

COMP 425—Advanced Concepts in Programming Languages

Department of Mathematics and Computer Science
Macalester College

What?

This course, run in a seminar format, covers a variety of current research topics in the design of programming languages. We read current research from the professional literature, and each student prepares a collection of papers on topics of interest and gives presentations to the class. The final course project involves investigation of one topic in depth, and can be very theoretical or practical in nature.

The topics investigated include the following: models of computation and their influence on language design, types and type theories (algebraic and recursive types, subtypes, polymorphism, type inference), foundations of object-oriented languages, and generalizations (Agent-oriented languages, Aspect-oriented languages), language specification and representation methods, mechanization of language definitions, abstract interpretation, formal semantics, and program transformation (techniques including interpretation, and compilation, partial evaluation, graph reduction, continuation passing methods).

Why?

Programming languages have been undergoing rapid development and change for more than 30 years. Each new language of the future seems to maintain its dominance in professional practice for shorter and shorter time intervals, until a new language comes to replace it, or evolving practice shows its limitations. Additionally, as is clear with human languages, it is becoming understood that even when one language is dominant for many applications, many languages will be used in practice, for there is no one best language. Thus it is important to study the nature of programming languages, to understand their design and use, and to follow their development and pay attention to the theoretical principles underlying them which will drive future change, so we can both influence and adapt to it.

Projects/Examples

- Theoretical foundations and applications of partial application as a program transformation technique.
- Development of an object oriented programming language based on a graphical user interface, so that language constructs are represented, and can be manipulated visually, rather than edited as low-level text.
- Formal semantics as a programming language design technique, in the development of a visual programming language affording the programmer multiple coherent views of programs under development.
- Surveys of Agent-oriented languages, and Aspect-oriented languages, as extensions to traditional object oriented languages.