CS 152: Programming Languages
Course Information and Syllabus
Spring 2011

Logistics and Contact Information: The instructor is Dan Grossman. See the course home page (`http://www.seas.harvard.edu/courses/cs152/2011sp/`) for pertinent information not in this document, particularly for policies regarding assignments.

Goals: We will investigate concepts essential to programming languages including control flow, assignment, scope, functions, types, objects, and threads. We will explore several other related concepts such as continuations, exceptions, monads, message-passing, and connections to logic. Our primary intellectual tools will be *operational semantics* and *Caml programs*. Prior knowledge of neither is required. We will build formal models and prove properties about them, which is a useful skill throughout computer science and fundamental to understanding the foundations of the field.

Successful course participants will learn to:

- Give precise definitions to programming-language features
- Prove properties of inductively defined sets (e.g., well-typed programs)
- Appreciate some programming-language theory jargon (e.g., inference rules) and find the research literature more approachable
- Write better programs by exploiting modern language features such as higher-order functions and objects

In short, our goal is to use theory to make us better programmers and computer scientists. Always think, "how is this related to programs I have written?" We will *not* learn lots of different programming languages or arcane features of particular languages.

Pre-requisites: CS51 is a prerequisite. Students who used OCaml in CS51 may have a slight advantage at the very beginning of the course, but others are more than welcome. Other prerequisites include good programming skills and comfort with recursion, induction, logic, and mathematical notation, though again very little specific knowledge will be assumed. In this vein, CS121 is recommended but not required.

Format: Two weekly lectures will develop the course content. There is no required textbook. Class materials will be posted on the course website, which will also point to books and other resources that can provide excellent "second explanations" for various course topics. The instructor hopes to write lecture notes for the course but may not be able to do so for all topics.

Assignments will extend the material discussed in lecture; expect to learn as you do them. Programming exercises must be done in the Caml language, which will be discussed in class. Exams will assess understanding of the topics covered in lecture and homework.

Assignments, Exams, and Grading: The number of homeworks and grading distribution is *subject to change*, but the intention is roughly as follows:

- Six assignments contributing equally to 45% of the course grade
- One research-article summary (details later), 10% of the course grade
- One in-class midterm contributing, 20% of the course grade
- One final exam, 25% of the course grade

Advice: It is likely you can pass this course without appreciating what we are doing or retaining much of what you learned. Doing so is a poor use of your time. To "appreciate" and "retain" it is best to:

- Determine how each lecture topic and homework problem fits into the course outline and course goals.
- Master the details of the early topics in the course. The material builds on early material more than it seems.

Probable Topics: (subject to change)

- Introduction to Caml
- Inductive definitions
- Operational semantics
- "Pseudodenotational" semantics
- Assignment and basic control flow
- Scope and variable binding
- Lambda calculus
- First-class continuations
- Continuation passing style and the CPS transform
- Simple types
- Type safety
- Curry-Howard isomorphism
- Universal and existential types
- Type-and-effect systems
- Shared-memory concurrency (threads, locks, transactions, memory-consistency models)
- Message-passing concurrency; Concurrent ML
- Object-oriented programming
- Multimethods
- Monads/workflows

Probable Nontopics: There are many topics that are appropriate for this course but we simply will not have time for. We will try to give a hint about what we are missing with respect to the following.

- Denotational semantics
- Axiomatic semantics
- Parametricity
- Type inference
- Logical frameworks
- Module systems
- Programming with lazy evaluation
- ...