Course Introduction

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science
Table of Contents

Introduction and Logistics

Objectives

Course

Course Overview

Haskell

Picking a Language

Bibliography
Welcome to CS 421!

Topics for discussion:

► Logisitics — instructor, grades, course objectives, lecture format
► What is a Language? — Models of computation, REPL
► Haskell
► How to succeed in this class.
Me!

Name  Mattox Beckman

History  PhD, Fall 2003, University of Illinois at Urbana-Champaign  
          Lecturer 2013–2015 Illinois Institute of Technology

Research Areas  Programming Languages, Mathematical Foundations of Computer Science, CS Education

Specialty  Partial Evaluation, Functional Programming

Professional Interests  Teaching; Partial Evaluation; Interpreters; Functional Programming; Semantics and Types; Category Theory

Personal Interests  Cooking; Go (Baduk, Wei-Qi, Igo); Theology and Philosophy; Evolution; Meditation; Kerbal Space Program; Home-brewing; ... and many many more ...
My Responsibilities

My job is to provide an “optimal learning environment”.

- Assignments will be clearly written and administered.
- Questions will be answered in a timely fashion.
- Objectives of lectures and assignments will be clearly communicated.
- Grades will be fair, meaningful, and reflect mastery of course material.
- If something’s not going the way it should, tell me!
Your Responsibilities

- Check the course web page frequently.
- Subscribe to Piazza and have at least digest email.
- Do the homework assignments in order to learn them.
- Attend lectures if at all possible.
- **Take responsibility and initiative in learning material** — experiment!

You are the one primarily responsible for your education.
Lectures
Speaking of lectures...

- The lecture is ancient technology; invented before the printing press.
- What usually happens during a lecture?
Attention vs. Time

![Graph showing the relationship between Retention and Time with a note saying "Next time..." at 10 minutes.]
Some observations about learning

- Traditional lectures are hard because:
  1. You have to be there at a certain time.
  2. ...and you have to be awake.
  3. ...and you can’t “rewind” if you miss something.
  4. But at least you can ask questions! (If you’re not shy.)

- Homeworks are hard because:
  1. What seemed obvious in lecture is not obvious later.
  2. You can’t ask the professor for help until office hours (or until (if) they check their email).
  3. The one time you see the professor is during lecture, and then they are busy lecturing.

- Proposal: we’re doing this backwards. Let’s do it the right way instead.
Reverse Lectures
The Right Way

- Lectures will be screen-cast and made available on the course website.
  - Usually 2–3 mini-lectures, about 10–20 minutes each.
  - Viewable on mobile devices.
  - Hard part: you do need to see them before the corresponding class period.

- During class:
  - Review time: “Any questions about the lectures?”
  - Activity. Work in groups of 2–3 people, reinforce lecture concepts, prepare you for exams. Activities are collectively worth 5% of your grade.

- This method is not common, but has been thoroughly tested, and it works.
Contact Info

Instructor  Mattox Beckman

Best Contact  via email. I use inbox zero, but not on weekends.

Email Addresses  <mattox@illinois.edu>.

Office  2227 Siebel Center

Office Hours  11:00–12:00 T,W; by my office (2227 Siebel)

Teaching Assistants  We’ve got five, plus four course assistants!
Machine Problems

- Machine Problems — collectively worth 15%
- Designed to help you study for the exams, and to achieve major course objectives.
- Full collaboration allowed for the programming part, but **you must cite your sources!**
- There will be a followup Machine Lab for many of these.
- Don’t use the “perturbation method” of solving machine problems! We expect you to *understand* the solution and the process very well.
- Expect eight assignments, and expect this number to change.
Machine Labs

- Collectively worth 15%
- There are five of them.
- We give you an MP solution with some interesting bits deleted.
  - We may give you something similar to your MP....
- You get one hour to complete the solution.
- Idea: testing if you learned the material on the MP.
Exams/Quizzes

- The purpose of an exam is to measure mastery of material.
  - We’ll use the computer based testing center.
  - Exams are subdivided into proficiency units.
  - The final exam will retest many of the proficiency units. If you improve your score, we update your midterm score with it!
- Two midterms: 20% each.
- Final exam: 25%
Grade Guarantees

99  A+
93  A
90  A-
87  B+
83  B
80  B-
77  C+
73  C
70  C-
60  D
<60  F

We will take the average of the top 5% undergraduate scores, and then take a percentage of that to determine the grade cutoffs.
Table of Contents

Introduction and Logistics
  Objectives

Course
  Course Overview

Haskell
  Picking a Language

Bibliography
Why study languages?

- *pai sei*
- Blub — see *Beating the Averages* by Paul Graham.
- Language Families
Themes

The Big Idea

A Programming Language is an Implementation of a Model of Computation

The course has three major themes:

1. Languages
   What is a language? What kinds of things can we say in a language?
   This covers a lot of areas.

2. Parsing
   How do we get the computer to read what we said?

3. Interpreting and Compiling
   How do we get the computer to do what we said?
Four Fundamental Models

A programming language is a model of computation.

Models

- von Neumann Machine
- Lambda Calculus (or term rewriting)
- Message Passing
- Unification

We’ll go over these in more detail as the course progresses.
So, what should you learn?

- Understand major classes of programming languages: techniques, features, styles.
- How to select an appropriate language for a given task.
- How to read a formal specification of a language and implement it.
- How to write a formal specification of a language.
- Four Powerful Ideas:
  1. Recursion
  2. Abstraction
  3. Transformation
  4. Unification

The emphasis is on learning the theory, knowing why the theory is valuable, and using it to implement a language.
How am I going to learn it?

There are two common approaches to teaching a PL course.

- **Approach 1: “Language of the Month Club”**
  - Lots of time spent on syntax, fundamentals tend to get lost.
  - You’ll forget them all anyway.

- **Approach 2: “Host Language”**
  - Learn one language, use it to write interpreters for all the other languages.
  - You actually get to see how a language is put together.
Table of Contents

Introduction and Logistics
  Objectives

Course
  Course Overview

Haskell
  Picking a Language

Bibliography
How to Pick an Implementation Language

- You all know a lot about Imperative/OO languages.
- Few or none of you know anything about functional languages.
- Functional languages are becoming increasingly important:
  - Roughly four times the programmer productivity.
  - Parallel computation
- Our main language is Haskell.
Features of Haskell

- It's an advanced higher order functional language.
- Has a very modern, concise syntax.
- Has automatic type inference with parametric polymorphism.
- Used a lot in research, and extremely influential in the development of other languages.
- Extremely well suited for writing languages.
Demo of Haskell

You should have a Project Euler activity handout. Try it now, and in a few minutes, the instructor will show you how to solve these in Haskell.
Table of Contents

Introduction and Logistics
  Objectives

Course
  Course Overview

Haskell
  Picking a Language

Bibliography
John Backus.
Can programming be liberated from the von neumann style? a functional style and its algebra of programs.
1997.
ACM ID: 1283933.

Paul Graham.
Beating the averages, April 2003.

George Orwell.
Politics and the english language.
*Horizon, 13(76):252—265, April 1946.*