Today:

- Welcome to CSci 5980/8980: Advanced Functional Programming
- Discuss syllabus and course logistics.
- Course introductions
  - to each other
  - to potential course topics

Lecture Format

- The nature of seminar courses ...
- There are no lectures, only discussions.
- Come prepared to participate.
- This will often require you to read material before class and to submit questions that you have about the material.

Laptops

- Laptops can be a distraction to you and to those seated near you.
- But, they are allowed if you find them useful in class.

Introductions

- Name
- Experience with functional programming
  - CSCi 1901 or 2041, other courses?
  - Extracurricular functional programming?
- Why are you here?
- What do you expect to find interesting?

Course topics

We will cover some, but not all of the following.

Please email me your thoughts on these topics.
Foundations
The lambda calculus

- conversion - $\beta, \alpha, \eta$
- encodings - Church numerals
- combinatory logic
- reduction strategies, evaluation strategies
- typing and types at a foundational level
  - simply typed lambda calculus

Advanced type systems
Much interesting work in type systems takes place in functional languages.

- type inference, Hindley-Milner
- implementing type systems
  - building a type inference system
- value and monomorphism restrictions
- Type classes
- GADTs - Generalized Algebraic Data Types
- Type families

Haskell has been a playground for some of this work recently.

Lazy Evaluation

- applications of it
- advantages, disadvantages

Efficient Data Structures

- Okasaki’s book
- in eager and lazy settings

Monadic Programming

- monads in Haskell for I/O
- monads as a design pattern
- monad transformers
Program verification in Coq

- Functional programs are “easier to reason about”
- Coq is a proof-assistant.
  - We can use it in proving properties of our programs.
- We will “mechanize” some of the proofs that we did in CSci 2041.

Dependent Types

- Add value expressions to the language of types.
- *e.g.* A polymorphic vector type that includes its length.
- programming in Coq or Idris

Functional Reactive Programming

- used in many reactive systems
- Yampa, a Haskell library for mobile robots
- Elm, a language for web GUIs
- this is basis for Erik Meijer’s Reactive Extensions (RX)

Parallel Programming

- nested data parallelism, as in NESL and Haskell
- 2 other varieties of parallelism in Haskell

Combinator Libraries - Embedded DSLs

Examples include

- Parser Combinators
- Sweirstra’s Pretty Printing algorithm

Implementing Functional Languages

- build a SECD machine implementation
- Categorical abstract machine
- Using continuations
- For lazy languages, graph reduction
  - G-machine, Three Instruction Machine