CS311
Computational Structures

James Hook
Portland State University
(Some class materials developed jointly with Professor Tim Sheard)

Syllabus and Class Preliminaries

Registration Details

CS 311 Computational Structures
Credits: 4 credits
Instructor: James Hook
CRN: 10998
Meeting times: Tue & Thur 14:00-15:50.
Meeting room: NH 237

Contact Details:

- James Hook
  Office: Engineering Building (EB) 502
  Telephone 503-725-5166
  email james.hook@pdx.edu
- Office Hours: Wednesdays, 3 – 5pm (arrive before 4:30), or by appointment
- My office is in the “Dean’s office suite” in EB 500
- Please come to my door to see if I’m available; if I’m not available please queue at the table near the entrance

Teaching assistant:

- Nate Launchbury
- Office hours: TBA

Exams

- Midterm:
  - Thursday, October 27, 2016 (in class)
- Final:
  - Monday, December 5, 2016. 10:15am to 12:05pm.
    - The University scheduled final exam period is not the same as normal class hours!

Methods of assessment:

Class Exercises (d2l record)	10%
Problem Sets (8 weekly homeworks)*	40%
Midterm	20%
Final exam	30%
TOTAL	100%

* = Subject to revision if impractical; working on grader coverage

Exercises

- Exercises are short (less than 1 hour) that are meant to make you think. I will use your answers to adjust lecture emphasis.
- Exercises are assigned on Tuesday and are due via D2L by class time on Thursday.
  - I frequently grade these at breakfast on lecture day
- Each exercise is worth 1 point.
  - You get full credit for making a good-faith effort to answer the questions.
  - You get 0 points otherwise.

Turning in Problem Sets

- You turn in homework via D2L. You may do it by hand, and then scan your assignment into a pdf, or you may use some tool to typeset your homework. Please submit a pdf file.
- In some cases we may elect to only grade some of the problems in a problem set. I will prioritize quality, timely responses over complete coverage of the problem set.

Policies:

- By default, all deadlines are firm.
- I will be as flexible as possible in accommodating special circumstances; but advance notice will make this a lot easier.

Academic Integrity

- We follow the standard PSU guidelines for academic integrity Students are expected to be honest in their academic dealings.
- Collaboration Policy
  - Unless explicitly instructed otherwise, please hand in solutions that you prepared individually without directly consulting other sources or notes.
  - Never represent the work of others as your own work.
  - (See amplifying comments on following slides)

Collaboration Policy (cont)

You may meet with other students to discuss homework problems, but please discard all notes from these sessions.

- Do not consult notes from discussions with other students or other solutions when preparing your solution.
- Do not provide other students with access to your solution.

Collaboration Policy (cont)

- If you require resources other than the book to solve a problem, please identify those resources with proper citations (but, as for collaborations, set the source aside and do not consult it directly when preparing your solution).
- When selecting other resources, give priority to original sources, texts, and lecture notes.
- Do not consult sample solutions specific to the problems assigned.

Collaboration Policy (cont)

- No exam problems are to be discussed until all students have handed in their exams.
- Students are responsible to keep their exam answers to themselves. Allowing a solution to be copied is as serious a breach of academic integrity as copying.

Access and Inclusion for Students with Disabilities (1)

- PSU values diversity and inclusion; we are committed to fostering mutual respect and full participation for all students. My goal is to create a learning environment that is equitable, useable, inclusive, and welcoming. If any aspects of instruction or course design result in barriers to your inclusion or learning, please notify me. The Disability Resource Center (DRC) provides reasonable accommodations for students who encounter barriers in the learning environment.

Access and Inclusion for Students with Disabilities (2)

- If you have, or think you may have, a disability that may affect your work in this class and feel you need accommodations, contact the Disability Resource Center to schedule an appointment and initiate a conversation about reasonable accommodations. The DRC is located in 116 Smith Memorial Student Union, 503-725-4150, drc@pdx.edu, https://www.pdx.edu/drc.
- If you already have accommodations, please contact me to make sure that I have received a faculty notification letter and discuss your accommodations.

Access and Inclusion for Students with Disabilities (3)

- Students who need accommodations for tests and quizzes are expected to schedule their tests to overlap with the time the class is taking the test.
- Please be aware that the accessible tables or chairs in the room should remain available for students who find that standard classroom seating is not useable.
- For information about emergency preparedness, please go to the Fire and Life Safety webpage (https://www.pdx.edu/environmental-health-safety/fire-and-life-safety) for information.

Course Text:

Introduction to the Theory of Computation

(3rd edition)

Michael Sipser

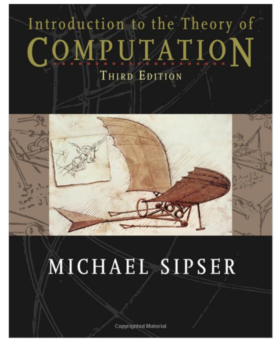
ISBN-13 978-1-133-18779-0

Home page of the text book:

http://www-math.mit.edu/~sipser/book.html

Note: If you have an earlier edition or an international edition you will need to adjust the page numbers, section numbers, and problem numbers. I do not plan to emphasize any of the material added in the third edition.

It looks like this!



Official Course Description

The main goal of the course is that students obtain those skills in the theoretical foundations of computing that are used in the study and practice of computer science. A second goal is that students become familiar with Prolog as an experimental tool for testing properties of computational structures. Upon the successful completion of this course students will be able to:

1. Find regular grammars and context-free grammars for simple languages whose strings are described by given properties.
2. Apply algorithms to: transform regular expressions to NFAs, NFAs to DFAs, and DFAs to minimum-state DFAs; construct regular expressions from NFAs or DFAs; and transform between regular grammars and NFAs.
3. Apply algorithms to transform: between PDAs that accept by final state and those that accept by empty stack; and between context-free grammars and PDAs that accept by empty stack.
4. Describe LL(k) grammars; perform factorization if possible to reduce the size of k; and write recursive descent procedures and parse tables for simple LL(1) grammars.
5. Transform grammars by removing all left recursion and by removing all possible productions that have the empty string on the right side.
6. Apply pumping lemmas to prove that some simple languages are not regular or not context-free.
7. State the Church-Turing Thesis and solve simple problems with each of the following models of computation: Turing machines (single-tape and multi-tape); while-loop programs; partial recursive functions; Markov algorithms; Post algorithms; and Post systems.
8. Describe the concepts of unsolvable and partially solvable; state the halting problem and prove that it is unsolvable and partially solvable; and use diagonalization to prove that the set of total computable functions cannot be enumerated.
9. Describe the hierarchy of languages and give examples of languages at each level that do not belong in a lower level.
10. Describe the complexity classes P, NP, and PSPACE.

My Course Description

The main goal of the course is that students obtain those skills in the theoretical foundations of computing that are used in the study and practice of computer science. ~~A second goal is that students become familiar with Prolog as an experimental tool for testing properties of computational structures~~. Upon the successful completion of this course students will be able to:

1. Find regular grammars and context-free grammars for simple languages whose strings are described by given properties.
2. Apply algorithms to: transform regular expressions to NFAs, NFAs to DFAs, ~~and DFAs to minimum-state DFAs~~; construct regular expressions from NFAs or DFAs~~; and transform between regular grammars and NFAs~~.
3. Apply algorithms to transform: between PDAs that accept by final state and those that accept by empty stack; and between context-free grammars and PDAs that accept by empty stack.
4. ~~Describe LL(k) grammars; perform factorization if possible to reduce the size of k; and write recursive descent procedures and parse tables for simple LL(1) grammars.~~
5. ~~Transform grammars by removing all left recursion and by removing all possible productions that have the empty string on the right side.~~
6. Apply pumping lemmas to prove that some simple languages are not regular or not context-free.
7. State the Church-Turing Thesis and solve simple problems with each of the following models of computation: Turing machines (single-tape and multi-tape); ~~while-loop programs~~; partial recursive functions; ~~Markov algorithms; Post algorithms; and Post systems.~~
8. Describe the concepts of unsolvable and partially solvable; state the halting problem and prove that it is unsolvable and partially solvable; and use diagonalization to prove that the set of total computable functions cannot be enumerated.
9. Describe the hierarchy of languages and give examples of languages at each level that do not belong in a lower level.
10. Describe the complexity classes P, NP, and PSPACE. (If time allows)

Key Techniques

- Proof
  - We will frequently define a computational system and then reason systematically from the definitions
- Induction
  - Reason about infinite sets where all the elements are finitely constructed
- Reduction
  - Use functions to map one description of a computational system to another
- Diagonalization
  - A technique to build a sequence not present in a countable set of sequences (First used by Cantor to show there are more real numbers than natural numbers)

Key Ideas

- All Computational Systems have finite descriptions
  - We can think of these descriptions as "programs"
- We can characterize computational problems as "languages"
  - Computational Models give a vocabulary for comparing problems precisely
- Not all mathematical functions are computable
  - Some of the non-computable functions would be very helpful if they existed