CSE396 Course Information Spring 2019

Instructor:

Dr. Kenneth W. Regan, 326 Davis Hall, 645-4738, regan@buffalo.edu

TAs and UTAs: (The common Davis 301-302 TA space is used for hours)

Yuhao Du	yuhaodu@buffalo.edu
Le Fang	lefang@buffalo.edu
Chaowen Guan	chaoweng@buffalo.edu
Junxuang Huang	junxuanh@buffalo.edu
Corey Ropell (UTA)	coreyrop@buffalo.edu

Office Hours:

- Regan, Wed. 1:15–3pm, other times TBA.
- TA hours TBA.

Lectures and Recitations:

(LEC) TuTh 11:00–12:20pm, in Norton 112

(R1) Mondays 8:00–8:50am, in Cooke 127A

(R2) Mondays 4:00–4:50pm, in Norton 210

(R3) Mondays 3:00–3:50pm, in Cooke 127A

(R4) Thursdays 3:30–4:20pm, in Norton 213

(R5) Tuesdays 3:00–3:50pm, in Clemens 106

(R6) Thursdays 8:00–8:50am, in Norton 210

(R7) Thursdays 2:00–2:50pm, in Norton 213

Examinations:

- Two *prelims*, the first either on **Thu., 3/8** or on **Tue., 3/13** in class period.
- One *cumulative* 3-hr. final.

1 Required Reading

(1) Text: Michael Sipser, *Introduction to the Theory of Computation, 3rd ed.* (Boston: Cengage Learning, 2013). *Intended coverage:* Chapters 0–5 and much of 7 (minus 2.4).

(2) Handouts and course notes provided by the instructors. Some of these will be given out in class besides being posted online.

(3) The course *TopHat* page. There will be online components to some assignments that will be administered from there.

(4) The *CSE396 Piazza page*. Although all official course information will be given in lectures, you may catch it first here, as well as information about the problem sets. Students are invited to post queries of general interest. Please do not, however, post answers unless and until cleared with the instructors and TAs.

(5) There will also be an *Autolab* page, primarily for grading...

(*) The *course webpage* will be for "static" information: syllabus info and notes and posted problem sets and answer keys, material that can be relied on not to change.

(*) I may put some texts on reserve, which will not be required reading.

2 Organization and Course Policies

Homework will consist solely of weekly problem sets—all due on *Thursdays* (midnight "stretchy") unless circumstances intervene. There are no programming projects or labs. Use of the *Turing Kit* software for certain assignments with "design a machine" problems is optional—as its printjob export can be wonky, the recommended way is to take screenshots (Windows Snipping Tool or similar) and import them.

Assignments must be submitted *as PDF files* (not MS Word files) in order to work with *CSE Autograder*. All submissions must have your *name and recitation* on them. The recitation should be the one that you regularly attend. They must be clearly legible on the PDF.

My (KWR) general policy is that *late work is not acceptable*. In return, you get an answer key shortly afterward, and a relatively quick turnaround of graded work before the next problem set is due. In an exceptional situation, you may contact me *beforehand* about a possible extension.

The course will be graded on a total-points system. Letter grades will also be given for individual exams and some assignments, as a help in telling you where you stand, but only the point totals will have official significance. The weighting of grades in this course is:

Attendance:	6%
Homework:	34% (about 10% online, 24% written)
Prelims:	24%
Final:	36%

Instructors reserve the right to 5% leeway in weighting while assigning the final letter grade—this is most typically done for students who do markedly well on the final exam, when it may be treated as if it were worth 41% for that student. This will only be done to an individual student's advantage, and will have no effect on others' grades. Academic honesty and recitation attendance are two prerequisites for this policy to be applicable.

Homework and all exams are designed for a curve with $\geq$ 90% = A, 84–89% = A-, 78–83 = B+, 72–77% = B, 66–71% = B-, 60–65% = C+, 54–59% = C, 48–53% = C-, 42–47% = D+, 36–41% = D, and $<$ 36% = F. Grades will be "curved" only if something systematic

happens to make clear that expectations on a problem were misplaced or snow happened or the allocated time was too short. Having course points sum to 667 makes the thresholds for the final course grades into nice, round numbers: 600/667 for an A, 560 for an A-, 520 for a B+, 480 for a B, 440 for a B-, 400 for a C+, 360 for a C, 320 for a C-, 280 for a D+, and 240 for a D and a pass.

2.1 Academic Honesty

A university is a *community*, and every community has values and rules that go hand-in-hand with membership in the community. At universities one rule is the standard of *academic honesty* as it has been understood and followed for **all** of the past millennium. This rule is not written down in a standard text such as Magna Carta or the Constitution, but is the same for every educational institution even though they all have individual statements of it. At bottom it is *honesty*. Different cultures and institutions may have different rules and expectations, but all have the principle of abiding by the local rules, and of not misrepresenting the nature of one's work. Students are required to read and abide by our version of the rules at

`http://engineering.buffalo.edu/computer-science-engineering/information-for-students/policies/academic-integrity-students.html`

In this course, all assignments will be individual, and the *rule* takes a particularly simple form: All assignments must be *your own work*. This term does not need a definition or legal parsing. Tens of thousands have graduated from my *alma mater* with no exam proctors or vetting, just needing to sign "This represents my own work in accordance with University regulations" on every exam paper or major submission. The only difficulty is for those actively (and quite justifiably) seeking help on assignments. The Department has a reasonable guideline tailored to programming projects at the above URL. Note that it is absolute that *writeups* of assignments must be completely your own production—in this course that is especially meaningful because *formal presentation* of solutions is an important course subject and goal in itself. I will talk about "reasonable discussion" of problems in class at some convenient and good time.

Well over half the cheating I used to catch on single problem sets were [*claimed to be*] one student copying off another who couldn't attend class and asked the "trusted friend" to submit for him/her. In such cases the University considers *both* students to be culpable, and my measures reflect this policy. There can be some leniency for "first offenses," but anything more serious is liable for an automatic F in the course and/or explusion from the Department (*cum*-University). The use of *Autograder* obviates this mechanism, but the philosophy remains. The much more serious recent technological development is students sending photos of their work to other students via smartphone—then the guilt of the enabler is sharper.

2.2 Incompletes and Withdrawals

This course shall follow strictly the University's standard for incompletes: they are only to make up work missed owing to circumstances beyond a student's control, and they are given only when a student has enough points to pass the course with already-completed work (i.e., no "I/F" grades). An "Administrative Withdrawal" after the "R" date is available only to students who withdraw from *all* their courses in a given term.

3 Nature and Purposes of the Course

The first main objective of the course is to convey those major concepts and results in the theory of computation that guide our thinking about the power of computers and the problems we can solve with them. This includes the entire historical origin of the field in the work of Alan M. Turing, John von Neumann, and Stephen C. Kleene. Finite automata, regular expressions, context-free (and other) grammars, pushdown automata, and idealized programs (if not the Turing machine, think of the Java Virtual Machine) are tools of everyday computing practice. Computational complexity theory asks the fundamental question of how much time, memory, and other computational resources computers need to solve certain problems, and today is relied upon for Internet security. This conveys the ability (ABET CS Outcome j) to apply mathematical foundations, algorithmic principles, and CS theory to model and design systems with comprehension of the tradeoffs involved in design choices.

A second main objective is not as "concrete" as the above-listed syllabus material, but is just as important. Computers are by-nature entirely formal entities—they do precisely what is prescribed in programming languages that are ultimately formal and mathematical. Not just to reason about them, but even to communicate effectively in the field and on the job, one must be able to state assertions precisely and design prototypes concisely. This requires fluency in the underlying mathematical language used to describe problems, computations, and objectives. This course gives valuable training in formal modes of reasoning, analysis, and presentation.

3.1 Chapters Covered

Part of Chapter 0 is assumed background; some parts will be addressed in the first lecture.

Chapter 1 all—hardcopy notes on the "Myhill-Nerode Theorem" will replace the text's "Pumping Lemma" and do the same examples.

Chapter 2, all except the long half of the proof of the equivalence of CFGs and PDAs, and excepting most of the section 2.4 on "DPDAs" which is new to this edition. (DPDAs will be introduced while covering chapter 3 as a special case of Turing machines.)

Chapter 3 all, with some shortcutting.

Chapter 4 all.

Chapter 5, all except section 5.2 on "PCP," and with more emphasis on mapping reductions.

Chapter 7, all or most, with hardcopy of a shorter version of the *Cook-Levin Theorem*. (Note that Chapter 6 is skipped.)

Sometimes the last lecture includes some topics from later chapters as "FYI" material.