A Scalable Stacking-based Context-Sensitive Points-to Analysis for Java

Xin Li

Japan Advanced Institute of Science and Technology
li-xin@jaist.ac.jp

The precision of Java points-to analysis has always to be sacrificed for practical scalability. In particular, almost all the existing scalable analysis are based on cloning calling contexts [2] to obtain context-sensitivity. As such, recursive procedure calls have to be over-approximated inevitably in the analysis. However, empirical study on practiced Java benchmarks shows that typically more than one thousand methods are involved in strongly connected components of the call graph. Therefore imprecise treatment on various recursions can potentially induce a great loss on precision. In view of this, we exploit a so-called *stacking-based* approach to context-sensitivity. By encoding the program as a pushdown system, the program calling context in terms of valid call paths are managed by the unbounded pushdown stack. Therefore, procedure calls are guaranteed to always return to the most recent call sites, and no limit is required on the call depth and recursions. We present and develop Japot, a scalable stacking-based context-sensitive points-to analysis for Java with no restriction on procedure calls. The novelty of our proposal is improving program modelling and iterative procedures in the analysis, which are keys to make the analysis scalable. Our empirical studies shows that the analysis scales well to Java applications of significant size. A pushdown system is known to naturally model procedure-oriented programs. We present ideas for modelling and analyzing object-oriented program features with pushdown model checking techniques. Applied to points-to analysis, such an extension provides the analysis with context-sensitivities regarding heap abstraction, heap access and call graph construction. Program analysis is observed to be regarded as model checking of abstract interpretation. Following this methodology, program analysis naturally enjoys soundness by applying abstract interpretation and "push-button" facilities from model checking. Based on this insight, we exploit weighted pushdown model checking [1] as the underlying analysis engine of our static analysis.

References

1. T. Reps, S. Schwoon, S. Jha, and D. Melski. Weighted pushdown systems and their application to interprocedural dataflow analysis. *Sci. Comput. Program.*, 58(1-2):206–263, 2005.
2. J. Whaley and M. Lam. Cloning-based context-sensitive pointer alias analysis using binary decision diagrams. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2004)*, pages 131–144, 2004.