Finite Automata

Part Two
Recap from Last Time
Old MacDonald Had a Symbol, \( \Sigma \)-eye-\( \varepsilon \)-ey\( \in \), Oh! ♫

• You may have noticed that we have several letter-E-ish symbols in CS103, which can get confusing!

• Here’s a quick guide to remembering which is which:

  • Typically, we use the symbol \( \Sigma \) to refer to an *alphabet*.

  • The *empty string* is length 0 and is denoted \( \varepsilon \).

  • In set theory, use \( \in \) to say “is an *element of*.”

  • In set theory, use \( \subseteq \) to say “is a *subset of*.”
DFAs

• A **DFA** is a
  • **D**eterministic
  • **F**inite
  • **A**utomaton

• DFAs are the simplest type of automaton that we will see in this course.
Recognizing Languages with DFAs

\[ L = \{ w \in \{ a, b \}^* \mid w \text{ contains } aa \text{ as a substring } \} \]
DFAs

• A DFA is defined relative to some alphabet $\Sigma$.

• For each state in the DFA, there must be exactly one transition defined for each symbol in $\Sigma$.
  
  • This is the “deterministic” part of DFA.

• There is a unique start state.

• There are zero or more accepting states.
New Stuff!
Which table best represents the transitions for the DFA shown below?

(A) 0 1
    q₀ q₁ q₀
    q₁ q₃ q₂
    q₂ q₃ q₀
    q₃ q₃ q₃

(B) 0 1
    q₀ q₀ q₁
    q₁ q₂ q₃
    q₂ q₀ q₃
    q₃ / / q₃

(C) 0 1 Σ
    q₀ q₁ q₀ / 
    q₁ q₃ q₂ / 
    q₂ q₃ q₀ / 
    q₃ / / q₃

Answer at PollEv.com/cs103 or text CS103 to 22333 once to join, then A, B, C, or D (none of the above).
Tabular DFAs

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tabular DFAs

<table>
<thead>
<tr>
<th>State</th>
<th>Input 0</th>
<th>Input 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_1$</td>
<td>$q_0$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_3$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_3$</td>
<td>$q_0$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$q_3$</td>
<td>$q_3$</td>
</tr>
</tbody>
</table>
Tabular DFAs

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>*q₀</td>
<td>q₁</td>
<td>q₀</td>
</tr>
<tr>
<td>q₁</td>
<td>q₃</td>
<td>q₂</td>
</tr>
<tr>
<td>q₂</td>
<td>q₃</td>
<td>q₀</td>
</tr>
<tr>
<td>*q₃</td>
<td>q₃</td>
<td>q₃</td>
</tr>
</tbody>
</table>
These stars indicate accepting states.
Since this is the first row, it's the start state.
My Turn to Code Things Up!

```c++
void kTransitionTable[kNumStates][kNumSymbols] = {
    {0, 0, 1, 3, 7, 1, ...},
    ...
};
bool kAcceptTable[kNumStates] = {
    false,
    true,
    true,
    ...
};
bool SimulateDFA(string input) {
    int state = 0;
    for (char ch: input) {
        state = kTransitionTable[state][ch];
    }
    return kAcceptTable[state];
}
```
The Regular Languages
A language $L$ is called a **regular language** if there exists a DFA $D$ such that $\mathcal{L}(D) = L$.

If $L$ is a language and $\mathcal{L}(D) = L$, we say that $D$ **recognizes** the language $L$. 
The Complement of a Language

• Given a language $L \subseteq \Sigma^*$, the complement of that language (denoted $\overline{L}$) is the language of all strings in $\Sigma^*$ that aren't in $L$.

• Formally:

$$\overline{L} = \Sigma^* - L$$
The Complement of a Language

• Given a language $L \subseteq \Sigma^*$, the complement of that language (denoted $\overline{L}$) is the language of all strings in $\Sigma^*$ that aren't in $L$.

• Formally:

$$\overline{L} = \Sigma^* - L$$
The Complement of a Language

- Given a language $L \subseteq \Sigma^*$, the *complement* of that language (denoted $\overline{L}$) is the language of all strings in $\Sigma^*$ that aren't in $L$.
- Formally:
  \[
  \overline{L} = \Sigma^* - L
  \]
The Complement of a Language

- Given a language $L \subseteq \Sigma^*$, the *complement* of that language (denoted $\overline{L}$) is the language of all strings in $\Sigma^*$ that aren't in $L$.

- Formally:

$$\overline{L} = \Sigma^* - L$$

Complements of Regular Languages

• As we saw a few minutes ago, a *regular language* is a language accepted by some DFA.

• **Question:** If $L$ is a regular language, is $\overline{L}$ necessarily a regular language?

• If the answer is “yes,” then if there is a way to construct a DFA for $L$, there must be some way to construct a DFA for $\overline{L}$.

• If the answer is “no,” then some language $L$ can be accepted by some DFA, but $\overline{L}$ cannot be accepted by any DFA.
Computational Device for $L$
Computational Device for $L$
Computational Device for $L$

input

Computational Device for $\overline{L}$

input
Computational Device for $L$

input

Yep!
Nope!

Computational Device for $\overline{L}$

input

Yep!
Nope!
Complementing Regular Languages

\[ L = \{ \ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \ \} \]

\[ \bar{L} = \{ \ w \in \{a, b\}^* \mid w \text{ does not contain } aa \text{ as a substring} \ \} \]
More Elaborate DFAs

$L = \{ w \in \{a, *, /\}^* \mid w \text{ represents a C-style comment} \}$
More Elaborate DFAs

\[ \overline{L} = \{ \ w \in \{a, *, /\}^* \mid w \text{ doesn't represent a C-style comment} \ \} \]
More Elaborate DFAs

\[ \overline{L} = \{ w \in \{a, *, /\}^* \mid w \text{ doesn't represent a C-style comment} \} \]
Closure Properties

- **Theorem:** If $L$ is a regular language, then $\overline{L}$ is also a regular language.
- As a result, we say that the regular languages are *closed under complementation*.