ICICTS2015 International Conference on Information and Convergence Technology for Smart Society
Jan. 19-21, 2016 in Ho Chi Minh, Vietnam

A Survey on Correctness Properties for Multithreaded Applications

Abdul Rahim Mohamed Ariffin, Isma Farah Siddiqui, Gayeon Kim, Scott Uk-Jin Lee
Department of Computer Science and Engineering, Hanyang University, South Korea
rahim750413@hanyang.ac.kr, isma2012@hanyang.ac.kr, scottlee@hanyang.ac.kr

Abstract— **Correctness property for concurrent systems is essential in checking multithreaded applications. In current multithreaded applications development, there exist a number of applications that have been verified to be incorrect or that produces unpredictable errors. This is due to no proof of correctness in stating that the correctness property holds in the development of multithreaded applications. There exist a number of tools and correctness property checkers, which only focuses on holding one of the correctness properties. This paper presents an analysis on existing verification tools and do evaluation on the correctness properties.**

Keywords-component; multithreading, verification tool, correctness property

I. INTRODUCTION

Correctness and completeness of a multithreaded application are an essential property to provide a correct multithreaded application. Therefore, there exist a number of verification tools to support and do verification on correctness properties in multithreaded applications. However, these tools cover not more than two correctness properties in verifying multithreaded applications. This occurs due to the different approaches and goals by developers of each proposed tools. Furthermore, the difficulties and constraints to develop multithreaded applications are also among major concerns in analyzing the correctness properties. Therefore, the purpose of this paper is to conduct a survey on the existing verification tools and to perform an analysis on the possibilities of a tool to adopt multiple correctness properties. The tool should also be able to verify the completeness and soundness of a multithreaded application while detecting false positives and false negatives inputs. In order for these criteria to be met, the tool should be able to perform verification of multithreaded applications correctly.

There exist verification tools that are used to perform atomicity checking such as [1], [2] and [3]. Others enable detection techniques on atomicity violations such as in [4]. These tools and techniques focus on atomicity checking, where one of the correctness properties is to prove atomicity to correctly develop concurrent programs. However, only by checking atomicity conditions is not enough to correctly verify multithreaded applications. This is due to its difficulty to prove deadlocks and data race occurrences in a concurrent system by only detecting atomicity violations. This is explained in [5], where the tools proposed is used to avoid deadlocks for multithreaded applications. Deterministic behavior of an application is also considered as one of the correctness properties, which has been mentioned in [6], [7].

II. CORRECTNESS PROPERTY

Atomicity can be described as a method that is atomic if for every arbitrarily interleaved program execution, there is an equivalent execution with the same overall behavior where the atomic method is executed serially [1]. Therefore, to detect where the error has occurred after each interleaving program execution is difficult when atomicity violation occurs. In [2], the authors proposed strong atomicity, which allows the atom blocks to be overlapped. Then the atom blocks will be executed one by one and the compiler will analyze which atom blocks will be executed first. However, the tools proposed can be exhaustive and overwhelming when dealing with a larger data set in multithreaded applications. Determinism is also one of the correctness property that needs to be considered in developing a correct multithreaded application. Determinism is a method where every event or state will be executed in a deterministic manner. SingleTrack is a tool to develop a dynamic analysis for verifying non-interference specification of multithreaded applications that uses this concept [5]. SingleTrack able to do detection which concurrent programs are deterministic. However, by performing detection deterministically, the verification tool ignores atomicity checking. This is due to the deterministic behavior of the program executions that is being executed deterministically while considering that the atomicity violation to not occur. Linearizability is a correctness conditions for concurrent objects that exploits the semantics of abstract data types [11]. Linearizability is achieved when there exist serial executions, which holds for both final program states with atomic blocks in the same execution and there exist no overlapping for the atomic block executions. Another correctness property for multithreaded applications is serializability. Serializability is achieved when there exist serial executions, which holds for both final program states with atomic blocks in the same execution. A transaction schedule is serializable if its outcome is equal to the outcome of its transactions executed serially [13].

Table 1. Correctness property for multithreaded application coverage based on existing tools

Verification tools / Correctness property checkers	Correctness Property for Multithreaded Applications					
	Atomicity violation detection	Determinism checking detection	Race condition detection	Deadlock detection	Linearizability	Serizalizability
Atomizer	Yes	No	Yes	Yes	No	No
SingleTrack	No	Yes	Yes*	No	No	No
Gadara	No	No	Yes	Yes	No	No
Verifast	No	Yes	Yes	No	Achieved	-
Spin	Yes	No	Yes	Yes	Achieved	-
Techniques / Methodologies						
Strong Atomicity	Yes	No	No	No	No	No
Determin	Yes	Yes	No	No	No	No
Semantic Atomicity	Yes	Yes	No	No	Yes	Yes

* - It is assumed that the race condition detected is already handled

III. Analysis on Exisiting Tools

A. Comparative Evaluation

Based on Table 1, Atomizer performed a full detection on atomicity violations and race conditions. However, it provided no detection for determinism in multithreaded applications. Similarly SingleTrack only performed detection on determinism and deadlock. The verification tools such as Verifast [9] and Spin [10] are among other verification tools known to correctly perform verifications for multithreaded applications. However, existing tools to correctly perform verifications for multiple correctness properties is not mentioned. Verifast is a verification tool based on separation logic but it does not provide verification for deadlock detection. It is assumed to be prevented due to the implementation of atomicity in the program executions. Verifast assumes that when performing verification program execution, it is considered as atomic. Spin, a verification tool designed for verifying multithreaded applications by applying high level language system description called PROMELA [12], also does not provide determinism checking. This is because Spin performs program executions non-deterministically and performs verification based on the assertion constraints provided by the tool. Therefore, Spin is not able to perform determinism checking for multithreaded applications. Table 1 also shows a number of techniques for detecting correctness properties in concurrent systems. The techniques and methodologies proposed have been proved to accurately hold correctness properties. Therefore, by applying these techniques to develop multithreaded applications, it will provide more accurate and precise results.

B. Multiple Correctness Property in Verification Tool

Previous work only focuses on one correctness property and other properties are not identified. In order to provide multiple correctness property checking in verification tool, compatibility and synchronicity of the correctness property is important. There are a number of techniques that are able to accurately perform detection and provide proof for correctness property in multithreaded applications. [2], [6], [8], provided decent techniques for each correctness property such as atomicity violations, determinism checking and deadlock avoidance. By applying these techniques to develop verification tools, it provides the ability to correctly verify multiple correctness properties. With the existing tools that we have discussed, Spin is considered to have covered the most conditions in verifying multithreaded applications correctly. Therefore, by implementing the techniques discussed, Spin will be able to provide more accurate and precise results when performing verifications on multiple correctness properties in multithreaded applications.

The goals of verification tools are completeness and soundness of concurrent programs. However, in order to achieve these goals, the tools must be able to correctly perform detection on false negatives and false positives inputs. The reason for this is to support developers to correctly develop multithreaded applications. Proving that multiple correctness properties are able to correctly verify an application will be the most important criteria in developing a verification tool. Thus, we provide an evaluation and analysis of the existing verification tools, which have different correctness criteria specifications.

IV. Conclusion

We have discussed correctness property of multithreaded applications based on existing checking tools and verification tools. There are various factors that have been discussed in previous work to acquire an effective tool which is able to correctly verify the multithreaded applications and detect correctness properties. The tool should able to provide counterexamples and do detection for multiple correctness properties. We have provided an analysis for the existing tools and evaluated the correctness properties which a tool must satisfy to correctly verify multithreaded applications. Hence, with the latest technology, an effective verification tool for

multithreaded applications can be developed. Based on the analysis, we have found that Spin will be able to comply and correctly verify multithreaded applications in collaboration with other effective correctness property detection techniques.

ACKNOWLEDGMENT

This work was supported by the ICT R&D program of MSIP/IITP. [12221-14-1005, Software Platform for ICT Equipment].

REFERENCES

[1] Cormac Flanagan, Stephen N. Freund, "Atomizer: A dynamic atomicity checker for multithreaded programs," Science of Computer Programming, 2008, vol. 71 issue 2, pp. 89–109.

[2] Kai Lu, Wenzhe Zhang, Xu Zhou, "Strong Atomicity: An Efficient and Easy-to-Use Mechanism to Guarantee Atomicity", International Conference on Computer Science and Service System, 2012, pp.562-565.

[3] Cormac Flanagan, Stephen N. Freund, Jaeheon Yi, "Velodrome: a sound and complete dynamic atomicity checker for multithreaded programs", Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2008, vol. 43 issue 6, pp 293-303.

[4] Jacob Burnim, George Necula, Koushik Sen, "Specifying and Checking Semantic Atomicity for Multithreaded Programs", ASPLOS, 2011, vol. 39 issue 1, pp. 79-90.

[5] Caitlin Sadowski, Stephen N. Freund, Cormac Flanagan, "SingleTrack: A Dynamic Determinism Checker for Multithreaded Programs", European Symposium on Programming, 2009, vol. 5502, pp 394-409.

[6] Claudio Basile, Zbigniew Kalbarczyk, Ravi Iyer, "A Preemptive Deterministic Scheduling Algorithm for Multithreaded Replicas", Proceedings of the International Conference on Dependable Systems and Networks, 2003.

[7] Jacob Burnim, Koushik Sen, "DETERMIN: Inferring Likely Deterministic Specifications of Multithreaded Programs", Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, 2010, vol. 1, pp. 415-424.

[8] Yin Wang, Terence Kelly, Manjunath Kudlur, St´ephane Lafortune, Scott Mahlke, "Gadara: Dynamic Deadlock Avoidance for Multithreaded Program", USENIX Symposium on Operating System Design and Implementation, 2008.

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

[10] Anna Zaks, Rajeev Joshi, "Verifying Multi-threaded C Programs with SPIN", SPIN, 2008, pp. 325-342.

[11] http://spinroot.com/spin/what.html

[12] Maurice P. Herlihy, Jeannette M. Wing, "Linearizability: A Correctness Condition for Concurrent Objects", ACM Transactions on Programming Languages and Systems, 1990, vol. 12, pp. 463-492.

[13] https://en.wikipedia.org/wiki/Serializability