

Abstract

The static aspect weaver Yiihaw uses a strategy of bytecode transformation to implement aspect-oriented programming. This strategy has been shown to produce highly efficient woven code. The weaver performs extensive checks at weave-time to ensure correctness, but the rules it employs are conservative and precludes many meaningful programs.

This thesis project builds on Yiihaw by simplifying and extending its semantics with a strong focus on the type checking performed as part of the weaving algorithm. The strategy of the project is to apply formal type theory in a semi-formal manner to develop a consistent notion of correctness and type safety of a static weaving. This foundation is then used to develop and implement new type rules in the Yiihaw weaver. Lastly, the semantics of Yiihaw are further generalized to support parametrized aspects.

The contributions of this project are thus (1) the development of a consistent notion of correctness and type safety of a static weaving, (2) a type system for Yiihaw that allows for a more flexible, yet type safe application of aspects, and (3) a generalized semantics and type system for weaving parametrized aspects.

Each of the developments in the project is implemented in the Yiihaw prototype along with a greatly revised pointcut language and an extensive test suite.