CS 340: Programming Paradigms and Patterns

Michael Saelee

January 16, 2019

1 Overview

The introductory programming sequence (CS 115 & CS 116 or CS 201) teaches students the basic syntax and semantics of an object-oriented programming language, and how to use the language to solve a range of problems. Due to time and curricular constraints, the sequence focuses on classical *imperative programming* constructs (variables and assignments, statements, loops, etc.), paying little or no attention to the theory and practice of *functional programming*, which has grown increasingly important in today's software development landscape.

The benefits of functional programming are significant in the areas of *reasoning and verification*, due to the lack of side effects and the natural pairing of induction and recursion, *abstraction*, owing to the emphasis on functions that operate on compound data as a whole instead of iteration (e.g., `map` and `fold`), and *concurrency*, due to referential transparency and the absence of state mutations.

This course focuses on teaching the functional programming paradigm and related techniques, including but not limited to the use of a strong, sophisticated type system, which is a natural extension of and boon to functional programming and reasoning.

2 Learning Outcomes

After completing the course successfully, students should be able to:

- Write substantial, well-typed programs using a purely functional programming language such as Haskell.

- Apply functional programming techniques such as recursion, higher-order functions, and pattern matching to solve problems and build data structures.
- Define and use types that make use of type classes and polymorphism.
- Use functional constructs such as functors and monads to build powerful, reusable abstractions.
- Apply formal, equational reasoning to software development.
- Identify opportunities for parallelism in code and exploit them by choosing appropriate data structures and function designs.
- Use automated testing tools such as Quickcheck to aid development.

3 Faculty and Staff

- Instructor
  - Michael Saelee
  - Office: SB 226C
  - Hours: WF 11:30-13:30
  - E-mail: saelee@iit.edu
- TA
  - Christopher Sherman
  - E-mail: csherman1@hawk.iit.edu

4 Prerequisites

Student are expected to be familiar with an imperative, statically-typed procedural or object-oriented language, and to have written reasonably sophisticated programs (500+ lines of code) with it. Having completed CS 115/116 or CS 201 fulfills this requirement.

5 References

- Miran Lipovača, **Learn You a Haskell for Great Good!**, No Starch Press, Apr 2011. `http://learnyouahaskell.com/`.
- Graham Hutton, **Programming in Haskell**, Cambridge University Press, 2^{nd} edition, Sep 2016.
- Bryan O'Sullivan, Don Stewart, and John Goerzen, **Real World Haskell**, O'Reilly Media, Dec 2008. `http://book.realworldhaskell.org/`.

6 Grading

Grades in the class are broken down as follows:

60% Machine Problems

20% Midterm Exam

20% Final Exam

The grade scale is: $\{\mathrm{A} \geq 90\%; \mathrm{B} \geq 80\%; \mathrm{C} \geq 70\%; \mathrm{D} \geq 60\%; \mathrm{E} < 60\%\}$.

There will be approximately 5-7 individual programming assignments (a.k.a. "machine problems"). Exams will be comprehensive; use of computing devices of any kind (laptops, tablets, smartphones/watches) during an exam is *not* permitted.

7 Detailed topics

- Functional programming overview
- Type systems and Static verification
- Partial evaluation and Currying
- Recursion
- Quickcheck
- Lazy evaluation

- Higher-order functions
- Algebraic data types, Type classes, and Polymorphism
- Persistent data structures
- Equational reasoning and Induction
- Functors, applicative functors, and monads
- Foldables
- Managing and isolating side effects
- Concurrency and Software transactional memory

8 Academic Integrity

You are welcome to discuss assignments with classmates, but all final work must be your own. For details on what constitutes academic dishonesty, consult the university's Code of Academic Honesty at `https://web.iit.edu/student-affairs/handbook/fine-print/code-academic-honesty`. Any confirmed cases of academic dishonesty will be reported to academichonesty@iit.edu, and any work involved will, at the very least, will receive a reduction in grade deemed appropriate by me.

9 Disability Accommodations

Reasonable accommodations will be made for students with documented disabilities. In order to receive accommodations, students must obtain a letter of accommodation from the Center for Disability Resources and make an appointment to speak with me as soon as possible. The Center for Disability Resources is located at 3424 S. State Street, Suite 1C3-2, 312-567-5744 or disabilities@iit.edu.