

# CS 3350 Automata, Computability, and Formal Languages Fall 2025 Syllabus

**Class Time:** Tuesdays and Thursdays 3-4:20 pm

**Room:** CCSB G.0208.

**General Prerequisites:** CS 2302 "Data Structures" and either Discrete Mathematics or Discrete Structures, both with grades C or higher.

**Alternative Prerequisites:** CS 2401 "Elementary Data Structures and Algorithms" and either Discrete Mathematics or Discrete Structures, both with grades B or higher.

**Instructor:** [Vladik Kreinovich](#), email [vladik at utep.edu](mailto:vladik@utep.edu), office CCSB 3.0404, office phone (915) 747-6951.

- The instructor's office hours are Tuesdays and Thursdays 1:30-2:50 pm or by appointment.
- If you want to come during the scheduled office hours, there is no need to schedule an appointment.
- If you cannot come during the instructor's scheduled office hours, please schedule an appointment in the following way:
  - use the instructor's appointments page </vladik/appointments.html> to find the time when the instructor is not busy (i.e., when he has no other appointments), and
  - send him an email, to [vladik at utep.edu](mailto:vladik@utep.edu), indicating the day and time that you would like to meet. He will then send a reply email, usually confirming that he is available at this time, and he will place the meeting with you on his schedule.
- If the meeting is scheduled, but something happened and you cannot come, please let the instructor know about it as soon as possible.

## Teaching Assistant (TA):

Saeefa Rubaieyt Nowmi, email [snowmi@miners.utep.edu](mailto:snowmi@miners.utep.edu), office hours TBD, in room CCSB 1.0706, or by appointment

**Instructor of Another Section of Automata:** Luc Longpre

- Time: Mondays and Wednesdays 12-1:20 pm
- Dr. Longpre's email is [longpre at utep.edu](mailto:longpre@utep.edu)

**Course Objectives:** 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.

## Major Topics Covered in the Course

- Regular languages, finite automata (FA), non deterministic FA (NFA)
- Context-free languages, pushdown automata (PDA)
- 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 be able to:

- 1a. Describe implications of Church-Turing thesis.
- 1b. Describe problems for which an algorithm exists, and problems for which there are no algorithms (non-recursive, non-recursively enumerable languages) and describe the implications of such results.
- 1c. Describe and explain the diagonalization process as used in proofs about computability.
- 1d. Describe the difference between feasible and non-feasible algorithms, describe the limitations of the current formalization of feasibility as polynomial-time.
- 1e. Describe 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 finite automaton into an equivalent deterministic finite automaton.
- 2b. Convert a non-deterministic finite automaton into an equivalent regular expression.
- 2c. Convert a regular expression into an equivalent finite automaton.
- 2d. Construct a regular expression for a regular language.
- 2e. Convert a context-free grammar into an equivalent pushdown automaton.
- 2f. Construct a context-free grammar for a given context-free language.
- 2g. Design an algorithm for a machine model to simulate another model.
- 2h. Build simple Turing machines.
- 2i. Prove formally properties of languages or computational models.
- 2j. Apply a parsing algorithm.
- 2k. Build a parse tree or a derivation from a context-free grammar.
- 2l. 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 finite automata, pushdown automata, and Turing machines.

**Textbook:** *Introduction to the Theory of Computation*, by Michael Sipser (all 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.

**Assignments:** Reading and homework assignments will be announced on the class website. You should expect to spend at least 10 hours/week outside of class on reading and homework.

**Exams:** There will be three tests and the final exam. The final exam will be on December 11, 4-6:45 pm.

For each test, I will post solutions, send you the grades, and answer questions if something is not clear.

As usual, if you are unable to attend the test, let me know, I will organize a different version of the test at a time convenient for you.

**Grading:** Each topic means home assignments (mainly on the sheets of paper, but some on the real computer). Maximum number of points:

- attendance: 5
- first test: 20
- second half-test: 10
- third test: 20
- home assignments and quizzes: 20
- final exam: 25

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.

**Example of grading:** Suppose that:

- a student got 70/100 on the first test, 75/100 on the second test, 80/100 on the third test, and 85/100 on the final exam;
- the student attended 19 out of 28 class sections and was excused for missing 2 sessions, to the total of 21, and
- the student's average grade on all the homeworks is 8.6/10.

Then, according to the above formulas, the overall student's percentage score is:

$$(21/28) * 5 + (70/100) * 20 + (75/100) * 10 + (80/100) * 20 + (8.6/10) * 20 + (85/100) * 25 =$$

$$0.75 * 5 + 0.7 * 20 + 0.75 * 10 + 0.8 * 20 + 0.86 * 20 + 0.85 * 25 =$$

$$3.75 + 14.0 + 7.5 + 16.0 + 17.2 + 21.25 = 79.7,$$

which I will round up to 80. This score is in the 80-89% range, so the overall student's grade for the class is a B.

**Example of a preliminary estimate:** Suppose that the student has just got the result of his/her second test, and he/she wants to estimate his/her grade so far. Suppose that:

- a student got 70/100 on the first test and 75/100 on the second test;
- the student attended 15 out of 20 class sections, and

- the student's average grade on all the homeworks is 8.2/10.

The largest number of points that the student can get at this stage is:

$$5 + 20 + 10 + 20 = 55$$

Out of these points, according to the above formulas, the student's percentage score so far is:

$$\begin{aligned} (15/20) * 5 + (70/100) * 20 + (75/100) * 10 + (8.2/10) * 20 = \\ 0.75 * 5 + 0.7 * 20 + 0.75 * 10 + 0.82 * 20 = \\ 3.75 + 14.0 + 7.5 + 16.4 = 41.65. \end{aligned}$$

The student got 41.65 out of the possible 55 points. So, the student's percentage score is:

$$(41.65/55) * 100 = 76.2$$

This score is in the 70-79% range, so the student's grade so far is a C.

**Homework Assignments:** Each topic means home assignments. Homeworks will be usually assigned on Tuesday and be due on Thursday, any time on Thursday is still OK. To submit a homework, send it to the Teaching Assistant (TA) by email. If your solution is not electronic, scan it and send her the scanned version. When you send your homework to the TA, please use subject "CS 3350 Homework 1" if this is Homework 1, "CS 3350 Homework 2" if this is a solution to Homework 2, etc. If two homeworks are due at the same day, send two separate emails with solution to each homework. If you have a legitimate reason to be late, let me and the TA know, you can then submit it until the following Tuesday. If you were simply late, you can still submit until next Tuesday, but then the TA will take off points for submitting late.

The TA will send you the grades. On Tuesday the next week, I will post correct solutions, and both I and the TA will be glad to answer questions if needed.

Since I will be posting correct solutions to homeworks, it does not make any sense to accept very late assignments: once an assignment is posted, it make no sense for you to copy it in your own handwriting, this does not indicate any understanding. So, please try to submit your assignments on time.

Things happen. If there is an emergency situation and you cannot submit it on time, let me know, you will then not be penalized -- and I will come up with a similar but different assignment that you can submit directly to me (not to the TA) when you become available again.

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 the instructor or the TA.

**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) by email to [cass@utep.edu](mailto:cass@utep.edu). 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.

*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.

**Daily Schedule:** (tentative and subject to change)*August 26:* topics to cover:

- motivations for the class: basis for compiler design; need to detect when a task is not algorithmically solvable; see [lecture](#);
- notion of a finite automaton; see [lecture](#).

*August 28:* topics to cover:

- simple examples of finite automata -- for recognizing integers and for recognizing real numbers; see [lecture](#);
- formal notations for describing a finite automaton; see [lecture](#);
- algorithms for designing a finite automata that recognize union and intersection of regular languages; see [lecture](#).

*September 2:* topics to cover:

- notion of a non-deterministic automaton (NFA);
- how to transform NFAs recognizing two languages into an NFA for their union;
- notion of concatenation of two languages;
- how to transform NFAs recognizing two languages into an NFA for their concatenation;
- the notion of a Kleene star;
- how to transform a NFA recognizing a language into a NFA for recognizing its Kleene star;
- notion of a regular expression;
- how to transform a NFA into a deterministic one;

see [lecture](#).

*September 4:* topics to cover:

- how to transform a finite automaton into a corresponding regular expression; see [lecture](#);
- how to simulate finite automata; see [lecture](#).

*September 9:* topics to cover:

- Pigeonhole Principle; Pumping Lemma; see [lecture](#);
- proof that some languages are not regular; examples of non-regular languages; see [lecture](#).

*September 11:* topics to cover:

- notion of a pushdown automaton; examples of pushdown automata; see [lecture](#);
- notion of a context-free grammar; examples of context-free grammars related to programming; examples of context-free grammars that generate non-regular languages; see [lecture](#).

*September 16:* topics to cover:

- how to transform a finite automaton into a context-free grammar; see [lecture](#);
- how to transform a context-free grammar into a (non-deterministic) pushdown automaton; see [lecture](#).

*September 18:* overview for Test 1

*September 23:* Test 1

*September 25:* topics to cover:

- Chomsky normal form; see [lecture](#);
- transforming a pushdown automaton into a context-free grammar; see [lecture](#)

*September 28:* topics to cover:

- overview of Test 1 results;
- ambiguous vs. unambiguous grammars; see [lecture](#).

*October 2:* topics to cover:

- transforming a pushdown automaton into a context-free grammar; see [lecture](#);
- Pumping Lemma for context-free grammars;
- proof that some languages are not context-free;

see [lecture](#).

*October 7:* practice for Half-Test 2

*October 9:* Half-Test 2

*October 14:* overview of Half-Test 2

*October 16:* topics to cover:

- how compilers actually work: priority technique; see first part of the [lecture](#).

*October 21:* topics to cover:

- priority techniques beyond simple arithmetic expression; see second part of the [lecture](#).

*October 23:* topics to cover:

- relation between compiling and context-free grammars;
- compiling of LL(k) languages;

see [lecture](#).

*October 28:* topics to cover:

- main ideas behind Turing machines;
- unary code;
- Turing machines for processing numbers in unary code;
- lowest-bit first vs. highest-bit first computer representation of numbers;
- Turing machines for simple operations with binary numbers;
- how to transform a finite automaton into a Turing machine;

see [lecture](#).

*October 30:* topics to cover:

- how to simulate a Turing machine; see [lecture](#);
- how to represent a Turing machine as a finite automaton with two stacks; see [paper](#).

*November 4:* topics to cover:

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

see Section 2.1.1 of a [paper](#) and [comments](#)

*November 6:* 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;

see [lecture](#) and Sections 2.2 and 2.3 of the [paper](#).

*November 11:* topics to cover:

- proof by contradiction: general idea; proof that square root of 2 is not a rational number; see [lecture](#);
- definition of a halting problem; proof that it is not possible to check whether a given program halts on given data; see [lecture](#).

*November 13:* topics to cover:

- Church-Turing thesis: what is it, is it a mathematical statement, is it a statement about the physical world; see [lecture](#);
- the notion of a recursive (decidable) language; the notion of a recursively enumerable (Turing-recognizable) language; see [lecture](#).

*November 18:* review for Test 3

*November 20:* Test 3

*November 25:* review of Test 3 results.

*December 2:* review for the final exam.

*December 4:* review for the final exam.