Counterexample-guided abstraction refinement for points-to analysis of object-oriented programs

Vu Quang Vinh (1410213)

School of Information Science,
Japan Advanced Institute of Science and Technology

August 04, 2016

Keywords: abstract refinement, points-to analysis, context-sensitive, Datalog.

Points-to analysis is a program static analysis technique. It statically computes points-to relation between pointers and objects. It is considered as a reachability problem and solved by a state transition system. The precise pointer analysis is undecidable, and that abstracting state space to be finite makes it decidable at the cost of imprecision. For an example, in context-sensitive points-to analysis, the context length is bounded by a maximum depth k and it becomes a k-context-sensitive analysis. The state space increases exponentially in k and the higher abstraction produces more precise result. Hence, there is a trade off between the precision of the abstraction and the efficiency of the analysis. Basically, the main problem is finding a good abstraction which reduces the number of states but still provides precise analysis results.

In my Master's thesis research, I focus on obtaining the higher precision by cheaper abstractions. I propose an adaptive context-sensitive analysis. Each part of the analyzed program is treated differently and the abstraction interpretation becomes flexible. Then, I follow counterexample-guided abstraction refinement(CEGAR) approach to design a CEGAR-based algorithm. It repeatedly performs an analysis step and a refinement step. First, the analyzing phase executes the adaptive points-to analysis and produces counterexamples. Then the refinement step picks a good abstraction for the next iteration by the guidance of the accumulated counterexamples.

Finally, I have successfully designed and implemented our analysis system based on the Doop framework. My experiments are performed on real Java programs in the DaCapo benchmark. Our CEGAR-based analysis obtained better results compared to non-adaptive context-sensitive analysis.

Contributions

- I propose a fine-grained adaptive points-to analysis. It is able to precisely analyze the essential program parts. Meanwhile, it saves the cost by treating the other parts cheaply.
- We do experiment on the Doop framework with real Java programs of the DaCapo benchmark.

- Using MaxSAT for abstraction choosing was proposed in 2014 by Zhang et. al., but it does not work with the Doop framework and big Java programs such as DaCapo's programs. I propose a partitioned (for performance) and repeated (for precision) MaxSAT abstraction picking.