멀티쓰레드 어플리케이션 검증을 위한 도구의 비교분석

압둘라힘 O, 김가연, 이욱진
한양대학교 컴퓨터공학과
rahim750413@hanyang.ac.kr, ruredi@hanyang.ac.kr, scottlee@hanyang.ac.kr

Comparative Analysis on Tools for Verifying Multithreaded Applications

Abdul Rahim Mohamed Ariffin O, Gayeon Kim, Scott Uk-Jin Lee
Department of Computer Science and Engineering, Hanyang University

Abstract

The growth rate for the usage of multithreaded applications has been increasing rapidly along with improved modern day software. For that reason, effective verification tools is needed. With the rapid growth in software development, there has been a number of verifiers that have been introduced due to the lower quality of software that were developed. Therefore, the purpose of this paper is to analyze the comparison of different tools for verifying multithreaded applications. Comparison and evaluation to find an effective verification tool for multithreaded applications will also be shown in this paper.

1. Introduction

Verification of multithreaded applications are becoming more important so that the quality of a multithreaded application is maintained without the need to provide more hardware features to increase the performances. Although there are various existing verifiers, there still exist issues concerning the development of a multithreaded application. There exist non-determinism issues that results in unpredictability of multithreaded applications, and also determinism issues which leads to the inconvenience of developing multithreaded applications for beginner level programmers. Therefore, the importance of a verifier to identify the correctness and completeness of multithreaded applications in order to support the developers has become a priority which need to be resolved. However, there are a number of good verifiers which are able to do verification from the abstraction level of a software design model and also verify the model of a multithreaded application such as Promela Language in SPIN. Alloy Analyzer were also introduced, which it derives from a first-order logic of a mathematical formulae to verify the design of a concurrent system. Java-based verifiers such as Veri-Fast [1] which used for verifying single-threaded and multithreaded C and Java programs. Veri-Fast support permission accounting which monitor the permissions given throughout a multithreaded Java programs. Although there exist non-determinism, these verifiers are still an effective and efficient verification tools.

2. Verification Tools

Spin in [2], is a generic verification system that supports the design of a software model. It verifies the correctness of a system's process interactions in the design phase in an asynchronous software system. Spin generates C source code as its framework to run the verifier. Promela is a model specification language of Spin used to simulate the verifier with the given design of a software model. Spin starts with the specification of a high level model of a concurrent system, therefore to verify the correctness of a multithreaded application it may be more suitable to use Spin.

Another verifier called Alloy Analyzer [3] is a symbolic model checker. Alloy uses the modeling language of a first-order logic to verify the design of the software model. Alloy consists of a set of signatures, noted (sig), and used to define its sets and relations. Alloy uses constraints and facts, to allow conditions in the values of the sets and relations. Alloy uses SAT solvers to verify the satisfiability of axioms defined in a model and provides counterexamples that follows the axioms. Figure 1 shows the basic architecture of a model checker for verifying a software model.

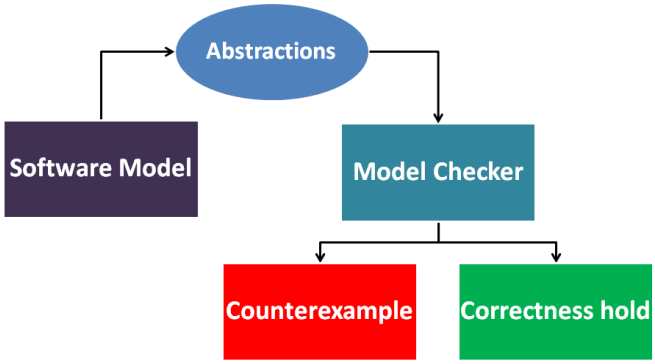

Figure 1. Shows the basic architecture of a model checker

Another verifier introduced in recent years is Threader [4], a verifier that implements rely-guarantee methods for the compositional verification of multithreaded applications. It uses compositional proof rules as its roots and automates the verification of a multithreaded program. There is another verifier which introduced in [5] called Blast, which able to verify Java language based multithreaded programs. However, Blast only able to do verification during the runtime execution of the program.

3. Comparison

In selecting the most effective verification tools, we conducted a survey on various known verifiers that needs to meet a specific set of queries in order to support the developers to program a good multithreaded application. Based on the research in [6] and [7], we have analyzed and concluded a few queries that may help in improving the quality and efficiency of a verification tool. Therefore, we present these queries along with the concerns into selecting an effective verifier for a multithreaded application.

Each verifier must be able to answer these following questions:

1) Is the verifier able to provide counterexamples to achieve soundness and completeness in a multithreaded application?
2) Is the verifier able to specify a complete set of false negatives and false positives produced by the application?
3) Will the verifier affect the system's functionality?
4) Does achieving complete correctness decreases the burden in developing a multithreaded application?

Based on the results of applying these queries, we performed evaluation in order to find the most effective verification tools. We have compared with previous verifiers to select the best verifier. The selected verifier should be able to comply with the aforementioned queries along with some improvement. Table 1 shows the effectiveness of a verification tools for a multithreaded application and its efficiency in providing counterexamples. Based on various verifiers that we have evaluated, Spin was found to have answered most of the specified queries that we have provided. Although Spin works well in supporting concurrent system, the verifier may still be improved using the following suggestions:

1) In analyzing the developer's understanding and confirming the requirements for developing the multithreaded application, Spin may be able to identify false positives and false negatives, thus provide counterexamples.
2) The ability of Promela to distinguish programming language such as Java, Python, etc., would enable it to be utilized for other multithreaded application from different platforms, thus overcoming portability issues.
3) Enabling Spin to be able to verify other programming language codes and not only C language program codes.

Table 1. Comparison of verification tools

Verification Tools / Queries	Spin	Alloy	Blast	Threader	Veri-Fast
Provide counterexamples	Yes	Yes	Yes	Yes	No
False positives and negatives detection	Yes*	No	No	No	No
Verification affects system	No	No	Yes	Yes	Yes
Helps develop multithreaded application	Yes	Yes	Yes	Yes	Yes

* with specific constraints to identify false positives and false negatives inputs

4. Conclusion

Model checkers or verifiers exists for the purpose of producing the most accurate and precise software model according to the requirement of the user and using the latest technology at hand without overestimating the output of the product. Verifiers provide more logical options for developers in order to develop a correct and efficient software model. In this paper, we have discussed known model checkers or verifiers and we have selected the best among selected verifiers. Based on our findings, Spin is very accurate in providing a concurrent software model and it is very efficient in verifying a multithreaded application. However with the suggested enhancement that Spin requires, Spin will be able to produce a more accurate and precise result in the future.

Acknowledgment

This work was supported by the ICT R&D program of MSIP/IITP.[R0601-15-1063, Software Platform for ICT Equipment]

References

[1] Bart Jacobs, Jan Smans, Pieter Philippaerts, Frédéric Vogels, Willem Penninckx, and Frank Piessens, “VeriFast: a powerful, sound, predictable, fast verifier for C and Java.” In Proceedings of the Third international conference on NASA Formal methods, pp. 41-55, 2011

[2] Gerard J. Holzmann, "The Model Checker SPIN." IEEE Transactions on Software Engineering, Vol. 23, No. 5. pp. 279-295, 1997

[3] Daniel Jackson, Ian Schechter, Ilya Shlyakhter, "Alcoa: the Alloy constraint analyzer." Proceedings of the International Conference on Software Engineering, pp. 730-733, 2000

[4] Ashutosh Gupta, Corneliu Poppeea, Andrey Rybalchenko, “Threader: A Constraint-Based Verifier for Multithreaded Programs” Computer Aided Verification, LNCS, pp. 412-417, 2011

[5] Dirk Beyer, Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, “The Software Model Checker Blast.” International Journal on Software Tools for Technology Transfer, vol. 9, no. 5, pp. 505-525, 2007

[6] Marc Frappier, Benoît Fraikin, Romain Chossart, Raphaël Chane-Yack-Fa, Mohammed Ouenzar, “Comparison of Model Checking Tools for Information Systems.” International Conference on Formal Engineering Methods, LNCS 6447, pp. 581-596, 2010

[7] Moonzoo Kim, Yunho Kim, Hotae Kim, “A Comparative Study of Software Model Checkers as Unit Testing Tools: An Industrial Case Study.” IEEE Transactions on Software Engineering, vol. 37, no. 2, pp. 146-160, 2011