Finite Automata

Part Two
Recap from Last Time
Formal Language Theory

- An **alphabet** is a set, usually denoted $\Sigma$, consisting of elements called **characters**.

- A **string over $\Sigma$** is a finite sequence of zero or more characters taken from $\Sigma$.

- The **empty string** has no characters and is denoted $\varepsilon$.

- A **language over $\Sigma$** is a set of strings over $\Sigma$.

- The language $\Sigma^*$ is the set of all strings over $\Sigma$. 
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.
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.
The Language of an Automaton

- If $D$ is a DFA that processes strings over $\Sigma$, the **language of $D$**, denoted $\mathcal{L}(D)$, is the set of all strings $D$ accepts.

- Formally:

$$\mathcal{L}(D) = \{ \ w \in \Sigma^* \mid D \text{ accepts } w \ \}$$
New Stuff!
Recognizing Languages with DFAs

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

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

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

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

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

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

$L = \{ w \in \{a, b\}^* | w \text{ contains } aa \text{ as a substring } \}$
Recognizing Languages with DFAs

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

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

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

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

\[ L = \{ w \in \{a, *, /\}\* | w \text{ represents a C-style comment} \} \]

Let's have the \( a \) symbol be a placeholder for "some character that isn't a star or slash."

Let's design a DFA for C-style comments. Those are the ones that start with /* and end with */.

Accepted:
- /*aa*/
- /**/
- /***/
- /*aaa*aaa*/
- /*a/a*/

Rejected:
- /***/a/*aa*/
- aaa/***/aa
- /*
- /***/
- //aaaa
More Elaborate DFAs

\[ L = \{ w \in \{a, *, /\}^* \mid w \text{ represents a C-style comment} \} \]
Tabular DFAs

\[
\begin{array}{c|cc}
\text{0} & \text{1} \\
\hline
q_0 & \text{start} \\
q_1 & 0 \\
q_2 & 1 \\
q_3 & \Sigma \\
\end{array}
\]
Tabular DFAs

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>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>
Tabular DFAs

These stars indicate accepting states.
Tabular DFAs

Since this is the first row, it's the start state.
Tabular DFAs

Question to ponder: Why isn't there a column here for $\Sigma$?
int kTransitionTable[kNumStates][kNumSymbols] = {
    {0, 0, 1, 3, 7, 1, ...},
    ...
};

bool kAcceptTable[kNumStates] = {
    false,
    true,
    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:

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

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

Good proofwriting exercise: prove $\overline{L} = L$ for any language $L$. 
Complementing Regular Languages

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

\[ \overline{L} = \{ w \in \{a, b\}^* \mid w \text{ does not contain } aa \text{ as a substring} \} \]
Complementing Regular Languages

$L = \{ \ w \in \{a, *, /\}\* \mid w \text{ represents a C-style comment} \ \}$
Complementing Regular Languages

\( \overline{L} = \{ w \in \{a, *, /\}^* \mid w \text{ doesn't represent a C-style comment} \} \)
Complementing Regular Languages

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

Question to ponder: are the nonregular languages closed under complementation?
NFAs
Revisiting a Problem

\[ q_0 \rightarrow 1 \rightarrow q_1 \rightarrow 1 \rightarrow q_2 \]

\[ q_0 \rightarrow 0 \rightarrow 1 \rightarrow q_3 \]

\[ q_3 \rightarrow 0 \rightarrow 1 \rightarrow q_3 \]

Start state: \( q_0 \)
NFAs

• An *NFA* is a
  • *N*ondeterministic
  • *F*inite
  • *A*utomaton

• Structurally similar to a DFA, but represents a fundamental shift in how we'll think about computation.
(Non)determinism

- A model of computation is **deterministic** if at every point in the computation, there is exactly one choice that can make.
  - The machine accepts if that series of choices leads to an accepting state.
- A model of computation is **nondeterministic** if the computing machine has a finite number of choices available to make at each point, possibly including zero.
  - The machine accepts if *any* series of choices leads to an accepting state.
  - (This sort of nondeterminism is technically called **existential nondeterminism**, the most philosophical-sounding term we’ll introduce all quarter.)
A Simple NFA

\begin{figure}
\centering
\begin{tikzpicture}
\node[state, initial] (q0) {$q_0$};
\node[state] (q1) [right of=q0] {$q_1$};
\node[state, accepting] (q2) [right of=q1] {$q_2$};
\node[state] (q3) [below of=q2] {$q_3$};
\path[->]
(q0) edge node {$1$} (q1)
(q1) edge node {$1$} (q2)
(q0) edge node {$0, 1$} (q3)
(q2) edge[bend right] node {$0$} (q3)
(q3) edge[bend right] node {$0, 1$} (q0);
\end{tikzpicture}
\end{figure}
A Simple NFA

$q_0$ has two transitions defined on 1!
A Simple NFA
A Simple NFA

$q_0$ 1 $q_1$

$q_0$ 0, 1 $q_1$

$q_0$ 0 $q_3$

$q_3$ 0, 1 $q_2$

$q_3$ 0, 1

0 1 0 1 1
A Simple NFA
A Simple NFA

start

$q_0$  1  $q_1$

0, 1

$q_1$  1  $q_2$

$q_3$

0, 1

0, 1

0, 1

0 1 0 1 1
A Simple NFA
A Simple NFA

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

- Transition 1: \( q_0 \rightarrow q_1 \) with inputs 0, 1
- Transition 2: \( q_1 \rightarrow q_2 \) with input 1
- Loop: \( q_2 \rightarrow q_3 \) with inputs 0, 1
- Transition 3: \( q_3 \rightarrow q_0 \) with inputs 0, 1
A Simple NFA

\[ \begin{array}{c}
\text{start} \\
q_0 \quad 1 \\
q_1 \quad 1 \\
q_2 \\
q_3 \quad 0, 1 \\
\end{array} \]
A Simple NFA
A Simple NFA

\begin{tikzpicture}
  \node[state, initial] (q0) {$q_0$};
  \node[state] (q1) [right of=q0] {$q_1$};
  \node[state, accepting] (q2) [right of=q1] {$q_2$};
  \node[state] (q3) [below of=q2] {$q_3$};

  \draw[->] (q0) -- node[above] {$1$} (q1);
  \draw[->] (q1) -- node[above] {$1$} (q2);
  \draw[->] (q2) -- node[above] {$0, 1$} (q3);
  \draw[->] (q3) -- node[above] {$0, 1$} (q2);
  \draw[->] (q3) -- node[above, pos=0.25] {$0, 1$} (q3);
  \draw[->] (q0) -- node[below] {$0, 1$} (q1);

\end{tikzpicture}
A Simple NFA

\[
\begin{array}{c}
\text{start} \\
q_0 \\
q_1 \\
q_2 \\
q_3 \\
0, 1 \\
0, 1 \\
0, 1 \\
0, 1 \\
0, 1 \\
1 \\
1 \\
1 \\
1 \\
\end{array}
\]
A Simple NFA
A Simple NFA

0 1 0 1 1
A Simple NFA
A Simple NFA

\[ q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \]

Transition labels:
- \( q_0 \) to \( q_1 \) on 1
- \( q_1 \) to \( q_2 \) on 1
- \( q_3 \) on any input

Input string: 0 1 0 1 1
A Simple NFA

\[
\begin{align*}
&\text{start} \\
&\quad q_0 \quad 1 \quad q_1 \\
&\quad q_1 \quad 1 \quad q_2 \\
&\quad q_3 \\
&\quad 0, 1 \\
&\quad 0, 1 \\
&\quad 0, 1 \\
&\quad 0, 1 \\
&\quad 0, 1 \\
&\quad 0, 1 \\
&\quad 0, 1 \\
&0 \quad 1 \quad 0 \quad 1 \quad 1
\end{align*}
\]
A Simple NFA
A Simple NFA
A Simple NFA

```
q_0 1 q_1 1 q_2
0, 1

start

q_0 q_1 q_2

q_3
0, 1

0 1 0 1 1
```
A Simple NFA

Start → $q_0$ \(\xrightarrow{1} q_1\) \(\xrightarrow{1} q_2\)

$0, 1$

$0$

$0, 1$

$0, 1$

Input: $01011$
A Simple NFA

![NFA Diagram](image_url)
A Simple NFA

start

$q_0$ 1 $q_1$ 1 $q_2$

$q_0$, 1

$q_3$

$q_0$, 1

$q_0$, 0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1

0, 1
A Simple NFA

start

$q_0$ 1 $q_1$

$q_1$ 1 $q_2$

$q_3$

$q_0, 1$

$q_1$

$q_2$

$q_3$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$

$q_0, 1$
A Simple NFA

Start

$q_0 \xrightarrow{0,1} q_1 \xrightarrow{1} q_2$

$q_3 \xrightarrow{0,1} q_3$

Input: 0 1 0 1 1
A Simple NFA
A Simple NFA

- **Start state:** $q_0$
- **States:** $q_0$, $q_1$, $q_3$, $q_2$
- **Transitions:**
  - $q_0$ to $q_1$: $1$
  - $q_1$ to $q_0$: $0, 1$
  - $q_1$ to $q_2$: $1$
  - $q_2$ to $q_3$: $0, 1$
  - $q_3$ to $q_2$: $0, 1$

Input sequence: 0 1 0 1 1
A Simple NFA
A Simple NFA

![A Simple NFA Diagram](image)
A More Complex NFA
A More Complex NFA

If a NFA needs to make a transition when no transition exists, the automaton dies and that particular path does not accept.
A More Complex NFA

+ start \rightarrow q_0 \quad \text{1} \quad q_1 \quad \text{1} \quad q_2
+ q_0 \quad \text{0, 1} \quad q_1
+ 0 1 0 1 1
A More Complex NFA

```
0 1 0 1 1
```

Diagram:
- Start state: $q_0$
- Transitions:
  - $q_0$ on input 1 to $q_1$
  - $q_1$ on input 1 to $q_2$
  - $q_0$ on input 0, 1 to itself
- Final state: $q_2$
A More Complex NFA

start \rightarrow q_0 \rightarrow q_1 \rightarrow q_2

0, 1

0 1 0 1 1
A More Complex NFA

- Start state: $q_0$
- Transitions:
  - From $q_0$ to $q_1$: $1$
  - From $q_1$ to $q_2$: $1$
- Accepting state: $q_2$

Input string: $01011$
A More Complex NFA

Start state: $q_0$

Transitions:
- From $q_0$ to $q_1$ on input 1
- From $q_1$ to $q_2$ on input 1

Input string: 01011

Final state: $q_2$
A More Complex NFA
A More Complex NFA

Oh no! There's no transition defined!
A More Complex NFA

\[
\begin{array}{c}
q_0 \quad 1 \quad q_1 \quad 1 \quad q_2 \\
\text{start} \quad 0, 1
\end{array}
\]
A More Complex NFA

Start state: $q_0$

Transitions:
- $q_0 \xrightarrow{1} q_1$
- $q_1 \xrightarrow{1} q_2$
- $q_1 \xrightarrow{0, 1} q_1$

Input string: 01011
A More Complex NFA

Start

$q_0$ 1 $q_1$ 1 $q_2$

0, 1

0 1 0 1 1
A More Complex NFA

0, 1

0, 1, 0, 1, 1
A More Complex NFA
A More Complex NFA

start → $q_0$ (1) → $q_1$ (1) → $q_2$

Input: 0, 1, 0, 1, 1
A More Complex NFA

[start] $q_0$ \rightarrow 1 \rightarrow [0, 1] \rightarrow [1] \rightarrow q_2}

Input: 010111
A More Complex NFA

0 1 0 1 1
A More Complex NFA

start

$q_0$ 1 $q_1$ 1 $q_2$

0, 1

0 1 0 1 1
A More Complex NFA

\[ q_0 \xrightarrow{1} q_1, q_1 \xrightarrow{1} q_2 \]

Input sequence: 010111
A More Complex NFA
Hello, NFA!
Hello, NFA!
Hello, NFA!
Hello, NFA!

-start-

$q_0 \xrightarrow{h} q_1 \xrightarrow{i} q_2$

$h \, \, i$
Hello, NFA!
Hello, NFA!
Tragedy in Paradise

start

$q_0$  h  $q_1$  i  $q_2$

h i p
Tragedy in Paradise
Tragedy in Paradise

\[ \text{start} \quad q_0 \quad \xrightarrow{h} \quad q_1 \quad \xrightarrow{i} \quad q_2 \]

\[ \begin{align*} \text{h} & \quad \text{i} & \quad \text{p} \end{align*} \]
Tragedy in Paradise

Start

\( q_0 \)  \( q_1 \)  \( q_2 \)

\( h \)  \( i \)

\[ h \quad i \quad p \]
Tragedy in Paradise

start $\rightarrow q_0 \rightarrow h \rightarrow q_1 \rightarrow i \rightarrow q_2$

$h \quad i \quad p$

Tragedy in Paradise

Diagram:

- Start state: $q_0$
- Transition: $h$ from $q_0$ to $q_1$
- Transition: $i$ from $q_1$ to $q_2$

Words on tape: $hip$
Tragedy in Paradise

\[ q_0 \rightarrow q_1 \rightarrow \text{sad emoji} \]
Tragedy in Paradise

start $q_0$ \text{ h } q_1 \text{ i } q_2$

h i p

Image of an otter.
The **language of an NFA** is
\[ \mathcal{L}(N) = \{ w \in \Sigma^* \mid N \text{ accepts } w \} \].

What is the language of each NFA? (Assume \( \Sigma = \{a, b\} \).)

Note that flipping the accept and reject states of an NFA doesn’t always give an NFA for the complement of the original language. *Why?*

**Question to ponder:**
Why is the answer \( \{ w \in \Sigma^* \mid w \text{ ends in } aaa \} \) not correct?

\( \{ w \in \Sigma^* \mid w \text{ ends in } aa \} \)

\( \emptyset \)

\( \{ \varepsilon \} \)

\( \Sigma^* \)
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
\(\varepsilon\)-Transitions

- NFAs have a special type of transition called the \textbf{\(\varepsilon\)-transition}.
- An NFA may follow any number of \(\varepsilon\)-transitions at any time without consuming any input.
\(\varepsilon\)-Transitions

- NFAs have a special type of transition called the \textbf{\(\varepsilon\)-transition}.
- An NFA may follow any number of \(\varepsilon\)-transitions at any time without consuming any input.
ε-Transitions

• NFAs have a special type of transition called the **ε-transition**.

• An NFA may follow any number of ε-transitions at any time without consuming any input.
\( \varepsilon \)-Transitions

- NFAs have a special type of transition called the \( \varepsilon \)-transition.
- An NFA may follow any number of \( \varepsilon \)-transitions at any time without consuming any input.
\(\varepsilon\)-Transitions

- NFAs have a special type of transition called the \textbf{\(\varepsilon\)-transition}.
- An NFA may follow any number of \(\varepsilon\)-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

![Diagram of ε-transitions](image)
\(\epsilon\)-Transitions

- NFAs have a special type of transition called the **\(\epsilon\)-transition**.
- An NFA may follow any number of \(\epsilon\)-transitions at any time without consuming any input.
\textbf{ε-Transitions}

- NFAs have a special type of transition called the \textbf{ε-transition}.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
**ε-Transitions**

- NFAs have a special type of transition called the *ε-transition*.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
**ε-Transitions**

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
- NFAs are not *required* to follow ε-transitions. It's simply another option at the machine's disposal.
Time-Out For Announcements!
Problem Set Three Graded

• Your diligent and hardworking TAs have finished grading PS3. Grades and feedback are now available on Gradescope.

• As always, please review your feedback! Knowing where to improve is more important than just seeing a raw score.

• Did we make a mistake? Regrades are currently open on Gradescope and are due by next Monday.
Back to CS103!
Intuiting Nondeterminism

• Nondeterministic machines are a serious departure from physical computers. How can we build up an intuition for them?

• There are two particularly useful frameworks for interpreting nondeterminism:
  • *Perfect positive guessing*
  • *Massive parallelism*
Perfect Positive Guessing
Perfect Positive Guessing

Start: $q_0 \rightarrow a \rightarrow q_1 \rightarrow b \rightarrow q_2 \rightarrow a \rightarrow q_3$

Input: $\Sigma$

Sequence: a b a b a a
Perfect Positive Guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]

Start

a b a b a a
Perfect Positive Guessing

\[ \Sigma \]

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

\[ \text{start} \]

\[ a \quad b \quad a \quad b \quad a \quad b \quad a \]
Perfect Positive Guessing

\[ \sum \]

\( q_0 \rightarrow a \rightarrow q_1 \rightarrow b \rightarrow q_2 \rightarrow a \rightarrow q_3 \)

\[
\begin{array}{cccccc}
  a & b & a & b & a & a \\
\end{array}
\]
Perfect Positive Guessing

\[ \Sigma \]

\[ \text{start} \quad q_0 \quad q_1 \quad q_2 \quad q_3 \]

\[ a \quad b \quad a \quad b \quad a \]

Start at \( q_0 \) and follow the path labeled with \( a \) and \( b \).
Perfect Positive Guessing

Start: $q_0$

- $a$: $q_0 ightarrow q_1$
- $b$: $q_1 ightarrow q_2$
- $a$: $q_2 ightarrow q_3$

Input alphabet: $\Sigma$

Sequence: $abaaba$
Perfect Positive Guessing

\[
\begin{aligned}
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{aligned}
\]

Input: \(\Sigma\)
Perfect Positive Guessing

\[
\begin{align*}
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{align*}
\]

\[\Sigma\]

Start

a b b a b b a a
Perfect Positive Guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]

\[ a \quad b \quad a \quad b \quad a \quad b \quad a \]
Perfect Positive Guessing

\[ \begin{align*}
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{align*} \]

\( \Sigma \)
Perfect Positive Guessing

- We can view nondeterministic machines as having *Magic Superpowers* that enable them to guess choices that lead to an accepting state.
  - If there is at least one choice that leads to an accepting state, the machine will guess it.
  - If there are no choices, the machine guesses any one of the wrong guesses.
- There is no known way to physically model this intuition of nondeterminism – this is quite a departure from reality!
Massive Parallelism

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

\[ \Sigma \]

\[ a \ b \ a \ b \ a \ b \ a \ a \]
Massive Parallelism
Massive Parallelism

\[ \sum \]

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

\[ \begin{array}{cccccc}
  a & b & a & b & a & a \\
\end{array} \]
Massive Parallelism

\[ \Sigma \]

q₀ → q₁ → q₂ → q₃

\[ \text{a b a b a b a} \]
Massive Parallelism

\\quad q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3

\begin{align*}
\begin{array}{cccccccc}
a & b & a & b & a & b & a \\
\end{array}
\end{align*}
Massive Parallelism

$a \ b \ a \ b \ a \ b \ a$
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Input: \( a \ b \ a \ b \ a \)
Massive Parallelism

\[
\begin{align*}
\Sigma & \\
q_0 & \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3
\end{align*}
\]

\[
\begin{array}{cccccc}
 a & b & a & b & a & a \\
\end{array}
\]
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]

Input sequence: \[ \text{a b b a b b a a} \]

Start state: \[ q_0 \]
Massive Parallelism

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

Start

\[ \Sigma \]

Input sequence: a b a b a b a
Massive Parallelism

\[ q_0 \xrightarrow{\text{a}} q_1 \xrightarrow{\text{b}} q_2 \xrightarrow{\text{a}} q_3 \]

\[ \Sigma \]

\[ \text{start} \]

\[ a \ b \ a \ b \ a \ b \ a \]
Massive Parallelism

\[ a \quad b \quad b \quad a \quad b \quad a \quad a \]
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Input sequence: \(abaabbaa\)
Massive Parallelism

\[
\Sigma
\]

start

\[
\begin{array}{c}
q_0 \\
q_1 \\
q_2 \\
\vdots \\
q_3
\end{array}
\]

\[
\begin{array}{c}
a \\
b \\
a \\
\vdots \\
a
\end{array}
\]

a b a b a b a
Massive Parallelism

start

$q_0$ → $q_1$ → $q_2$ → $q_3$

$a$ $b$ $a$ $b$ $a$ $b$ $a$
Massive Parallelism

\[
\begin{align*}
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{align*}
\]

Input: \(a \ b \ a \ b \ a \ b \ a\)
Massive Parallelism

\[ \sum \]

\begin{align*}
q_0 & \rightarrow a \rightarrow q_1 \rightarrow b \rightarrow q_2 \rightarrow a \rightarrow q_3 \\
\text{start} & \rightarrow q_0
\end{align*}

Input: \(ababaab\)
Massive Parallelism

start

$q_0$ $a$ $q_1$ $b$ $q_2$ $a$ $q_3$

$\Sigma$

a b a b a b a
Massive Parallelism

Figure: A diagram illustrating a state transition with symbols 'a', 'b', and a transition from $q_0$ to $q_1$, $q_1$ to $q_2$, and $q_2$ to $q_3$. The transition symbols are labeled as $\Sigma$, 'a', and 'b' respectively. The sequence of symbols 'a b b a b a a' is shown at the bottom.
Massive Parallelism

\[ \Sigma a b a b a a \]
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Start state: \( q_0 \)

Input alphabet: \( \Sigma = \{a, b\} \)
Massive Parallelism

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

\[ \sum \rightarrow a \rightarrow b \rightarrow q_2 \rightarrow a \rightarrow q_3 \]

Input sequence:

\[ a \ b \ a \ b \ a \ b \ a \]
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Input string: a b a b a a
Massive Parallelism

\[
\begin{align*}
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{align*}
\]
Massive Parallelism

\[ \sum a \\ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Input: \texttt{abaaba}
Massive Parallelism

\[ \sum \]

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

Input: \[ a b a b a a \]
Massive Parallelism
Massive Parallelism
Massive Parallelism

\[ \sum \]

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

Input sequence: \[ \text{a b a b a b a} \]
Massive Parallelism

\[ q_3 \]

\[ q_2 \]

\[ q_1 \]

\[ q_0 \]

\[ \sum \]

start

\[ a \]

\[ b \]

\[ a \]

\[ a \]

\[ b \]

\[ a \]

\[ a \]

\[ a \]
Massive Parallelism

We're in at least one accepting state, so there's some path that gets us to an accepting state.
Massive Parallelism

\[ \Sigma \]

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ a \ b \ a \ b \ b \]
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Input: \( \Sigma \)

States: \( q_0, q_1, q_2, q_3 \)

Transitions:
- \( q_0 \xrightarrow{a} q_1 \)
- \( q_1 \xrightarrow{b} q_2 \)
- \( q_2 \xrightarrow{a} q_3 \)
- \( q_3 \xrightarrow{a} q_3 \) (loop)

Input sequence: a b a b a b
Massive Parallelism

\[ q_0 \xrightarrow{\sum} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_3 \]

\[
\begin{array}{cccc}
\text{a} & \text{b} & \text{a} & \text{b} \\
\end{array}
\]
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]

\[ a \ b \ a \ b \ a \ b \]

\[ \uparrow \]
Massive Parallelism

\[ \Sigma \]

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \begin{array}{cccccc}
q_0 & q_1 & q_2 & q_3 \\
\text{start} & a & b & a
\end{array} \]
Massive Parallelism

\[
\begin{align*}
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{align*}
\]

\[\Sigma\]

\[
\begin{array}{cccc}
a & b & a & b \\
\end{array}
\]
Massive Parallelism

a b a b b

start

$q_0$ $q_1$ $q_2$ $q_3$

$\Sigma$
Massive Parallelism

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \]

\[ \Sigma \]

\[
\begin{align*}
q_0 & \rightarrow a \rightarrow q_1 \\
q_1 & \rightarrow b \rightarrow q_2 \\
q_2 & \rightarrow a \rightarrow q_3
\end{align*}
\]

\[
\begin{array}{cccc}
a & b & a & b \\
\end{array}
\]
Massive Parallelism

\[ \sum \]

\[
\text{start} \quad q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3
\]

\[
\begin{array}{cccccc}
\text{a} & \text{b} & \text{a} & \text{b} & \text{a} & \text{b}
\end{array}
\]
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]

Start

\[
\begin{array}{cccc}
\text{a} & \text{b} & \text{a} & \text{b} \\
\end{array}
\]
Massive Parallelism

\[
\sum \quad q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3
\]

\[
a \quad b \quad a \quad b \quad a \quad b
\]
Massive Parallelism

\[ Q_0 \rightarrow a \rightarrow Q_1 \rightarrow b \rightarrow Q_2 \rightarrow a \rightarrow Q_3 \]

\[ \Sigma \]

\[ \text{a b a a b b} \]
Massive Parallelism
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]

Start

[Sequence: a b a b b b]
Massive Parallelism

\[
\begin{align*}
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{align*}
\]

\[\Sigma\]

Input sequence: \[a \ b \ a \ b \ a \ b\]
Massive Parallelism

\[ q_0, q_1, q_2, q_3 \]

\[ \sum \]

\[ a, b, a, b \]

\[ \text{start} \]
Massive Parallelism

\[ \begin{align*}
q_0 & \xrightarrow{a} q_1 & q_1 & \xrightarrow{b} q_2 & q_2 & \xrightarrow{a} q_3 \\
\text{start} & & & & & \Sigma
\end{align*} \]
Massive Parallelism
Massive Parallelism

start

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$

$a b a b a b$

∑
Massive Parallelism

a b a b b
Massive Parallelism

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Input: \( \Sigma \)

States: \( q_0, q_1, q_2, q_3 \)

Transitions: \( a, b \)

Starting state: \( q_0 \)

Ending state: \( q_3 \)

Words accepted: \( ababab \)
Massive Parallelism

\[
\sum \rightarrow q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3
\]
Massive Parallelism

\[ \sum \]

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ a \ b \ a \ b \]
We're not in any accepting state, so no possible path accepts.
Massive Parallelism

• An NFA can be thought of as a DFA that can be in many states at once.

• At each point in time, when the NFA needs to follow a transition, it tries all the options at the same time.

• (Here's a rigorous explanation about how this works; read this on your own time).

  • Start off in the set of all states formed by taking the start state and including each state that can be reached by zero or more $\varepsilon$-transitions.

  • When you read a symbol $a$ in a set of states $S$:
    - Form the set $S'$ of states that can be reached by following a single $a$ transition from some state in $S$.
    - Your new set of states is the set of states in $S'$, plus the states reachable from $S'$ by following zero or more $\varepsilon$-transitions.
Just how powerful are NFAs?
Next Time

- **The Powerset Construction**
  - So beautiful. So elegant. So cool!
- **More Closure Properties**
  - Other set-theoretic operations.
- **Language Transformations**
  - What’s the deal with the notation $\Sigma^*$?