CSE 341: Programming Languages

Hal Perkins

Spring 2011

Lecture 11— Modules; Abstract Types

Where are we

- Today: Modules
- Friday: Parametric polymorphism; Equivalence
- Monday: Scheme basics
- Wednesday: midterm
  - Does not include Scheme basics
  - You can have one side of one 8.5x11 sheet of paper
  - Old midterms posted shortly
  - Will read code, write code, and write English
  - Heavily biased toward later lectures because we have been building
  - (Old exams are difficult — maybe this quarter's too; don't panic.)

Modules

Large programs benefit from more structure than a list of bindings.

Breaking into parts allows separate reasoning:

- Application-level: in terms of module (in ML, structure) invariants
- Type-checking level: in terms of module types
- Implementation level: in terms of module code-generation

By providing a *restricted* interface (in ML, a signature), there are *more* equivalent implementations in terms of the interface.

Key restrictions:

- Make bindings inaccessible
- Make types abstract (know type exists, but not its definition)

SML has a much fancier module system, but we'll stick with the basics.

Abstract types are a "top-5" feature of modern languages.