CSI 5137 Software Foundations
(3 hours of class per week, 3 credits, category T,E)

Fall 2018
Course Outline

Mathematical underpinnings of reliable software. Topics include basic concepts of logic, computer-assisted theorem proving and the Coq proof assistant, functional programming, operational semantics, Hoare logic, and static type systems.

Course Objectives:

• To develop a solid theoretical background in the mathematical underpinnings of reliable software, in particular techniques for specifying and reasoning about properties of software.

• To obtain substantial experience using tools for helping validate these kinds of properties, in particular the use of proof assistants to construct rigorous logical arguments.

• To gain a deep understanding of the idea of functional programming, both as a method of programming and as a bridge between programming and logic.

Prerequisites:

• Significant programming experience
• Mathematical sophistication
• Undergraduate functional programming or compiler class
• At least one of:
  – Be enrolled in a computer science or math Ph.D. program
  – Be enrolled in a computer science or math Master’s program with thesis
  – Have a computer science 4-year undergraduate degree
  – Be an undergraduate in computer science or math with special permission to enroll

Professor:

Dr. Amy Felty
SITE 5-068
afelty@uottawa.ca

Time and Place:

• Morisset Hall Room 219
• Tuesday 14:30–16:00
• Friday 16:00–17:30
Office hours:  Tuesday, 9:00–11:00

Textbook:  *Software Foundations*, Volumes 1 and 2, Benjamin C. Pierce et. al.
The version of the textbook used for this course is available online from the course web site.

Other Resources:


Evaluation:

30%  Assignments (approximately weekly)
35%  2 Term Tests
35%  Final Exam

Web Info:


Course Outline:

Note that this schedule is subject to change during the semester.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Textbook Chapter</th>
<th>Vol.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Preface</td>
<td>1</td>
</tr>
<tr>
<td>Functional Programming in Coq</td>
<td>Basics</td>
<td>1</td>
</tr>
<tr>
<td>Proof by Induction</td>
<td>Induction</td>
<td>1</td>
</tr>
<tr>
<td>Working with Structured Data</td>
<td>Lists</td>
<td>1</td>
</tr>
<tr>
<td>Polymorphism and Higher-Order Functions</td>
<td>Poly</td>
<td>1</td>
</tr>
<tr>
<td>More Basic Tactics</td>
<td>Tactics</td>
<td>1</td>
</tr>
<tr>
<td>Logic in Coq</td>
<td>Logic</td>
<td>1</td>
</tr>
<tr>
<td>Inductively Defined Propositions</td>
<td>IndProp</td>
<td>1</td>
</tr>
<tr>
<td>Total and Partial Maps</td>
<td>Maps</td>
<td>1</td>
</tr>
<tr>
<td>Simple Imperative Programs</td>
<td>Imp</td>
<td>1</td>
</tr>
<tr>
<td>Program Equivalence</td>
<td>Equiv</td>
<td>1</td>
</tr>
<tr>
<td>Hoare Logic (Part 1)</td>
<td>Hoare</td>
<td>2</td>
</tr>
<tr>
<td>Hoare Logic as a Logic</td>
<td>HoareAsLogic</td>
<td>2</td>
</tr>
<tr>
<td>to be continued</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>