CSC 473, Fall 2020

Automata, Grammars, and Languages

Eric Anson

THE UNIVERSITY
OF ARIZONA

CSC 473: Automata, Grammars, Languages

THE UNIVERSITY OF ARIZONA
Computer Science

Class website https://piazza.com/arizona/fall2020/csc473/home
D2L – videos, quizzes, calendar
GradeScope – turning in assignments and exams

- **Professor: Eric Anson**
  - eanson@email.arizona.edu
  - Office Hours: Mon 2-3, Wed 3-4 – via zoom

- **Teaching Assistants:**
  - Jesse Liu
  - Amanda Bertsch
  - Jack Zhang
  - Office Hours: TBA

- **Class email:**
  - cs473f20@cs.arizona.edu

CSC 473: Learning Outcomes

From the ACM-IEEE 2013 Curriculum Guidelines

THE UNIVERSITY OF ARIZONA
Computer Science

- Discuss the concept of finite state machines.
- Design a deterministic finite state machine to accept a specified language.
- Generate a regular expression to represent a specified language.
- Design a context-free grammar to represent a specified language.
- Define the classes P and NP.
- Explain the significance of NP-completeness.
- Explain why the halting problem has no algorithmic solution.

Covid Stuff

- We will be online until phase 3 begins (date???)
- Then those who wish will see lectures live in Chemistry 134
- Lectures will be recorded for viewing at another time.
- Lectures will also be on zoom.
- Because they are being recorded, please leave your cameras off.
  - I miss seeing students faces, but for the recording it is better to have just one video source up.
- After Thanksgiving all classes will be online again for the remainder of the semester.

Classroom

- When (if) we are in class together <u>everyone must wear a face mask</u>.
- It is actual school policy that everyone wear a mask everywhere on campus. (except in your own room/office or outside where a distance of 6ft can be maintained)
- Also, please use social distancing. No students should sit next to any other student (the room is huge)
- This is a crazy time and I will be SOOO glad when it is over, but until then we all have to do are part to slow the spread of the disease.

About CSC 473: Grading

- **Two Midterms (each 22%) 44%**
  - Dates: ~September 30 and November 4
- **Assignments 20%**
  - These will be written assignments that consist mostly of questions from the text.
- **Final 30%**
  - On all the material covered in the course
- **Quizzes 6%**

Books

The University of Arizona
Computer Science

- **Required:**
- ***Sipser, Michael; Introduction to the Theory of Computation, 3rd Edition; (Cengage Learning)***

Assignments

- There will be written assignments that will be 20% of your grade. These assignments will be mostly (if not entirely) from the book.
- Homework is for you to learn the material and most of your grade comes from tests so . . .
- Unless you are told otherwise, working on homework with your friends is NOT cheating for this class. BUT NEVER copy your work from any source.
- So to restate copying is cheating but discussing the problems (unless told otherwise) is not.
- You must create a pdf of your assignment and upload it to GradeScope.

Attendance

- This is a strange semester. I hope some of you will want to attend class when/if it is given in person.
- It would also be beneficial to attend the zoom session live so you can ask questions.
- You will be responsible for material covered in class even if it does not appear in the book.
- 6% of the final grade will come from quizzes. Most of the time quizzes will refer directly from the lectures, so attending or listening to the lectures will be required.

Professor: Dr. Eric Anson

CSC 473: Automata, Grammars, and Languages

- BS Math/Comp Sci - Pepperdine University - 1985
- MS Mathematics - University of Arizona - 1993
- PhD Comp Sci - University of Arizona – 2000
- Joined faculty of University of Arizona in fall 2015
- Have taught 101, 210, 245, 345, 352, 473
- Also taught math courses in 1988-1993 as a TA
- Took the graduate version of this class while in the math department in 1991

Why Theory For Me?

THE UNIVERSITY OF ARIZONA
Computer Science

1. If you can master theory, you can master anything (take with a grain of salt).

Why Theory For Me?

THE UNIVERSITY OF ARIZONA
Computer Science

1. If you can master theory, you can master anything (take with a grain of salt).
2. There is nothing “fuzzy” and there are no calculation errors.

Why Theory For Me?

1. If you can master theory, you can master anything (take with a grain of salt).
2. There is nothing “fuzzy” and there are no calculation errors.
3. Your lab is your brain.

Why Theory For Me?

THE UNIVERSITY OF ARIZONA
Computer Science

1. If you can master theory, you can master anything (take with a grain of salt).
2. There is nothing “fuzzy” and there are no calculation errors.
3. Your lab is your brain.
4. Theory never becomes obsolete.

What will we be studying in this class?

THE UNIVERSITY OF ARIZONA
Computer Science

1. Finite Automata and Regular Expressions
2. Pushdown Automata and Context Free Grammars
3. Turing Machines and Complexity

Finite Automata and Regular Languages

Applications to search, compilers, software verification, protocols, circuits, …

- String matching and searching
- Lexical analysis in compilers
- Software Verification
- Communication Protocols
- Software for designing and checking the behavior of digital circuits.
- Software for scanning large bodies of text
- Vending machines
- Video games

Why Study Grammars and PDAs?

Applications to programming language specifications, compilers, manuals

- Computer Languages
- Compiler Front Ends
- Code Generation
- Natural Language Processing
- Computational Biology

Computational Complexity (Turing Machines)

THE UNIVERSITY OF ARIZONA
Computer Science

Theory provides the tools for dealing with inherently hard problems

- **There are limitations on what software can do**
  - **Intractable** problems: there are programs, but no known fast programs
  - **Undecidable** problems: there can be no program to solve them

- **The class of seemingly unrelated intractable problems**
  - Precise term: the class of NP-complete problems
  - A fast solution to one implies fast solutions to all of them
  - Examples of NP-complete problems:
    Traveling Salesman, Multiple Sequence Alignment

- **What can we do?**
  - Settle for a less than perfect or approximate answer

Preparation

- Read Chapter 0 and 1.1
- This class will involve many proofs, especially inductive proofs. You may want to review those.

Sets

- **A set is a group of objects.**
  - unordered
  - No notion of multiple occurrences of an element
- **Subsets**
  - Given two sets A and B we say A is a subset of B ($A \subset B$) iff $\forall x, x \in A \Rightarrow x \in B$
  - A is a proper subset of B iff $A \subset B$ AND $\exists x \in B$ s.t. $x \notin A$
  - Make sure you also remember the definitions for union, intersection, etc.
- **Cartesian Product**
  - The cartesian product of two sets A and B ($A \times B$) is $\{(x,y) : x \in A \text{ and } y \in B\}$

Sets

- **Cardinality**
  – The cardinality of a set A (|A|) is the number of members of A if A is finite.
    - What if A is infinite?

Sets

- **Cardinality**
  – The cardinality of a set A ($|A|$) is the number of members of A if A is finite.
    - What if A is infinite?
      – We'll get to that later.

Sets

- **Cardinality**
  - The cardinality of a set A (|A|) is the number of members of A if A is finite.
    - What if A is infinite?
      - We'll get to that later.
- **Question:**
  - If A and B are finite, what is the cardinality of $A \times B$? ($|A \times B|$)

Functions

- **Definition**
  - A function or mapping from set A to set B ($f: A \rightarrow B$) is an assignment of a single element of from B for each element of A.
    - How do we formalize this?

Functions

- **Definition**
  - A <u>function</u> or <u>mapping</u> from set A to set B (f: A → B) is an assignment of a single element of from B for each element of A.
- **1-1 (one to one)**
  - A function f: A -> B is <u>1-1</u> or an <u>injection</u> iff
    $f(a) = f(b) \Rightarrow a = b$
    - In other words f doesn't map two different elements of A to the same element of B
- **Onto**
  - A function f: A -> B is <u>onto</u> or a <u>surjection</u> iff
    $\forall\, b \in B, \exists\, a \in A$ s.t. $f(a) = b$
    - In other words every element in B is mapped to by some element of A
- **1-1 Correspondence**
  - A function f is a <u>1-1 correspondence</u> or <u>bijection</u> if it is 1-1 and onto

Cardinality

- **Comparing cardinality**
  - For sets A and B, we say $|A| = |B|$ iff $\exists$ bijection $f: A \rightarrow B$
    - This works for finite AND infinite sets
  - We say $|A| \leq |B|$ iff $\exists$ 1-1 function $f: A \rightarrow B$

Cardinality

- **Comparing cardinality**
  - For sets A and B, we say $|A| = |B|$ iff $\exists$ bijection $f: A \rightarrow B$
    - This works for finite AND infinite sets
  - We say $|A| \leq |B|$ iff $\exists$ 1-1 function $f: A \rightarrow B$
- **Countable**
  - We say a set A is <u>countable</u> iff $|A| \leq \mathcal{Z}^+$ (positive integers)
    - Note all finite sets are countable
    - Many infinite sets are also countable (we often refer to them as <u>countably infinite</u>)

Cardinality

- **Comparing cardinality**
  - For sets A and B, we say |A| = |B| iff ∃ bijection f: A → B
    - This works for finite AND infinite sets
  - We say |A| ≤ |B| iff ∃ 1-1 function f: A → B
- **Countable**
  - We say a set A is <u>countable</u> iff |A| ≤ $\mathcal{Z}^+$ (positive integers)
    - Note all finite sets are countable
    - Many infinite sets are also countable (we often refer to them as <u>countably infinite</u>)
- **Thm: Given sets A, B then |A| = |B| iff |A| ≤ |B| and |B| ≤ |A|**
  - In other words if there is an injection from A to B and an injection from B to A, then there is a bijection from A to B.
  - We will skip the proof of this

Languages

- **Alphabet**
  - An alphabet is a finite, nonempty set of symbols.
  - We will denote the set by Σ
    - For example $\Sigma = \{0, 1\}$
    - or $\Sigma = \{0, 1, 2, \ldots, 9\}$
- **String**
  - A string is a finite sequence (possible empty) over some alphabet
  - The empty string is denoted by ϵ
  - The length of a string w, denoted by $|w|$ is the number of symbols in w (counting repeats)
  - The concatenation of two string s and t is the string formed by appending t to s

Languages

- **String (cont)**
  - if w is a string and n is a nonnegative integer then w^n is the string w concatenated n times
    or defined recursively:
    - $w^0 = \epsilon$
    - $w^{i+1} = w^i w$
  - if w is a string then define the reverse of w denoted w^R as
    - if $|w| = 0$, $w = \epsilon$ and $w^R = w$
    - if $|w| \geq 1$ then $\exists\ a \in \Sigma$ s.t. $w = xa$, $w^R = (xa)^R = ax^R$
  - Theorem: If w and x are strings then $(wx)^R = x^R w^R$

Languages

- **Def: Let Σ^i be all the strings over Σ of length i**
-

Languages

- **Def: Let Σ^i be all the strings over Σ of length i**
- **Def: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$**

Languages

- Def: Let Σ^i be all the strings over Σ of length i
- Def: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
- So Σ^* is all the strings over Σ
  (* is called the Kleene Star)
- How big is Σ^* ?

Languages

THE UNIVERSITY OF ARIZONA
Computer Science

- **Def: A language over an alphabet Σ is a subset of Σ^***

Languages

- **Def: A language over an alphabet Σ is a subset of Σ^***

- **How many languages are there?**

Languages

THE UNIVERSITY OF ARIZONA
Computer Science

- **Languages are sets, so we can perform any set operation on them.**
- **We will can define concatenation and Kleene Star on languages.**
  - Given languages L_1, L_2 define
    $L_1L_2 = \{ w_1w_2 : w_1 \in L_1 \text{ and } w_2 \in L_2\}$
  - Now define $L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots$

Problem

- Is the following true or false (prove your answer)
- $\forall\ L_1, L_2\ (L_1 = L_2 \text{ iff } L_1{}^* = L_2{}^*)$