Static Analysis of Object-Oriented Programs

Christoph Steindl

Institute for Practical Computer Science, System Software Group
Johannes Kepler Universität Linz, A-4040 Linz, Austria
{steindl}@ssw.uni-linz.ac.at

Static Analysis of Object-Oriented Programs

Static analysis derives information by inspection of the source code. The information must be valid for all possible executions of the program. Conservative assumptions must be taken if the program uses conditional branches and iteration since it is not known at compile time which branches will be taken at run time and how many iterations there will be. Static information is necessarily less precise than dynamic information (obtained by monitoring one specific execution of a program); but it can be computed once for all possible executions, whereas dynamic information must be computed again and again.

Two main concepts of object-oriented programming are polymorphism and dynamic binding. These dynamic aspects are difficult to integrate into static analysis, e.g. in most cases the exact destination of polymorphic call sites cannot be determined by static analysis. Additionally, data flow analysis for heap allocated objects is difficult. Since the number of objects is unbounded, they cannot be handled individually. If they are classified into groups, then all objects of a group are aliases for the data flow analysis.

We have developed a program slicer that models dynamic aspects of object-oriented programs correctly. Starting from conservative assumptions about dynamic binding and aliases, new user guidance techniques are used to reduce these assumptions. In this way, static analysis can be enriched with user-supplied knowledge to yield information with a precision similar to dynamic information.