Project (Lecture Notes) for Compiler Construction course (VYPe)

Purple Dragon book - Chapter 12.5 & 12.6 (Context-Insensitive Interprocedural Analysis & Context-Sensitive Pointer Analysis)

Ondřej Horníček, xhorni09@stud.fit.vutbr.cz,
Faculty of Information Technology, Brno University of Technology, 3rd November 2012

Abstract

In my presentation I discuss two ways of analyses within the scope of the whole program, context-insensitive interprocedural analysis and context-sensitive pointer analysis. Despite the naming, they are not exact opposites, we will use same methods and models in describing the process.

The first and relatively easier part does not take history of function, procedure and method calls into consideration, hence the analysis is context-insensitive. It uses copy statements, does not track actual values of input/output parameters. But for simplification, it still needs to determine the types of objects, especially in languages like Java, where object-orientation and inheritance cause impossibility to identify receiver object precisely. The technique used to thin the possible class set is introduced. Call targets are computed from the call graph, where edges represent method invocations links between procedures and call sites, where those procedures are called. This graph is computed on the fly, iteratively until no points-to set can be changed and no new call targets are found. For that purpose, we use Datalog language, similar to Prolog, which describes data-flow analysis.

The second part, analysis for pointers, has to be context-sensitive but as mentioned earlier, there are similar ways to handle it. Cloning-based context-sensitive analysis is introduced, producing clone-based call graph, where for different contexts new clones of procedures are created, and context-insensitive analysis is used on them. Simple and mutually recursive functions are found by graph algorithms in cloned-based call graph as strongly connected components and handled in special way to avoid infinite loops and the node explosion. Again the Datalog implementation of algorithm is shown.

Presented solutions improves the effectiveness of analysis but still need some improvements.