CS 476: Programming Language Design

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Summary

- We’ve looked at PLs as *designers* (what’s in a language? how should it work?) and *implementers* (how do we get a computer to run it?)
- We described PLs with three different meta-languages:
  - Natural language (intuitive description, examples)
  - Math (grammars, type rules, big- and small-step semantics rules)
  - OCaml (typecheckers, interpreters, etc.)
- We examined the features of:
  - Imperative languages (variables, control flow, function calls)
  - Object-oriented languages (objects, inheritance, references)
  - Functional languages (functions as values, pattern-matching, type inference)
  - And more!

Summary

We’ve learned to:

- Write OCaml code (or code in another functional language)
- Identify and describe common language features
- Translate inference rules into code
- Think through the implications of adding features to a language
- Implement a new language, or add a feature to an existing language design

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

What else is there in PL?

- A lot more languages and language features!
- Actually making languages work: parsing, optimization, translation to machine code (see also CS 473)
- Social aspects: communities, tools, documentation, industry support and adoption, how languages get made and spread
- Metatheory: prove that well-typed programs return values of those types, prove that specific programs do what they're supposed to (see also CS 472)