EXPANDED COURSE DESCRIPTION

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Lassonde School of Engineering
Electrical Engineering Computer Science

LE / EECS 2001 3.0 SECTION E
INTRO. TO THE THEORY OF COMPUTATION
FALL 2017 / WINTER 2018

Last Modified Date: 08/18/2017

COURSE CALENDAR DESCRIPTION

Introduction to the theory of computing, including automata theory, formal languages and Turing machines; theoretical models and their applications in various fields of computer science. The emphasis is on practical applications of the theory and concepts rather than formal rigour. Prerequisites: General Prerequisite; LE/EECS 1021 3.00 or LE/EECS 1022 3.00 or LE/EECS 1720 3.00 or LE/EECS 1030 3.00; LE/EECS 1028 3.00 or SC/MATH 1028 3.00 or LE/EECS 1019 3.00 or SC/MATH 1019 3.00. Course credit exclusions: LE/SC/CSE 2001 3.00. (NOTE: The General Prerequisite is a cumulative GPA of 4.50 or better over all major EECS courses. EECS courses with the second digit “5” are not major courses.) (NOTE: The General Prerequisite is a cumulative GPA of 4.50 or better over all major EECS courses. EECS courses with the second digit “5” are not major courses.)

INSTRUCTOR(S)

<table>
<thead>
<tr>
<th>Name</th>
<th>Section / Format / Term</th>
<th>Contact Email</th>
<th>Contact Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edmonds, Jeffrey A</td>
<td>Sec. E / LECT / F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOPICS AND CONCEPTS

This course introduces different theoretical models of computers and studies, their capabilities and theoretical limitations. Topics covered typically include the following:

- Finite automata and regular expressions; practical applications, e.g. text editors
- Pushdown automata and context-free grammars; practical applications, e.g. parsing and compilers
- Turing machines as a general model of computers
- Introduction to unsolvability and the halting problem

LIST OF LEARNING OUTCOMES AND EXAMPLES OF

Learning Outcomes

In this course, you will be invited to develop your ability to think clearly and carefully about computing, and to improve your skills in expressing those thoughts about computing in a precise way. By the end of this course, you will be able to do the following things.

**Design machines (e.g., finite automata, Turing machines) to solve specified problems.**
- The central goal of computer science is the design of algorithms to solve computational problems. The ‘machines’ that we describe in this course are not physical machines; they are just ways to formally describe algorithms. We will also talk a bit about how to specify problems formally.

**Design regular expressions and context-free grammars for a given formal language**
- A formal language is just a set of strings. We will see that they can be used to give formal specifications of computational problems. This part of the course looks at some ways to describe these languages in a compact way. Regular expressions and grammars are also useful in the design of programming languages and compilers, and in natural language understanding.
Explain why an object designed according to one of the above two bullets correctly meets its specification

- Computer scientists should be able to design solutions for computational problems. But they must also be able to explain their solution to others. This goal of the course will help you develop your skills in doing this. You will find that proving that your solution is correct also helps you find mistakes in your solution.

Prove simple properties about models of computation (e.g., that the class of regular languages is closed under complement)

- Here, we start to develop an overall theory of computing. Rather than thinking about one problem at a time, we look at the class of problems that can be solved by certain kinds of algorithms and prove properties about that class. This gives us deeper insight into how computers can solve problems.

Demonstrate limits of computing by proving that a problem is not solvable within a particular model of computation

- This is one of the most important components of the course. It's important for a computer scientist to know the limitations of computers, to avoid wasting time on trying to solve a task that turns out to be impossible.

Show how one problem can be reduced to another

- This is a skill that computer scientists use all the time. Instead of solving a problem from scratch, it helps if you can relate it to another problem that has already been solved, and then use the solution to that second problem to solve the original one.

**GRADED ASSESSMENT**

**Grading:**
- 4 Take Home Tests: 5% * 4 = 20%
- Midterm: 25%
- Class Participation: 10%
- Exam: 45%

**Tests:**
- **Partner:** For each "Take Home Test", you may either work by yourself or with a partner. Both partners should work on every problem, since these tests are intended to help you learn the material and receive feedback on your work.
- **No Notes:** As with any test, after studying, you are to write it without looking at any notes or talking to anyone other than your partner.
- **Time:** You may have as much time as you like.
- **Rewrite:** If you get stuck, you can stop the test. You can then read anything and talk to anyone you like. I encourage you to come to office hours. When you feel you are ready you may then restart the test. You must set aside any notes and wait one hour to clear your mind before restarting the test.
- **Due Date:** The test must be handed in on the Monday at midnight: Oct 2, Oct 16, Nov 13, and Nov 27.

**ADDITIONAL INFORMATION**

**Recommended Textbook**

Michael Sipser. *Introduction to the Theory of Computation, Third Edition*. Course Technology, 2012. Errata. (If you have an older edition, it will be fine too.) Note that the bookstore offers temporary access to an electronic version of the book more cheaply than the price of a hard copy. (However, if you plan to take EECS4115, some instructors use the same textbook for that course.)

**Other References**

If you used Rosen's book, *Discrete Mathematics and its Applications*, for EECS 1019, it has a chapter on the topics of this course with lots of exercises. (It is chapter 12 in the 6th edition.) This book is available on reserve at the library.
The following list gives other useful references.


ACADEMIC INTEGRITY LINKS

- Senate Policy on Academic Honesty - http://secretariat-policies.info.yorku.ca/policies/academic-honesty-senate-policy-on/
- Academic Integrity - http://lassonde.yorku.ca/academic-integrity

STUDENT LINKS

- Student Rights and Responsibilities - http://oscr.students.uit.yorku.ca/student-conduct
- Religious Observance - https://w2prod.sis.yorku.ca/Apps/WebObjects/cdm.woa/wa/regobs
- Counselling and Disability Services - http://cds.info.yorku.ca/

Many courses utilize Moodle, York University’s course website system. If your course is using Moodle, click here to access it.

Moodle @ York University