CSCI 2670
Introduction to Theory of Computing

Fall 2014

Instructor and Lecture Times

- **Instructor**: Don Potter (potter@uga.edu)
- **Office**: Boyd GSRC 111;
- **Office Hours**: TBA
- **Lecture Times and Locations**

9:05-9:55	Mon	Life Sciences, C114
9:30-10:45	Tues & Thurs	Life Sciences, C114

Brief Description

This is a first course on the theory of computing.
Fundamental topics include: finite automata, regular expressions and languages, context-free grammars and languages, push-down automata, pumping lemmas for regular languages and for context-free grammars, the Chomsky hierarchy of language classes, Turing machines and computability, undecidability of the halting problem, reducibilities among decision problems and languages, time complexity, and NP-completeness and tractability.

Prerequisites: CSCI/MATH 2610: Discrete Mathematics; or CSCI 2611: Discrete Mathematics for Engineers.

Informative Description

- The course focuses on questions of the following sort:
  - *Which problems are solvable by computers and which aren't?*
  - *For a given solvable problem, how difficult is it to solve?*
- Questions like the first are addressed in *computability theory*. Questions like the second are addressed in *complexity theory*.
  - Historically, computability theory was developed before/independently of actual computers (1920s-1930s).
  - Complexity theory came later (really taking off in the 1970s).
- In order to answer these questions, we must first define what "problem" and "computer" mean.

Models of Computation: Automata

- This course presents several types of abstract machines (*automata*).
- They can be viewed as having different amounts of memory.
  - Finite State Machines: constant memory.
    - Deterministic Finite Automata (DFAs)
    - Nondeterministic Finite Automata (NFAs)
  - Push Down Automata (PDAs), having a stack
  - Turing Machines, having an unlimited tape
- The machine types are of different computational "power" (with the TM being the most powerful of all).
- "Computation" is defined via the activity of these machines.

Formal Languages

- The machines are closely related to types of *formal languages* (generated by formal grammars).
  - A formal language L is just a set of strings defined over an alphabet of symbols. $\{a, aa, aaa, aaaa, \ldots\}$ is a language.
- Each machine M has a formal language $L(M)$ associated with it, the language it "**accepts**" or "**recognizes**".
  - Roughly, M accepts a string if it stops after reading the string and says "YES". $L(M)$ is the set of strings M accepts.
- A fundamental type of problem: Given a language L, create a machine that accepts it.
- Related problem: Given L, create a machine that can **decide** whether or not an arbitrary string is in L.

Decidable and Undecidable Languages

- Note that these problems can not always be solved.
- Example Decidable Language: The language consisting of the tautologies of propositional logic.
- Example Undecidable Language: The language consisting of the tautologies of predicate logic.

More generally, determining whether a set of formulas of propositional logic is satisfiable is a decidable problem, while the same problem for predicate logic is not.

Formal Grammars

- Formal languages are also associated with **formal grammars**.
  - A grammar is a set of rules for generating strings over a given alphabet.
- E.g., the grammar below produces $\{b, ab, aab, aaab, \ldots\}$.

$$S \to aS \text{ (S may be replaced with } aS)$$
$$S \to b \text{ (S may be replaced with } b)$$

- Similar to machines, associated with each grammar G will be a language $L(G)$.
- Formal grammars can be viewed as another model of computation.
- Problem: Given language L, create a grammar G such that $L(G) = L$.

The Chomsky Hierarchy

- There is an equivalence between the abstract machines and types of grammars.

Language/Grammar	**Automaton**
Unrestricted	Unrestricted Turing Machine
Context Sensitive	Linearly Bounded Turing Machine
Context Free	Push Down Automaton (PDA)
Regular	Finite State Machine (DFA and NFA)

- The languages here form a hierarchy. Each level is contained in the one above it.
- Machines lower in the hierarchy are weaker than those higher:
  - A non-regular, context free language cannot be recognized by any finite state machine.
  - But every regular language can be recognized by some PDA.

Quick History

- 1936 Alan Turing proposes the Turing machine, and proves that there exists an unsolvable problem.
- 1943 McCulloch and Pitts develop finite automata.
- 1940's von Neumann proposes the Stored Program Concept.
- 1956 Kleene develops regular expressions and proves the equivalance of regular expressions and finite automata. Also, Chomsky defines the Chomsky Hierarchy, which organizes languages recognized by different automata into hierarchical classes.
- 1959 Rabin and Scott introduce nondeterministic finite automata and prove equivalence to deterministic finite automata.
- 1965 Hartmanis and Stearns define time complexity, and with Lewis they define space complexity.
- 1971 Cook shows the first NP-Complete problem, the satisfiability problem.

Course Objectives

- By the end of the semester, students should be able to:
  - Given an NFA M, create a DFA or a regular expression that accepts $L(M)$.
  - Given a regular language L, create an NFA accepting it.
  - Use pumping lemmas to prove a language is not regular or not context-free.
  - Given a description of a context-free language L, develop a context-free grammar (CFG) G such that $L(G) = L$.
  - Convert a CFG into Chomsky Normal Form (CNF).
  - Given a CFG G in CNF and a string w, use the CYK algorithm to determine if G generates w.
  - Given an context-free grammar G, create a push-down automaton (PDA) that accepts $L(G)$.
  - Identify if a given language is regular, context-free but not regular, or neither.

Course Objectives

- By the end of the semester, students should be able to:
  - Given a language L, create a Turing machine TM that accepts L.
  - Convert between different variations of the Turing machine model (e.g., multi-tape to single tape).
  - Create a Turing machine that performs a function.
  - Define decidabilty and demonstrate that a language is decidable.
  - Reduce one problem to another one.
  - Use reductions to prove a problem is undecidable.
  - Define P, NP and NP-complete.
  - Show a problem is in P and determine its computational complexity.
  - Write pseudo-code describing a non-deterministic Turing machine's steps to solve a problem.
  - Prove a problem is NP-complete.

Textbook and Website

- **Required Text**: Michael Sipser. Introduction to the Theory of Computation (3rd Edition). ISBN-13: 978-1133187790
- **Course Website**: Certain course materials, maybe student grades, might be posted to the course's eLearning Commons (eLC) website:

  `https://www.elc.uga.edu/`.

  Secondary websites might also be used.
- **The Wiki**: There might be a wiki, where course materials will be posted. Users can also post links to useful resources there.

  `TBA`

  Students will each be given login details if we use the wiki.

Lectures and Homework

- **Lectures**: Cover Chapters 0–7, excluding Chapter 6.

  Readings will be assigned beforehand, and students hopefully read these before class begins.

  Quizzes and other activities will take place during lecture periods.

- **Homework**: Students must typeset and submit printed copies of HW solutions on the assigned due date.

  Late submissions are generally not accepted.

  At the end of the semester, the lowest HW grade will be dropped.

  You are encouraged to use LaTeX or some other appropriate formatting system for the HWs.

Exams

3 in-class exams and a separate final exam. Tentative dates:

In-class Exam 1:	September 18th.
In-class Exam 2:	October 21st.
In-class Exam 3:	November 20th.
Final Exam:	December 11th, 2014, 8:00–11:00am (tentative)

Makeup tests will not be given. However, a missed in-class exam *might* be excused. If excused, the average of the other in-class exams and the final exam will be used instead.

Grading

Exam 1 (16%), **Exam 2** (16%), **Exam 3** (16%),
Final Exam (25%), **Homework** (22%), **Quizzes/other work** (5%).

Final Letter Grades

$A \geq 93$	$80 > C+ \geq 77$
$93 > A- \geq 90$	$77 > C \geq 73$
$90 > B+ \geq 87$	$73 > C- \geq 70$
$87 > B \geq 83$	$70 > D \geq 60$
$83 > B- \geq 80$	$F < 60$

The instructor reserves the right to curve grades. Under no circumstances will this lower a student's grade.

Regrading

With the exception of the final exam, students may request a reevaluation of any test, quiz, or graded assignment. In order to be considered, however, the request must be made no more than 7 days after the graded material has been returned to the class.

Other

- **Electronic Devices** You likely will not want to use them unless necessary.
- **Email** Use UGA email in communications. Include "CSCI 2670" in the subject line.
- **In-Class and On-line Behavior** Be courteous and respectful.

Academic Honesty

- As a UGA student, you have agreed to abide by the University's academic honesty policy, "A Culture of Honesty," and the Student Honor Code.
- All academic work must meet the standards described in "A Culture of Honesty" found at: `http://honesty.uga.edu/`.
- Lack of knowledge of the academic honesty policy is not a reasonable explanation for a violation.
- Questions related to course assignments and the academic honesty policy should be directed to the instructor.

Academic Honesty

Common forms of academic dishonesty:

- copying from another student's test paper or laboratory report, or allowing another student to copy from you;
- fabricating data (computer, statistical) for an assignment;
- helping another student to write a laboratory report or computer software code that the student will present as his own work, or accepting such help and presenting the work as your own;
- turning in material from a public source such as a book or the Internet as your own work.

For you to do this week...

- ☐ Try to access the course on eLC, and your other courses.
- ☐ Review the syllabus. A sheet stating you have read it will be circulated in class for you to sign.
- ☐ Read Chapter 0 of the textbook.
- ☐ Download a LaTeX distribution (if you want; such as TexStudio).

Tomorrow: We begin Chapter 0.