional experiments are needed to strictly evaluate its effectiveness.

4. Conclusion and Future Work

In this paper, we proposed an automated random simulation technique of an EPNAT model in order to detect failures in the model. Also, we developed a prototype tool of the technique, and applied a trial model to it. Two failures inserted into the trial model were successfully detected, and a few challenges were found in the experiment.

In future study, we plan to (1) improve the model execution algorithm so as to reach check objects efficiently, (2) develop a technique to express more complex non-basic constraints, and (3) evaluate the effectiveness by using large and complex models.

Acknowledgements

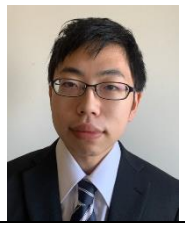
A part of this work was developed from the graduation thesis written by Mr. Hikaru Morimoto at Kagawa University. This work was supported by JSPS KAKENHI Grant Number JP22K11976, and Young Scientists Fund of Kagawa University Research Promotion Program 2021 (KURPP).

References

1. T. Takagi and R. Kurozumi, Software Modeling Technique and its Prototype Tool for Behavior of Multiple Objects Using Extended Place/Transition Nets with Attributed Tokens, *Journal of Robotics, Networking and Artificial Life*, Vol.7, No.3, pp.194-198, Dec. 2020.
2. J. Fitzgerald, P.G. Larsen, P. Mukherjee, N. Plat and M. Verhoef, *Validated Designs for Object-Oriented Systems*, Springer-Verlag London, 2005.
3. T. Takagi and R. Kurozumi, Simulation and Regression Testing Technique for Software Formal Specifications Based on Extended Place/Transition Net with Attributed Tokens, *Journal of Robotics, Networking and Artificial Life*, Vol.8, No.2, pp.112-116, Sep. 2021.
4. T. Takagi and Z. Furukawa, Test Case Generation Technique Based on Extended Coverability Trees, *Proceedings of 13th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, IEEE, Kyoto, Japan, pp. 301-306, Aug. 2012.
5. B. Beizer, *Software Testing Techniques*, Van Nostrand Reinhold, 2nd edition, 1990.

Authors Introduction

Mr. Sho Matsumoto



He is an undergraduate student in the Faculty of Engineering and Design at Kagawa University. His research interests are in software engineering, particularly software quality control.

Dr. Tetsuro Katayama

He received a Ph.D. degree in engineering from Kyushu University, Fukuoka, Japan, in 1996. From 1996 to 2000, he has been a Research Associate at the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. Since 2000 he has been an Associate Professor at the Faculty of Engineering, Miyazaki University, Japan. He is currently a Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include software testing and quality. He is a member of the IPSJ, IEICE, and JSSST.

Dr. Tomohiko Takagi

He received the B.S., M.S. and Ph.D. degrees from Kagawa University in 2002, 2004 and 2007, respectively. He became an assistant professor in 2008, and a lecturer in 2013 in the Faculty of Engineering at Kagawa University. Since 2018 he has been an associate professor in the Faculty of Engineering and Design at Kagawa University. His research interests are in software engineering.