Today's learning goals

- Convert between regular expressions and automata
- Choose between multiple models to prove that a language is regular
- Describe the limits of the class of regular languages

On Gradescope: interim grade report

Is your clicker being recorded?

Office hours today 12:30-2
DFA equiv NFA

**Theorem:** For each language \( L \),

\[ L \text{ is recognizable by some DFA} \iff L \text{ is recognizable by some NFA} \]

*Note:* machine \( M \) recognizes language \( L \),

\[ L(M) = L \]
Details

Suppose \( M = (Q, \Sigma, \delta, q_0, F) \) is a DFA. An equivalent NFA is \( N = (Q, \Sigma, \delta', q_0, F) \) where

\[
\delta'((q, x)) = \begin{cases} 
\{ \delta((q, x)) \} & \text{if } q \in Q, x \in \Sigma \\
\emptyset & \text{if } q \in Q, x = \varepsilon
\end{cases}
\]

Conversely, suppose \( N = (Q, \Sigma, \delta, q_0, F) \) is a NFA. An equivalent DFA is \( M = (Q', \Sigma, \delta', q_0', F') \) with \( Q' = \) the power set of \( Q = \{ X \mid X \text{ is a subset of } Q \} \)

\( q_0' = \{ q_0 \} \cup \delta((q_0, \varepsilon)) \) ... i.e. states accessible from \( q_0 \) via spontaneous moves

\( F' = \{ X \mid X \text{ and } F \text{ are not disjoint} \} \)

\( \delta'((X, x)) = \{ q \in Q \mid q \text{ is in } \delta((r, x)) \text{ for some } r \in X \text{ or is accessible via spontaneous moves} \} \)

states of \( M \) keep track of all possible compass \( \delta N \)
Theorem: For each language L,

L is recognizable by some DFA
iff
L is recognizable by some NFA
iff
L is describable by some regular expression
From RegExp to DFA

**Structural induction!**

<table>
<thead>
<tr>
<th></th>
<th>Base cases</th>
<th>Inductive steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$R = a$, where $a \in \Sigma$</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>$R = \epsilon$</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>$R = \emptyset$</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>$R = (R_1 \cup R_2)$</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>$R = (R_1 \circ R_2)$</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>$(R_1^*)$</td>
<td></td>
</tr>
</tbody>
</table>

- Build DFAs (or NFAs) corresponding to base cases in inductive definitions of regular expressions.

- Describe constructions for DFAs corresponding to each of the inductive steps: union, concatenation, Kleene star.
Structural induction

Thm 1.45, 1.47, 1.49 pp 59-62

- Base cases

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

\[ L(#1) = \{ a \} \]

\[ L(#2) = \emptyset \]

\[ L(#3) = \{ \varepsilon \} \]

Which of these recognizes \( L(\emptyset) \)?

A. NFA #1
B. NFA #2
C. NFA #3
D. More than one of the above
E. None of the above

For all strings, \( M \) reject that string.
Structural induction

- **Base cases**

- **Inductive steps**

  - Union
  - Concatenation
  - Kleene star

Thm 1.45, 1.47, 1.49 pp 59-62
Example

Base case: $a^* (ab)^*$
From DFA to RegExp

Trace possible paths from start state to accept state.

Intermediate machines (called Generalized NFA) can have regular expressions on transitions.

First

1. add new start state that has $\varepsilon$ arrow to old start state
2. add new accept state that has $\varepsilon$ arrows from old accept states (and $\emptyset$ arrows from nonaccept states)
From DFA to RegExp

Remove one state at a time.

- Restore automaton by modifying regular expressions on transitions that went through removed state.
Regular languages

To prove that a set of strings is regular:

1. Build a **DFA** whose language is this set. OR
2. Build an **NFA** whose language is this set. OR
3. Build a **regular expression** describing this set. OR
4. Use the **closure** properties of the class of regular languages to construct this set from others known to be regular (**complementation**, **union**, **intersection**, **concatenation**, **Kleene star**, flipbits, reverse, etc.)
All roads lead to … regular sets?

Are there any languages over \{0,1\} that are not regular?

A. Yes: a language that is recognized by an NFA but not any DFA.

\[x\]

B. Yes: there is some finite language over \{0,1\} that is not described by any regular expression.

C. No: all languages over \{0,1\} are regular because that's what it means to be a language.

D. No: for each set of strings over \{0,1\}, some DFA recognizes that set.

E. None of the above.
For next time

- Work on Group Homework 2 due Saturday

Pre class-reading for Wednesday: page 77.
Counting languages

How many languages over \{0,1\} are there?

A. Finitely many because \{0,1\} is finite.
B. Finitely many because strings are finite.
C. Countably infinitely many because \{0,1\}^* is countably infinite.
D. Uncountably many because languages are in the power set of \{0,1\}^*.
E. None of the above.
Counting regular languages over \{0,1\}

| \{ regular languages\} | \leq | \{ regular expressions \} |

Each regular expression is a finite string over the alphabet

\{0, 1, \varepsilon, \emptyset, ( , ) , \cup , \ast\}

The set of strings over an alphabet is countably infinite.

Conclude: countably infinitely many regular languages.
Where we stand

Fact 1: There exist nonregular languages.

Fact 2: If we know some languages are nonregular, we can conclude others must be too. (cf. Discussion)

But, we don't have any specific examples of nonregular languages.

Yet.