CS 3350 Automata, Computability, and Formal Languages

Spring 2020 Course Syllabus

Course Description:
Theoretical computing models and the formal languages they characterize: finite state machines, regular expressions, pushdown automata, context-free grammars, Turing machines and computability. Capabilities and limitations of each model, and applications including lexical analysis and parsing.

Prerequisites:

- either CS 2302 Data Structures and MATH 2300 Discrete Mathematics, both with grades C or higher,
- or CS 2401 Elementary Data Structures and Algorithms and MATH 2300, both with grades B or higher.

Textbook:
Reading and laboratory assignments will be drawn from *Introduction to the Theory of Computation*, by Michael Sipser (both 2nd and 3rd editions are OK). This book is available at the bookstore and through major online book retailers, and you are expected to acquire a copy for your use in this course. Photocopied textbooks are illegal and their use will not be tolerated.

Exams:
There will be three tests, on February 20, March 24, and April 30, and the final exam on May 12, 10 am - 12:45 pm.

The purpose of the exams is to allow you to demonstrate mastery of course concepts. Make-up exams will be given only in extremely unusual circumstances. If you must miss an exam, please meet with an instructor, BEFORE the exam if at all possible.

Grading:

- first test: 15
- second test: 15
- third test: 15
- home assignments and quizzes: 20
- final exam: 30
- attendance: 5

The nominal percentage-score-to-letter-grade conversion is as follows:

- 90% or higher is an A
- 80-89% is a B
- 70-79% is a C
- 60-69% is a D
- below 60% is an F

We reserve the right to adjust these criteria downward, e.g., so that 88% or higher represents an A, based on overall class performance. The criteria will not be adjusted upward, however.

Homework Assignments: Homework and lab assignments are designed to allow you to practice using the concepts presented in lecture and in your reading. Homework and lab assignments may include written problems, tutorial exercises, and programming problems. Assignments usually will be due at the start of the next class. Late homework will be accepted only in unusual circumstances, by prior arrangement if at all possible.

Homework must be done individually. While you may discuss the problem in general terms with other people, your answers and your code should be written and tested by you alone. If you need help, consult a TA or a professors.

Quizzes: The purpose of a quiz is to ensure that you have read the weekly reading assignment and to verify that you have mastered the major concepts of recent lectures. Quizzes typically will be about 5-10 minutes in length and will cover the material assigned to be read for the upcoming lecture plus selected concepts from previous lectures. There will be no make-up on missed quizzes.

Special Accommodations: If you have a disability and need classroom accommodations, please contact the Center for Accommodations and Support Services (CASS) at 747-5148 or by email to cass@utep.edu, or visit their office located in UTEP Union East, Room 106. For additional information, please visit the CASS website at http://www.sa.utep.edu/cass. CASS's staff are the only individuals who can validate and if need be, authorize accommodations for students.

Scholastic Dishonesty: Any student who commits an act of scholastic dishonesty is subject to discipline. Scholastic dishonesty includes, but not limited to cheating, plagiarism, collusion, submission for credit of any work or materials that are attributable to another person.

Cheating is:

- copying from the test paper of another student;
- communicating with another student during a test to be taken individually;
- giving or seeking aid from another student during a test to be taken individually;
- possession and/or use of unauthorized materials during tests (i.e. crib notes, class notes, books, etc.);
- substituting for another person to take a test;
- falsifying research data, reports, academic work offered for credit.

Plagiarism is:

- using someone's work in your assignments without the proper citations;
- submitting the same paper or assignment from a different course, without direct permission of instructors.

To avoid plagiarism see: https://www.utep.edu/student-affairs/osccr/_Files/docs/Avoiding-Plagiarism.pdf

Collusion is unauthorized collaboration with another person in preparing academic assignments.

Instructors are required to -- and will -- report academic dishonesty and any other violation of the Standards of Conduct to the Dean of Students.

NOTE: When in doubt on any of the above, please contact your instructor to check if you are following authorized procedure.

Faculty Information:
Professor: Luc Longpré
Office: 3.0420 CCS building
Phone: 747-6804
e-mail: longpre @ utep . edu
Office Hours: MW 3-4pm.
See https://www.utep.edu/cs/people/longpre.html, select "Student appointments" for instructions on how to make appointments at other times.

Teaching Assistant (TA):
Namrata Sharma, email nsharma3@miners.utep.edu, office hours CCS building 1.0706 MW 10am-12noon.

Instructor of another section of Automata:
Vladik Kreinovich, email vladik@utep.edu, office hours TR 8:30-9 am, 10:30-12 pm, 1-1:30 pm.

Major Topics Covered in the Course:

- Regular languages, finite automata, non deterministic FA
- Context-free languages, pushdown automata
- Parsing, normal forms, ambiguity
- Pumping lemmas and closure properties
- Turing machines and other equivalent models
- Decidable languages, non-decidable languages, recognizable languages, Chomsky hierarchy

Learning Outcomes:

Level 1: Knowledge and Comprehension
Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. The material has been presented only at a
superficial level.
Upon successful completion of this course, students will:

1a. Be familiar with the implications of Church-Turing thesis.

1b. Understand that there are problems for which an algorithm exists, and problems for which there are no algorithms (non-recursive, non-recursively enumerable languages) and understand the implications of such results.

1c. Understand and explain the diagonalization process as used in proofs about computability.

1d. Understand the difference between feasible and non-feasible algorithms, understand the limitations of the current formalization of feasibility as polynomial-time.

1e. Understand the main ideas behind the concepts of NP and NP-hardness, know examples of NP-hard problems.

Level 2: Application and Analysis
Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details.
Upon successful completion of this course, students will be able to:

2a. Convert a non-deterministic FA (respectively transition graph) into an equivalent deterministic FA, convert a transition graph or NFA into an equivalent regular expression, and convert a regular expression into an equivalent FA.

2b. Construct a regular expression (respectively a context-free grammar) for a regular language (respectively context-free language).

2c. Convert a context-free grammar into an equivalent pushdown automaton.

2d. Construct a context-free grammar for a given context-free language.

2e. Design an algorithm for a machine model to simulate another model.

2f. Build simple Turing machines.

2g. Prove formally properties of languages or computational models.

2h. Apply a parsing algorithm.

2i. Build a parse tree or a derivation from a context-free grammar.

2j. Use the closure properties in arguments about languages.

Level 3: Synthesis and Evaluation
Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery.
Upon successful completion of this course, students will be able to:

3a. Compare regular, context-free, recursive, and recursively enumerable languages.

3b. Compare FA, PDA, and Turing machines.

Daily Schedule: (This is the tentative schedule of Dr. Kreinovich's section. We will attempt to follow the same schedule in both sections. Some of the details may be different, the schedule is subject to change, and the two sections may not always be exactly synchronized.)

January 21: topics to cover:

- motivations for the class: basis for compiler design; need to detect when a task is not algorithmically solvable;
- notion of a finite automaton;
- simple examples of finite automata -- for recognizing integers and for recognizing real numbers;
- formal notations for describing a finite automaton;
- algorithms for designing a finite automata that recognize union and intersection of regular languages.

assignments:

- tracing how a word is checked by an automaton;
- describing a given finite automaton in formal notations;
- designing a new automaton for simple tasks;
- applying the algorithms for union and intersection of regular languages that we learned in class to simple examples.

January 23: topics to cover:

- examples of union and intersection of regular languages;
- notion of a non-deterministic automaton (NDA);
- how to transform NDAs recognizing two languages into an NDA for their union;
- notion of concatenation of two languages;
- how to transform NDAs recognizing two languages into an NDA for their concatenation.

January 28: topics to cover:

- the notion of a Kleene star;
- how to transform a NDA recognizing a language into a NDA for recognizing its Kleene star;
- notion of a regular expression;
- how to transform a NDA into a deterministic one.

assignments:

- design a finite automaton for recognizing a given regular expression;

- transform a simple NDA into a deterministic one;
- transform a given finite automaton into a regular expression;
- write a program that simulates finite automata.

January 30: topics to cover:

- how to transform a NDA into a deterministic one (cont-d);
- how to transform a finite automaton into a corresponding regular expression;
- how to simulate finite automata.

February 4: topics to cover:

- how to transform a finite automaton into a corresponding regular expression (cont-d);
- how to simulate finite automata (cont-d).

assignment:

- transform a given finite automaton into a regular expression.

February 6: topics to cover:

- Pigeonhole Principle;
- Pumping Lemma;
- proof that some languages are not regular;
- examples of non-regular languages.

February 11: topics to cover:

- notion of a pushdown automaton;
- examples of pushdown automata;
- notion of a context-free grammar;
- examples of context-free grammars related to programming;
- examples of context-free grammars that generate non-regular languages.

assignments:

- use the proof that we had in class as a sample to prove that a given language is not regular;
- tracing pushdown automata;
- tracing simple context-free grammars.

February 13: topics to cover:

- how to transform a finite automaton into a context-free grammar;
- how to transform a context-free grammar into a (non-deterministic) pushdown automaton;
- overview for Test 1.

February 20: Test 1 that covers all the material that we have studied so far.

assignments:

- transforming a given finite automaton into a context-free grammar;
- transforming a given context-free grammar into a pushdown automaton.

February 22: overview of the results of Test 1.

February 27: topics to cover:

- how to design a context-free grammar corresponding to a given language;
- ambiguous vs. unambiguous grammars; Chomsky normal form.

assignments:

- transform a given context-free grammar into Chomsky normal form;

February 29: topics to cover:

- Chomsky normal form (cont-d);
- transforming a pushdown automaton into a context-free grammar.

March 3: topics to cover:

- transforming a pushdown automaton into a context-free grammar (cont-d);
- how compilers actually work: priority technique.

assignments:

- transform a given pushdown automaton into a context-free grammar.

March 5: topics to cover:

- priority techniques beyond simple arithmetic expression;
- compiling LL(k) languages.

assignments:

- show how a computer will parse and compute a given arithmetic expression;
- apply priority techniques to parse a given piece of code.

March 10: topics to cover:

- Pumping Lemma for context-free grammars;
- proof that some languages are not context-free.

assignments:

- applying Pumping Lemma to a given context-free grammar;
- prove that a given language is not context-free.

March 12: preview for Test 2

March 24: Test 2

March 26: topics to cover:

- proof that some languages are not context-free (cont-d);
- main ideas behind Turing machines;
- unary code;
- Turing machines for processing numbers in unary code.

March 31: topics to cover:

- overview of the results of Test 2;
- lowest-bit first vs. highest-bit first computer representation of numbers;
- Turing machines for simple operations with binary numbers.

assignments:

- trace a simple Turing machine;
- design a simple Turing machine for a given task.

April 2: topics to cover:

- Turing machines for simple operations with binary numbers;
- how to transform a finite automaton into a Turing machine;
- how to simulate a Turing machine.

April 7: topics to cover:

- how to simulate a Turing machine (cont-d);
- how to represent a Turing machine as a finite automaton with two stacks.

assignments:

- transform a given finite automaton into a Turing machines;
- write a program that simulates a general Turing machine.

April 9: topics to cover:

- feasible vs. non-feasible algorithms;
- example showing that the current definition of feasibility is not always adequate.

April 14: topics to cover:

- a general notion of a problem -- examples, resulting definition of the class NP;
- class P;
- P = NP problem;
- notion of reduction; examples of reduction;
- notion of NP-hardness and NP-completeness;
- examples of NP-hard problems.

assignments:

- represent a Turing machine as a finite automaton with two stacks;
- give other examples where the current definitions of feasibility are not adequate.

April 16: topics to cover:

- preview for Test 3.

April 21: topics to cover:

- proof by contradiction: general idea;
- proof that square root of 2 is not a rational number.

assignment: proof that the square root of 3 is not a rational number.assignments: recall definitions of P, NP, and related notions.<

April 23: topics to cover:

- definition of a halting problem;
- proof that it is not possible to check whether a given program halts on given data;
- overview of the results of Test 3.

April 28: topics to cover:

- Church-Turing thesis: what is it, is it a mathematical statement, is it a statement about the physical world;
- preview for the final exam.

assignment: reproduce the proof of the halting problem in all the necessary detail.

April 30: Test 3.

May 5: overview of Test 3 results.

May 7: topics to cover:

- the notion of a recursive (decidable) language;
- the notion of a recursively enumerable (Turing-recognizable) language.