Final Review

Frank Pfenning

15-150, April 30, 2020

What is Functional Programming About?

- Abstraction, abstraction, abstraction
  - A function abstracts over a computation
  - A data type abstracts over memory
  - A signature abstracts over a data structure
- Composition, composition, composition
  - A higher-order function composes abstractions
  - A polymorphic data type composes abstractions
  - A functor composes abstractions

Some Key Activities

- Reasoning
  - With induction (type structure, usually)
  - With equations (program structure, usually)
  - With invariants (data structures)
- Analysis
  - Work: sequential complexity
  - Span: parallel complexity
  - Cost graph / Brent's Theorem
- Transformation
  - Creating abstractions
  - Restaging computation
  - Exploiting invariants

Some False Dichotomies

- Sequential vs. Parallel Computation
- Effect-Free vs. Imperative Computation
- Persistent vs. Ephemeral Data Structures
- Continuations vs. Exceptions

Important Algorithms and Data Structures

- Sorting (work and span)
- Search (abstracting control)
- Regular expressions (termination & induction)
- Game tree search (refactoring, exploiting invariants)
- Sequences (fork/join parallelism)
- Streams (laziness, benign effects)

Code is Art

- Poetry — Red/Black Trees, Brzozowski Derivatives
- Essays — 2-Player Games
- Novels — Zoom (unfinished)
- Series — Operating System

Advice for the Final

- Review and organize slides, lecture notes, homework assignments
- Work practice final and lab problems
- Read the questions carefully
- Read the questions carefully

What's Next?

- 15-210: Parallel data structures and algorithms
- 15-312: Principles of Programming Languages
- 15-317: Constructive Logic
- 15-411: Compiler Design
- 15-451: Algorithms
- 15-814: Types and Programming Languages
- 80-413: Category Theory

Be Well!