Syllabus for CECS 329, Concepts of Computer Science Theory

Instructor: Dr. Todd Ebert

Spring 2024, Last Updated March 21st, 2024

General Course Information

Academic Unit Department of Computer Engineering and Computer Science, California State University, Long Beach

Prerequisite CECS 328 with a grade of "C" or better.

Catalog Description Seminar on fundamental topics in theoretical computer science. Topics include regular languages, finite automata, context-free languages, Turing machines, computability theory, computational complexity, and NP-completeness.

Section Call Number 7027 (Section 1)

Instructor Dr. Todd Ebert (Todd.Ebert)

Instructor Office Hours Tuesday, Wednesday, Thursday: 9:30-10:30 am in ECS 548.

Teaching Assistant Abhishek Jajoo

Course Meeting Times TuTh 8:00-9:15 am (ECS Room 105)

Required Textbook M. Sipser, "Theory of Computation", Cengage Learning, 2012, 3rd Edition, 978-1133187790

Optional Textbook N. Cutland, "Computability: An Introduction to Recursive Function Theory", Cambridge Press, 1980, 978-0521294652

Course Website `http://www.csulb.edu/~tebert/teaching/spring24/329/intro.html`

1 Course Structure and Topics

The course is divided into three one-unit sections, each of which spans five weeks and ends with an an exam on the material from that unit.

Unit1: Weeks 1-5. Complexity Theory: Turing and Map Reducibility, Computational Complexity Classes (`P`, `NP`, and co-`NP`), NP-completeness, and NP-completeness proofs.

Unit2: Weeks 6-10. Computability Theory: Unlimited Register Machines, Encoding and Decoding of Programs, Universal Programs, Self-Knowing Programs, Decidability and Undecidability.

Unit3: Weeks 11-15. Automata and Formal Language Theory: Regular Expressions, Context-Free Languages, Deterministic and Nondeterministic Finite Automata, Turing Machines.

Reasons for Studying Computer Science Theory

1. allows one to develop a "big-picture" perspective on computing
2. many employers test for knowledge of theory as a means for identifying the top applicants
3. every computer science discipline has its theoretical aspects whose mastery contributes to a mastery of the discipline
4. theoretical breakthroughs can lead to technological breakthroughs
5. Lewin's maxim: "nothing as practical as a good theory"
6. allows one to better understand both the limitations and possbilities of computers
7. the contemplation and practice of abstract computing ideas can improve one's ability to deconstruct and analyze complex problems and systems
8. prepares students to tackle the abstract math and proofs found in many advanced computer-science papers
9. computer science theory saves time: the theoretical understanding of a data structure, algorithm, or some other computing concept usually requires far less time and effort than implementing it in practice.
10. unlike computer technology which is transient, computer science theory endures
11. studying and advancing computer science theory can seem fun, fascinating, and fullfilling to where some make it their lifelong pursuit!

Reading Assignments

A reading assignment will be provided on most weeks of the semester. Reading the textbook will offer a somewhat alternative and more comprehensive viewpoint of the subject matter. Please check the "Reading Assignments" link for the current and past assignments.

Class Meetings

On Tuesday's a full lecture will be provided. Each Thursday meeting (excluding midterm days) from 8:00-8:20 am students will be given a problem to solve (see Learning Outcomes section). Students may start on the problem as early as 7:45 am. At 8:20 am, all problems will be collected and we'll briefly review the solution before starting on the lecture.

Each lecture is supplemented with notes that are available online in pdf format. These notes have several examples, some of which are not accompanied by a solution, and they will be solved during lecture. The notes also contain all the needed definitions, formulas, theorems, exercises, and exercise solutions. Links to all course notes shall be provided at the course website (see above).

Learning Outcomes

There will be twelve **core learning outcomes (LO's)** that will be assessed during the semester. Each outcome is assessed via a problem that is provided at the beginning of class each Thursday. Each problem shall be graded as Pass (P) or No Pass (NP). For a passing mark, a solution must be **entirely correct**, within reason. For a student to earn a C for the course, it is necessary (and sufficient!) to correctly solve one problem in at least ten of the twelve learning outcomes. If a student does not pass the first time, then there will subsequent opportunities to solve a makeup problem provided on a following week.

The following are some rules and guidelines for solving the learning-outcome problems.

1. When preparing for the learning-outcome problem, carefully read its official description along with the list of lecture-note examples and exercises that pertain to it.
2. The problem will seem similar to one of the practice problems found in the lecture-note examples and/or exercises. It's important to carefully follow the directions and include all the pertinent steps that lead to your solution.
3. All problems are to be solved at the beginning of Thursday's class. The official start time is 8:00 am, but problems will be made available as early as 7:45 am if you wish to have more time.

4. All computing devices (cell phones, laptops, smart watches, etc.), notes, and books must be put away before starting.

5. To help prevent cheating, when possible, please select a seat that leaves an empty seat between you and your neighbor. **Any student suspected of cheating on an LO assessment or exam will receive an F course grade with no possibility of having the grade forgiven**.

6. Each student is allowed to solve AT MOST TWO PROBLEMS in one sitting: either two makeup problems from previous LO's or one makeup problem and the new problem. Submitting more than two solutions will result in the two *worst* submissions being counted.

7. Use a separate sheet for each solution to an LO problem and write on only one side of each sheet if possible. Make sure to write your name and the LO being solved.

8. Please have the courtesy to hand in your solution upon request. You will be issued a warning should you ignore the request and your work will not be graded for all subsequent infractions.

The following list of learning outcomes will be updated throughout the semester.

LO1. The ability to provide the definition of what it means to be a mapping reduction f from some problem A to another problem B, and demonstrate one of the following reductions, and verify that the solution to $x \in A$ the same as the solution to $f(x) \in B$.

(a) `Even` $\leq_m$ `Odd` See Exercise 1 of the Mapping Reducibility lecture for the kind of problem to expect.

(b) `Max Independent Set` $\leq_m$ `Max Clique`

(c) `Set Partition` $\leq_m^p$ `Subset Sum`

(d) `Hamilton Path` $\leq_m^p$ `LPath`

(e) `Subset Sum` $\leq_m^p$ `Set Partition`

(f) `Vertex Cover` $\leq_m^p$ `Half Vertex Cover`

(g) `Clique` $\leq_m^p$ `Half Clique`

References: Sections 2-5 of the "Turing and Mapping Reducibility" Lecture, Exercises 3-13

LO2. The ability to show that a decision problem belongs to complexity class `NP`. **References.** Computational Complexity Lecture: Examples 4.2-4.6, 4.9, Exercises 3-8

LO3. The ability to demonstrate and/or answer questions about one of the following polynomial-time mapping reductions that establishes the NP-completeness of some decisiion problem: 3SAT to Clique, 3SAT to Subset Sum, SAT to 3SAT, 3SAT to DHP, DHP to UHP, and Hamilton Cycle to Traveling Salesperson. References: Complexity Lecture Examples: 6.3, 6.6, 6.8, 6.10, Exercises 9-15, Mapping Reducibility Lecture: Examples 6.3, 7.2, 7.4 Examples 14-19

LO4. A general understanding of the basic concepts of mapping reducibility and complexity theory including the complexity classes `P`, `NP`, and `co-NP` (their definitions and the problems that belong in each class), `NP`-complete problems, how to prove that a problem is `NP`-complete, and the role that polynomial-time mapping reducibility plays in complexity theory. **references:** Mapping Reducibility (Sections 2-7) and Complexity Lectures

LO5. The ability to establish that a function is URM computable by writing a URM program for computing the function). **References:** Section 3 of "Computability Basics" lecture and Exercises 1-3.

LO6. The ability to encode and decode a URM program and to demonstrate how a universal program P_U, on inputs x and y, simulates a single step (i.e. moving from one configuration encoding to the next) of the computation $P_x(y)$. **References:** Sections 5, 6, and 8 of the "Computability Basics" lecture and Exercises 8-11, 14

LO7. An understanding of the use of the diagonalization method for proving the undecidability of a given programming property. In particular, understanding how the method is used for proving the undecidability of both the `Self Acceptance` and `Total` decision problems. The ability to explain why an attempted use of the diagonalization method fails to work in relation to some decision problem. **References.** Sections 3-8 and Exercises 1-4 of "Undecidability and the Diagonalization Method".

LO8. Given a description of some program property A, the ability to determine if some specific program has that property. Furthermore, the ability to prove that A is undecidable by writing a program P that makes use of i) the `self` programming concept and ii) the assumption that $d_A(x)$ is total computable in order to have P behave in a way that is contradictory to whether or not P has property A. Finally, the ability to perform a case-by-case analysis that establishes that P's behavior contradicts $d_A(\gamma(P)) = d_A(\text{self})$. **References:** Examples 4 and 5, and Exercises 2-8 of the Self-Referencing Programs lecture.

LO9. The ability to design and draw the state diagram of a DFA that accepts a given regular language. The ability to provide the computation of the DFA on one or more input words. **References:** Finite Automata Lecture: Examples 1-6. Exercises 1-10.

LO10. The ability to i) provide the δ-transition table for an NFA N that is defined via a state diagram, ii) demonstrate an NFA computation of N on some input word w, and iii) convert N to an equivalent (in terms of the language it accepts) DFA M by associating each state of M with an appropriate subset of states from N. **References:** Finite Automata Lecture: Examples 8-12. Exercises 13-16, 18, 19.

LO11. Given the description of a regular language L, the ability to provide a regular expression E for which $L(E) = L$. Also, given a word $w \in L$, the ability to parse w in relation to the E. References: Finite Automata lecture Examples 23-25, Exercise 28

LO12. TBD

Exams

There will be two midterm exams and one final exam. The purpose of these exams is twofold: i) to provide more opportunities to solve learning-outcome makeup problems, and ii) to earn course points by solving more advanced problems and/or problems involving topics that are not directly represented by any of the twelve learning outcomes. There will be six problems each worth 25 points,

for a total of 150 possible points for each midterm. In addition, a makeup problem will be provided for each past learning outcome. In general, these makeup problems will be worth 0 points unless otherwise indicated. The final exam will be similar to the midterms, with the exception that there will be two problems worth 50 points and four 25-point problems, for a total of 200 points. The guidelines for exams are the same as for solving the learning-outcome problems. However, **each student is allowed to have a maximum of six solutions graded on any exam. This includes solutions to makeup problems**.

Final Grade Determination

At the end of the semester, grades will be assigned according to the number of passed learning outcomes and the total points earned throughout the semester. The following is a list of grades and the conditions that must be met in order to attain at least that grade.

D Pass at least five learning outcomes.

C Pass at least ten learning outcomes.

B Pass at least ten learning outcomes and finish in the second tier of students with respect to total course points earned.

A Pass at least ten learning outcomes and finish in the top tier of students with respect to total course points earned.

In addition to earning points on exams, other ways to earn points in the course are as follows.

1. There will be three writing assignments worth 40 points each.
2. Earn 25 bonus points for each LO passed that goes beyond passing ten LO's.
3. Identify errors (excluding grammatical and spelling) in the lecture presentation (5 pts each, first come first serve, must raise hand during lecture).
4. Identify errors (excluding grammatical and spelling) in the lecture notes that could possibly lead to student misunderstanding (5 pts each, first come first serve, 10 pts for an error found in an exercise solution). Note: this only includes errors in the original notes and not annotation errors made during lecture (those errors should be identified during lecture).
5. Identify an error in the solution to an exam or learning outcome problem: 5 points.
6. Class participation points (5 points per question) for raising your hand and answering a review question or asking an exceptional question.
7. At the end of each exam meeting, any student who has collected all of his/her graded papers will earn 10 points. In other words, at the end of the exam meeting, if we have one of your graded papers in our possession, then you will *not* earn the 10 points.

Exam Dates

Midterm 1 February 22nd

Midterm 2 March 28th

Final Exam Thursday May 16th, 8:00-10:00 am

Cheating

As an undergraduate at CSULB, I had a bad experience with cheating during the final exam for my Combinatorics class. The instructor passed out the exams, then left the room, only to return two hours later. Once he left, some students began working in groups and discussing all the problems. I felt very upset, both with the students and the instructor for not caring if students cheated. Once I became instructor I vowed to do better at upholding the integrity of the class, degree, and university.

Therefore, I take cheating very seriously. When taking an in-class exam or solving a LO problem, all of the following consitute cheating: use of an electronic device, use of prepared notes or book, copying from another student, or communicating with anyone (outside of the instructor and ISA) about the exam or problem being solved. When taking an online exam or solving an LO problem online, all of the following consitute cheating: using any resource outside of the course notes, textbook, and course lectures, copying from another student, or communicating with anyone (outside of the instructor and ISA) about the exam or problem being solved.

If I have strong evidence that suggests you've cheated, you will receive an F for the course and I will send a letter to the university that documents the incident. I will also make it impossible for you to use the university's "grade forgiveness" policy for removing the F from your GPA and transcript. The only exception is in class if I catch your eyes wandering on your neighbor's paper, I will issue you a warning and assign you a permanent seat in the front row. However, the second time I catch you will result in the aforementioned penalties.

During the pandemic I failed over 30 students. I'm not proud of this, but it does show that I care about fairness and upholding the integrity of our education. When you receive credit for anything in this course, I want to make sure it's due to you being able to understand and internalize knowledge taught in the course. In the end, you cheat yourself and your classmates when cheating towards your educational goals.

Tips for Studying

1. Please seek help from the instructor if you find yourself struggling with the material.
2. Outside of the class meetings, study for at least 60-90 minutes, six days per week. Learning computer-science theory is similar to learning how to program or play an instrument. It requires daily practice in order to attain mastery. Studying every day helps break down the resistance to learning new and more complex topics as the semester progresses.
3. Make a study schedule and stick with it. In the evening write down your goals for the next day so that, while sleeping, your brain can work on how to achieve them.
4. Focus on reading the notes and textbook and working homework problems. Your final grade will largely be based on your ability to solve new problems that are similar to ones from the homework.

5. Whenever a problem uses a term that was defined in lecture, force yourself to spend at least one minute recalling the definition before looking it up. This is a very good exercise for your brain and will help you accelerate towards mastery. Remembering formulas, definitions, and examples has a functional effect on the brain that leads to increased skill and creativity towards the subject.

6. Keep a journal with all your attempted problem solutions. Review them and improve them when preparing for quizzes and exams. Learning is more cyclic than linear, in that we benefit from returning to past work and improving it based on our increased level of understanding.

7. Make sure to get 6-8 hours of sleep each night. Much of your learning takes place while you sleep!

8. Stay connected with other students in the class and form study groups.

9. For the sake of additional practice, use your creativity to create problems that are similar to the HW problems.