Abstract

Programming languages are ever evolving, with new languages being invented to solve new problems, and old languages being extended to solve old problems in new ways. With the continued evolution of programming languages, and with new and improved static program analyses, we need flexible systems for building our static analyses and compilers. Declarative programming with Reference Attribute Grammars (RAGs) is a fruitful approach for building extensible and maintainable static program analyses and compilers. With declarative programming, code is easier to write, easier to understand, and easier to extend.

This thesis presents contributions to declarative static program analysis implementation with RAGs. In particular, I have developed new language extensions for the Java programming language, as well as new static analyses and tools for Java, based on the extensible Java compiler ExtendJ. Language extensions include an implementation of Java 7, and a new programming mechanism, multiplicities. A new static analysis-based tool presented in this thesis is a regression test selection tool, which reduces testing time for Java software development.

My contributions also include new design patterns for declaratively specifying static analyses in RAGs, and significant improvements to the ExtendJ compiler. My work in ExtendJ includes developing the first version of the compiler with fully declarative static analysis.

This dissertation also includes contributions to RAGs, including new algorithms for concurrent attribute evaluation with implementation in the JastAdd metacompiler. With these new algorithms, it is possible to parallelize any RAG-implemented analysis. In particular, I parallelized static analysis in the ExtendJ compiler to achieve a twofold speedup and orders of magnitude reduction of latency.