CSCI 410/CSIS 616 Automata Theory- Course Announcement
Spring 2024

Instructor: Michael Levet; levetm (at) cofc (dot) edu.
Meetings: TBD

Prerequisites: The formal prerequisites include:

- Math 307 (C- or better), OR
- CSCI 310 (C- or better), OR
- Math 295 (C- or better).

Course Description: CSCI 410/CSIS 616 is an advanced undergraduate/introductory graduate course in theoretical computer science, which is divided into three key areas: Automata theory, Computability theory, and Computational Complexity theory. The goal is to ascertain the power and limits of computation. To this end, it is necessary to define precisely what constitutes a model of computation as well as what constitutes a computational problem. This is the purpose of Automata theory. Computability theory studies which problems can (and cannot) be solved by computers. Finally, Computational Complexity theory seeks to classify computational problems based on how efficiently they can be solved.

We will discuss several models of computation and the classes of computational problems they solve, including finite state automata and regular languages, pushdown automata and context-free languages, and Turing machines and recursively enumerable languages. Once the basic framework from automata theory is established, we will move into more interesting material from computability and complexity theory including the undecidability of the halting problem, reducibility, the Church-Turing thesis, and the P vs. NP problem. Should time allow, we may delve further into computability and complexity theory.

Ultimately, this course is mathematical in nature. The obvious connections are with combinatorics and graph theory. However, theoretical computer science also has deep connections to algebra, topology, analysis, and logic. These deep interlays with other areas of mathematics provide deep insights, such as the undecidability of the halting problem, PRIMES is in P, and the breakthrough result exhibiting a quasi-polynomial time algorithm to decide if two graphs are isomorphic. In order to appreciate such results in theoretical computer science, formal proofs and the underlying ideas will be examined in great detail. Therefore, a key objective in this course is to develop your mathematical maturity; that is, your ability to understand mathematical statements and formulate rigorous mathematical proofs. This will ultimately be the best indicator for success (outside of hard work). I will rigorously prove mathematical statements in class and discuss proof strategy throughout this course. Every student will be expected to formulate proofs on homework and assessments. We will begin with a brief survey of discrete mathematics and proof techniques.

Course Objectives: The obvious course objective is learning the material outlined above. Beyond that, the development of rigorous mathematical thought, mathematical maturity, and sharpness of proof writing will be emphasized. The underlying goal is for you to improve your ability to read and write mathematics, as well as appreciate the design and usage of axioms in a theoretical discipline. A third goal is to provide a solid preparation for subsequent courses in Theoretical Computer Science, such as one might encounter at the graduate level, as well as theoretical courses in other departments (including, but certainly not limited to: Math, Economics, Physics).

Note: I am not assuming that everyone in the class feels comfortable writing mathematical proofs. There will be considerable emphasis on teaching you all how to write (better) proofs. If you find the material interesting and *want* to get better at writing proofs, then I encourage you to take this class!

Homework/Quizzes: Your learning in this class will ultimately come from making a good faith attempt to answer the homework questions. You should write up your problems formally and correctly, as if you were explaining the material to someone else. Proofs should be rigorous and flow logically. Homework will be assigned regularly, and will be submitted via OAKS. Each homework problem will be graded according to the following scale: Outstanding/Proficiency/Progress/Attempted/No Attempt. Grades of Outstanding and

Proficiency count equally for full credit, but are included to signal that a perfect solution is not required for full credit (Proficiency). For each homework assignment, you may revise and resubmit any (and every) question on which you did not receive a grade of Proficiency or Outstanding. Such revisions should be accompanied by reflections and will be due within two weeks of receiving your homework grade or the day before Reading Day (whichever comes first). Your homework score will be:

$$(\#\text{Outstanding} + \#\text{Proficiency})/(\#\text{Possible Problems}).$$

I will also regularly assign quizzes, mainly to check your understanding. Quizzes will be administered asynchronously via OAKS, timed at 45 minutes (scaled for students with disability accommodations). These are mainly to check your understanding of the content and for me to provide timely feedback. Quizzes will be graded according to the same scale as homework. Grades of Outstanding and Proficiency on quizzes will contribute positively to the numerator, but not to the denominator (number of possible problems). Thus, attempting quizzes can help you, but will not hurt you.

There will be two midterms, where each midterm question will be administered as its own OAKS quiz. Midterm questions will be graded according to the same scale as homework. However, unlike quizzes, the midterm questions will count towards the denominator (the number of possible problems). In my experience, students who keep up with the homework and quizzes are not surprised by the midterms and do quite well on them. There will not be revisions on quizzes or midterms.

Final: Instead of a traditional final exam, there will be a final reflection.

Challenge Problems: I will post challenge problems throughout the semester. These are more involved than standard homework problems and are designed to both deepen your understanding of the course content and be rewarding problem solving experiences. There will be suggested due dates for the challenge problems (mainly to help you plan your time), but these will not be strictly enforced. You will be able to turn in challenge problems up to a week before Reading Day (so that I have time to grade them). You will be able to revise your solutions to challenge problems as necessary (and up to the deadline), provided your revisions demonstrate a good faith effort at incorporating the feedback. Challenge problems are not required to pass the class, but are necessary to earn a grade of B+ or better.

Final Grades: Final grades will be issued according to the following cutoffs. Note that you must satisfy all of the conditions in a given row to earn a grade (e.g., a CSCI 410 student must earn an 87% on HW, submit a satisfactory final reflection, and demonstrate proficiency on at least two challenge problems to earn an A). I reserve the right to adjust these cutoffs downwards (more favorably) as I find is warranted, based on the interests of learning and fairness.

Grade	CSCI 410			CSIS 616		
	HW	**Final**	**Challenge**	**HW**	**Final**	**Challenge**
A	87%	Satisfactory	2	90%	Satisfactory	5
A-	84%	Satisfactory	1	87%	Satisfactory	4
B+	80%	Satisfactory	1	84%	Satisfactory	3
B	77%	Satisfactory	0	80%	Satisfactory	2
B-	74%	Satisfactory	0	77%	Satisfactory	1
C+	70%	Satisfactory	0	74%	Satisfactory	0
C	67%	Satisfactory	0	70%	Satisfactory	0
C-	64%	Not Satisfactory	0			
D+	60%	Not Satisfactory	0			
D	55%	Not Satisfactory	0			
D-	50%	Not Satisfactory	0			

For students considering enrolling in CSIS 616, please note that the College of Charleston does not have grades between D- and C- for graduate classes.

Grades in the **C-/C/C+** range indicate a solid command of the mechanics (e.g., working through procedures, coming up with basic examples/counterexamples). Grades in the **B-/B/B+** range indicate a solid command of the mechanics and a moderate understanding of the theoretical underpinnings. Grades in the **A-/A** range indicate a strong proficiency with both synthesis-based problem solving and formulating mathematical proofs, and this is a very solid indicator of preparedness for taking subsequent courses in Theoretical Computer Science and Math.

As a remark for students considering applying for PhD programs in Computer Science: depending on the program, it may be possible to count an undergraduate course for graduate credit provided that your graduate program deems it reasonable that the courses are equivalent. If you enroll in CSCI 410 and earn an A- or A at the graduate level (the CSIS 616 cutoffs), I can make note of this (with your permission) in a letter of recommendation or in discussion with the graduate program director wherever you land. Therefore, it may be worth pursuing an A- or A under the CSIS 616 cutoffs.