Overview

What is the Course About?

• Chapter Introduction
• Finite Automata
• Formal Definition

What is computation?
Goals of Course

• Automata
  - What is a formal model of computing?
  - Are there different models, with different power?

• Computability
  - Are there some problems that are not possible to compute?

• Complexity
  - How much time or space do problems take?
  - Can we separate different classes of problems
    + Polynomial, log, exponential?

Expected Outcomes

• Learn about theory of computation
  - Different models, computability, complexity

• Just as you might know how computation happens at gate level, operating system level, programming level, good to understand it from a formal standpoint as well

• In practical terms,
  - Understand why heuristics might be needed for some problems
  - Understand the power of using simpler models of computation
  - Understand that there are limitations to computation

• Be able to do symbolic proofs
Homework

- Homework is due Saturday at 11:55pm
  - If needed, you can email me with a request for an extension
  - Extension for one homework will be automatically granted

- Homework must be submitted via Sakai
  - Upload a single pdf
  - Homework should be typeset
  - Will need to include diagrams of FSAs and PDAs
    + Can include hand-drawn solutions taken with cellphone
    + Can use tikz package to draw them

Logistics

- Grading
  60% homework
  20% midterm
  20% final

- Course website: cslu.ohsu.edu/~heeman/cs533
Critique

• Sakai will give you answer key after you submit your answers
  - You will have until Sunday at 11:55pm to submit a critique
  - Explain what you did wrong, and why you made that mistake
  - Worth up to half the marks that you lost
    + Really good explanations might even get more
  - Should show that you reviewed and understood answer key and
    understood whether your answer was correct
  - See sample homework for how to format this

Academic Integrity

• You can do the homeworks with your colleagues
  - Document how you collaborated
    - e.g., worked on entire question together, versus clarified what the
      question meant
• After you have come up with a solution, redo it on your own
  - Nothing in, nothing out, wait at least an hour between
• Do not google the answer
Overview

- What is the Course About?
  ⇒ Chapter Introduction
- Finite Automata
- Formal Definition

Chapter 1: Regular Languages

- What is a computer?
  - Definition gets into all sorts of complicated issues
    + Types of memory
    + How it is addressed?
    + What types of operations are supported?
- Let’s focus on an idealized computer
  - A ‘computational model’
  - As with any model, may be accurate in some ways, but not in others
- Start with as simple of a model as possible
  - finite state machine or finite automaton
Overview

• What is the Course About?
• Chapter Introduction
  ⇒ Finite Automata
• Formal Definition

Automatic Door Problem

• What can a machine with extremely limited amount of memory do?
• Say we want a controller for an automatic door that swings open
  - Only for people going into store
  - If someone is on the front pad and door is closed
    + Should keep the door open until the person is off of the rear pad, so that the door doesn’t knock the person over
  - If someone is on the read pad and door is close
    + Should keep the door closed
  - If someone steps on front pad and rear pad at the same time, it should not open the door, otherwise it might knock over the person on the rear pad
A Finite Automata

- So, machine needs two states
  - Is the door open or closed?
- On sensor input, it might transverse to the other state
- Transition Table:

<table>
<thead>
<tr>
<th>Input Signal</th>
<th>Neither</th>
<th>Front</th>
<th>Rear</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Closed</td>
<td>Open</td>
<td>Closed</td>
<td>Closed</td>
</tr>
</tbody>
</table>

- What will controller do if started in closed and receives FRONT, REAR, NEITHER, FRONT, BOTH, NEITHER, REAR, NEITHER?

Finite Automatas

- Many problems can be solved by an FA
  - e.g. Elevator controller, Thermostat, Dishwasher, ...
  - If problem solvable by an FA, why use something more complicated?
    - Use simplest computational model possible!
- So, let’s understand what this model
  - is capable of
  - and not capable of
- So, need to be more formal about this model
  - Need a precise definition of it
    - So, we know exactly what it is
  - Formal definition makes it possible to do proofs of its power
  - This will prepare us for studying more complex models
What is an Finite Automata?

• Consider the following FA \( M_1 \)
  - Called a state diagram
• It has
  - 3 states, start state, accept state
  - transitions: what state it transverses from/to with a certain input
• Processes an input
  - As it processes each character, it transitions from state to state
  - Example: 1101
• Accept or reject
  - If in an accepting state at end of input, accept, else reject
  - So, for any input, it just makes a binary decision
• What are the strings that \( M_1 \) accepts?

Still need a formal definition

• State diagram gives us a good intuition about what an FA is
• But, we still need a formal definition
  - Must there be an accept state? Can there be several?
  - Must there be a reject state? Can there be several?
  - Must there be a transition for every character from each state?
  - Can there be several transitions from a state for the same character?
• Need a precise definition for doing proofs about its power!
Overview

• What is the Course About?
• Chapter Introduction
• Finite Automata
  ⇒ Formal Definition

Formal Definition

A finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where
1. \(Q\) is a finite set called the states
2. \(\Sigma\) is a finite set called the alphabet
3. \(\delta : Q \times \Sigma \rightarrow Q\) is the transition function
4. \(q_0 \in Q\) is the start state
5. \(F \subseteq Q\) is the set of accepting states
Language of a Machine

• If \( A \) is the set of all strings that machine \( M \) accepts
  - We say that \( A \) is the language of machine \( M \)
  - We write \( L(M) = A \)
  - We say \( M \) recognizes \( A \)
    + We reserve the word \textit{accepts} to talk about individual strings: \( M \textit{ accepts} \)

• \( M \) may accept a lot of different strings, but only recognizes one language
  - If the machine accepts no strings, it still recognizes one language: \( \emptyset \)

• For our example:
  - Let \( A = \{ w \mid w \text{ contains at least one 1 and an even number of 0s following the last 1} \} \)
  - \( L(M_1) = A \) or \( M_1 \) recognizes \( A \)
Another Example

- Definition of $M_2$:
- Language of $M_2$:

---

Yet Another Example

- Definition of $M_3$:
- Language of $M_3$:
- What is the relationship between $M_2$ and $M_3$?
**Formal Definition of Computation**

Let \( M = (Q, \Sigma, \delta, q_0, F) \) be a finite automaton.
Let \( w = w_1w_2...w_n \) be a string over \( \Sigma \).
We say that \( M \) accepts \( w \) if there is a sequence of states \( r_0, ..., r_n \) such that
1. \( r_0 = q_0 \)
2. \( r_{i+1} = \delta(r_i, w_{i+1}) \) for \( i = 0, ..., n - 1 \)
3. \( r_n \in F \)

- Back to language of an FA
  - We say that \( M \) recognizes language \( A \) if \( A = \{ w | M \text{ accepts } w \} \)

**And Another**

- Design a FA that takes a sequence of digits (0-5)
  - Accepts if sum of digits is evenly divisible by 3
Regular Language

**Definition:** A language is called a **regular language** if some finite automaton recognizes it.

- Note that a language defines what the automata accepts, so defines the computation that the automata does.
  - What it answers ‘accept’ to

Designing Finite Automata

- Think of what states you need to represent the important part of what you have seen in the input so far
- Think of how you can transition between states
- Examples:
  - language of all strings with an odd number of 1s
  - language in which all strings contain the substring 001
  - Example: language of \{good, bad\}