Today's learning goals

- Use and design a finite automaton via its
  - Formal definition
  - State diagram
- Identify the strings and languages accepted by a given finite automaton
- Design a finite automaton which accepts a given language
- Define the regular operations on languages
Deterministic Finite automaton

- Input: finite string over a fixed alphabet
- Output: "accept" or "reject"

Language of the machine is the set of strings it accepts,
\[ L(M) = \{ x \in \Sigma^* \mid \text{if } x \text{ is accepted by } M, x \in L \} \]
Deterministic finite automaton

- Input: finite string over a fixed alphabet
- Output: "accept" or "reject"

Does this DFA accept the empty string?

A. Yes
B. No
C. I don't know
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. \(\delta : Q \times \Sigma \to Q\) is the transition function
4. \(q_0 \in Q\) is the start state
5. \(F \subseteq Q\) is the set of accept states.

\(\delta((q, x))\) next state is \(q'\)
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. \(\delta : Q \times \Sigma \to Q\) is the transition function
4. \(q_0 \in Q\) is the start state
5. \(F' \subseteq Q\) is the set of accept states.

Can there be more than one start state in a finite automaton?

A. Yes, because of line 4.
B. No, because of line 4.
C. I don't know
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 accept states.

How many outgoing arrows from each state?

A. May be different number at each state.
B. Must be 2.
C. Must be \(|Q|\).
D. Must be \(|\Sigma|\).
E. I don't know.
An example

Define \( M = (Q, \Sigma, \delta, q_0, F) \) where the function \( \delta \) is specified by its table of values:

<table>
<thead>
<tr>
<th>Input in ( Q \times \Sigma )</th>
<th>Output in ( Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q1,a)</td>
<td>q3</td>
</tr>
<tr>
<td>(q2,a)</td>
<td>q2</td>
</tr>
<tr>
<td>(q3,a)</td>
<td>q3</td>
</tr>
<tr>
<td>(q4,a)</td>
<td>q2</td>
</tr>
</tbody>
</table>

Draw the state diagram for the DFA with this formal definition.
An example

What's an example of a

- length 1 string accepted by this DFA?
- length 1 string rejected by this DFA?
- length 2 string accepted by this DFA?
- length 2 string rejected by this DFA?

\{q_1, q_2, q_3, q_4\}, \{a, b\}, \delta, q_1, \{q_4\}
An example

What's the best description of the language recognized by this DFA?

A. Starts with b and ends with a or b
B. Starts with a and ends with a or b
C. a's followed by b's
D. More than one of the above
E. I don't know.

and using set notation?
An example

This DFA recognizes the language of all strings of the form a's followed by b's

\[ \{ a^n b^k \mid n, k \geq 1 \} \]
Specifying an automaton

\((\{q_1,q_2,q_3\}, \{a,b\}, \delta, q_1, ?)\)

What state(s) should be in \(F\) so that the language of this machine is \(\{ w \mid ab \text{ is a substring of } w \}\)?

A. \(\{q_2\}\)
B. \(\{q_3\}\)
C. \(\{q_1,q_2\}\)
D. \(\{q_1,q_3\}\)
E. I don't know.
Specifying an automaton

\[(\{q_1, q_2, q_3\}, \{a, b\}, \delta, q_1, ?)\]

What state(s) should be in \(F\) so that the language of this machine is \(\{w | \text{b's never occur after a's in } w\}\)?

A. \(\{q_2\}\)
B. \(\{q_3\}\)
C. \(\{q_1, q_2\}\)
D. \(\{q_1, q_3\}\)
E. I don't know.

\(\Sigma = \{a, b\}\)
Building DFA

Typical questions

Define a DFA which recognizes the given language $L$.  

or

Prove that the (given) language $L$ is regular.

i.e. $\exists M \ ( M \text{ is a DFA and } L(M) = L)$
Building DFA

Example
Define a DFA which recognizes
\{ w | w has at least 2 a's \}

\[ \Sigma = \{ a, b \} \]
Building DFA

Example

Define a DFA which recognizes

\{ w \mid w \text{ has at most 2 a's}\}
Building DFA

Remember

States are our only (computer) memory.

Design and pick states with specific roles / tasks in mind.

"Have not seen any of desired pattern yet"

"Trap state"
Justification?

To prove that the DFA we build, M, actually recognizes the language $L$

$$\textbf{WTS } L(M) = L$$

(1) Is every string accepted by M in $L$?
(2) Is every string from $L$ accepted by M?

or contrapositive version: Is every string rejected by M not in $L$. 
A useful (optional) bit of terminology

When is a string accepted by a DFA?

**Computation of M on w**: where do we land when start at $q_0$ and read each symbol of $w$ one-at-a-time?

$$\delta^*( q, w ) =$$

Recursively defined function
Regular languages

- For DFA $M$ over the alphabet $\Sigma$
  - For each string $w$ over $\Sigma$, $M$ either accepts $w$ or rejects $w$
  - The **language recognized by $M$** is the set of strings $M$ accepts
  - The **language of $M$** is the set of strings $M$ accepts
  - $L(M) = \{ w \mid w \text{ is a string over } \Sigma \text{ and } M \text{ accepts } w \}$

A language is **regular** iff there is some finite automaton that recognizes **exactly** it.
Regular languages: bounds?

Is **every** finite language regular?

A. No: some finite languages are regular, and some are not.
B. No: there are no finite regular languages.
C. Yes: every finite language is regular.
D. I don't know.
For next time

• Complete Review Quiz 1 by Friday night
• Continue working on Homework 1 due Monday night
  • Set up course tools: Gradescope, Piazza, JFLAP, haskell
  • Find group members
  • Read all the questions
  • Start working 😊
  • Review CSE 20 / Math 109 / CSE 21 / Sipser Ch 0 as needed.