Towards Scalable Verified Validation of Static Analyzers

Jeehoon Kang[1], Sungkeun Cho[1], Joonwon Choi[2], and Chung-Kil Hur[1]

[1] Seoul National University, Korea
[2] Massachusetts Institute of Technology, USA

Style of presentation: poster only

Static analyzers based on abstract interpretation are ideal target for formal software verification for the following reasons.

- **Clear specification:** The formal verification cost of a static analyzer based on abstract interpretation is not too expensive. Such an analyzer has a simple specification for its main part: the resulting abstract state should include all possible concrete states that can occur during execution of the analyzed program.
- **Importance of reliability:** Reliability of a static analyzer is important, especially when it is applied to safety-critical software, because reliability of analyzed software depends on that of the analyzer.
- **Difficulties in testing:** Reliability of a static analyzer is hard to establish by testing. The reason is that whether an analysis result is valid is not evident because both valid and invalid results can contain false alarms. Moreover, if the analyzer is deliberately unsound, the results even can miss true alarms. Also, it is infeasible to detect bugs by manually examining intermediate data such as an abstract state of the analyzed program because such data is simply too big unless the analyzed program is very small.
- **Low verification cost:** The verification cost of analyzers can be reduced by using the verified validation approach[7–9]. The idea is to develop a validator that checks whether a given analysis result is valid or not, and then formally verify that the validator is correct. Because the validator is much smaller and simpler than the analyzer, we can greatly reduce the verification effort.

However, a downside of translation validation is that the runtime cost of validation might become too expensive. If we implement a naive validator in order to reduce its verification effort, then the runtime cost of the validator may be much larger than that of the analyzer. Indeed, an early version of our validator was 100 times slower than the associated highly-optimized analyzer. Thus there is a trade-off between *development scalability* and *runtime scalability*. By the former we mean scalability regarding development cost (*i.e.*, implementation and verification cost) of a validator, and by the latter regarding runtime cost of a validator.

In this work, we demonstrate that the verified validation approach is a *scalable* and *effective* method to establish reliability of static analyzers based on abstract interpretation. To demonstrate scalability, we developed a verified validator for a real-world static analyzer Sparrow[1–4, 6, 5], which uses various complex algorithms. To demonstrate effectiveness, using our validator we validated the analysis results of the analyzer for 16 open-source programs, during which we effectively identified and fixed 13 bugs using the validation results.

References

1. W. Lee, W. Lee, and K. Yi. Sound non-statistical clustering of static analysis alarms. In *Proceedings of the 13th international conference on Verification, Model Checking, and Abstract Interpretation*, VMCAI'12, pages 299–314, Berlin, Heidelberg, 2012. Springer-Verlag.
2. H. Oh. Large spurious cycle in global static analyses and its algorithmic mitigation. In *Proceedings of the Asian Symposium on Programming Languages and Systems*, volume 5904 of *Lecture Notes in Computer Science*, pages 14–29, Seoul, Korea, December 2009. Springer-Verlag.
3. H. Oh, L. Brutschy, and K. Yi. Access analysis-based tight localization of abstract memories. In *VMCAI 2011: 12th International Conference on Verification, Model Checking, and Abstract Interpretation*, volume 6538 of *Lecture Notes in Computer Science*, pages 356–370. Springer, 2011.
4. H. Oh, K. Heo, W. Lee, W. Lee, and K. Yi. Design and implementation of sparse global analyses for C-like languages. In *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation(***PLDI***)*, 2012.
5. H. Oh, W. Lee, K. Heo, H. Yang, and K. Yi. Selective context-sensitivity guided by impact pre-analysis. In *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation(***PLDI***)*, 2014.
6. H. Oh and K. Yi. Access-based localization with bypassing. In *APLAS 2011: 9th Asian Symposium on Programming Languages and Systems*, volume 7078 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2011.
7. J.-B. Tristan and X. Leroy. Formal verification of translation validators: A case study on instruction scheduling optimizations. In *POPL*, 2008.
8. J.-B. Tristan and X. Leroy. Verified validation of lazy code motion. In *PLDI*, 2009.
9. J.-B. Tristan and X. Leroy. A simple, verified validator for software pipelining. In *POPL*, 2010.