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 consists of:
  • A set of states
  • Exactly one element of the set of states designated as a start state
    - (as a consequence, the set of states must be nonempty)
  • A subset of the states designated as accepting states
  • An alphabet $\Sigma$
  • A transition function that maps (state, character) ordered pairs to states
    - (i.e., for each state in the DFA, there must be *exactly one* transition defined for each symbol in $\Sigma$)
The Language of an Automaton

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

\[ \mathcal{A}(D) = \{ w \in \Sigma^* \mid D \text{ accepts } w \} \]
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

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

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

Diagram:
- Start state: $q_0$
- Transitions:
  - $q_0 \xrightarrow{a} q_1$
  - $q_0 \xrightarrow{b} q_0$
  - $q_1 \xrightarrow{a} q_2$
  - $q_1 \xrightarrow{b} q_1$
  - $q_2 \xrightarrow{a} q_2$
  - $q_2 \xrightarrow{b} q_2$
  - $q_2 \xrightarrow{a,b}$

Final state: $q_2$
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

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

Let’s have the \textbf{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:

- /*a*/
- /**/
- /***/
- /*aaa*/aaa*/
- /*a/a*/

Rejected:

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

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

*Start state*: $q_0$

- Transition on 1: $q_0 \rightarrow q_1$
- Transition on $\theta$: $q_1 \rightarrow q_2$
- Transition on 1: $q_2 \rightarrow q_3$
- Transition on $\Sigma$: $q_3 \rightarrow q_0$

<table>
<thead>
<tr>
<th>State</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
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.
Tabular DFAs

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

<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

Question to ponder: Why isn’t there a column here for $\Sigma$?
Code? In a Theory Class?

```cpp
int 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 \textit{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 $\bar{L}$) is the language of all strings in $\Sigma^*$ that aren't in $L$.

Formally:

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

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

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

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

$L = \{ w \in \{a, *, /\}^* | w \text{ represents a C-style comment} \}$
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 $\bar{L}$ 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
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
A Simple NFA

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

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

\[ 0, 1 \rightarrow 0, 1 \]

\[ 0, 1 \rightarrow 0, 1 \]

\[ 0, 1 \rightarrow 0, 1 \]

\[ 0 \rightarrow 0 \]

\[ 1 \rightarrow 1 \]

\[ 1 \rightarrow 1 \]

\[ 1 \rightarrow 1 \]

\[ 1 \rightarrow 1 \]

\[ 1 \rightarrow 1 \]

\[ 1 \rightarrow 1 \]
A Simple NFA
A Simple NFA

Start state: $q_0$

States: $q_0, q_1, q_2, q_3$

Transitions:
- $q_0 \xrightarrow{0, 1} q_3$
- $q_0 \xrightarrow{1} q_1$
- $q_1 \xrightarrow{1} q_2$
- $q_1 \xrightarrow{0} q_3$
- $q_2 \xrightarrow{0, 1} q_3$
- $q_3 \xrightarrow{0, 1} q_0$

Input alphabet: $\{0, 1\}$

Sample input: $010111$
A Simple NFA

start

$q_0$ 1 $q_1$

$q_1$ 1 $q_2$

$q_3$

0, 1

0

0, 1

0, 1

0, 1

0 1 0 1 1
A Simple NFA

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

```
0 1 0 1 1
```
A Simple NFA

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

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

\[ q_3 \xrightarrow{0,1} q_2 \]

\[ q_3 \xrightarrow{0,1} q_3 \]

\[ 0 1 0 1 1 \]
A Simple NFA

Start

$q_0$ -> $q_1$ with 1
$q_1$ -> $q_2$ with 1

$q_3$ with 0, 1

Input:

```
0 1 0 1 1
```
A Simple NFA

0 1 0 1 1
A Simple NFA

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

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

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

\[ q_3 \rightarrow \emptyset, 1 \]

[0 1 0 1 1]
A Simple NFA

\[
\begin{array}{c}
\text{start} \\
q_0 \quad 1 \rightarrow q_1 \quad 1 \rightarrow q_2 \\
\theta, 1 \rightarrow q_0 \\
\theta, 1 \rightarrow q_3 \\
\theta, 1 \rightarrow q_2
\end{array}
\]

0 1 0 1 1 1
A Simple NFA

\[
\begin{array}{c}
q_0 \xrightarrow{1} q_1 \\
q_1 \xrightarrow{1} q_2 \\
q_0 \xrightarrow{\emptyset, 1} q_3 \\
q_1 \xrightarrow{\emptyset} q_3 \\
q_2 \xrightarrow{\emptyset, 1} q_3 \\
q_3 \xrightarrow{\emptyset, 1} q_0 \\
\end{array}
\]
A Simple NFA
A Simple NFA

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

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

start

$q_0 \xrightarrow{\empty, 1} q_1 \xrightarrow{1} q_2 \xrightarrow{\empty, 1} q_3 \xrightarrow{\empty, 1} q_3

\begin{array}{lllll}
0 & 1 & 0 & 1 & 1
\end{array}
A Simple NFA

[Diagram of a non-deterministic finite automaton (NFA) with states q0, q1, q2, and q3. The transitions are labeled with 0, 1, and \(\emptyset, 1\).]
A Simple NFA

start

$q_0$ 1 $q_1$

0 1 $q_2$

$q_3$

0 1 0 1 1
A Simple NFA

start

$q_0 \rightarrow q_1 \rightarrow q_2$

$q_0 \rightarrow q_3$

$q_1 \rightarrow q_2$

$q_1 \rightarrow q_3$

$q_2 \rightarrow q_3$

$q_3 \rightarrow q_2$

$q_3 \rightarrow q_0$

Input: 0 1 0 1 1
A Simple NFA

\begin{center}
\begin{tikzpicture}
    
    % Define nodes
    \node[state, initial] (q0) at (0,0) {$q_0$};
    \node[state, accepting] (q1) at (2,0) {$q_1$};
    \node[state, accepting] (q2) at (4,0) {$q_2$};
    \node[state] (q3) at (2,-2) {$q_3$};

    % Draw transitions
    \draw[->] (q0) edge node {$1$} (q1);
    \draw[->] (q1) edge node {$1$} (q2);
    \draw[->] (q0) edge node {$\emptyset, 1$} (q3);
    \draw[->] (q2) edge node {$\emptyset, 1$} (q3);
    \draw[->] (q3) edge[loop below] node {$\emptyset, 1$} (q3);

    \end{tikzpicture}
\end{center}
A Simple NFA

- **start**
- $q_0 \xrightarrow{0, 1} q_3$
- $q_0 \xrightarrow{1} q_1$
- $q_1 \xrightarrow{1} q_2$
- $q_3 \xrightarrow{\emptyset, 1} q_2$
- $q_3 \xrightarrow{0} q_3$
- $q_2$ is an accepting state.

Input:
- $\emptyset 1 0 1 1$
A Simple NFA

start \rightarrow q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2

q_2 \xrightarrow{\emptyset,1} q_3 \xrightarrow{\emptyset,1} q_2

0 1 0 1 1

SEAL

OF APPROVAL
A More Complex NFA

\[
\begin{array}{c}
\text{start} \\
q_0 \quad 1 \quad q_1 \quad 1 \quad q_2
\end{array}
\]

\[\begin{align*}
\text{Transition} & : \\
q_0 & \rightarrow q_1 \\
q_1 & \rightarrow q_2 \\
\end{align*}\]
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

0 1 0 1 1
A More Complex NFA

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

\[ \emptyset, 1 \]

0 1 0 1 1
A More Complex NFA

A More Complex NFA

start

$q_0$ 1 $q_1$ 1 $q_2$

0, 1

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

The diagram shows a non-deterministic finite automaton (NFA) with states $q_0$, $q_1$, and $q_2$. The transitions are as follows:

- From $q_0$ to $q_1$ with input $1$
- From $q_1$ to $q_2$ with input $1$
- $q_0$ is the start state.

The input sequence is $01011$. The automaton transitions through the states as follows:

- Start at $q_0$.
- Move to $q_1$ with input $1$.
- Move to $q_2$ with input $1$.

The automaton accepts the input sequence $01011$.
A More Complex NFA

\[\begin{array}{ccc}
q_0 & \xrightarrow{1} & q_1 \\
q_1 & \xrightarrow{1} & q_2 \\
\end{array}\]
A More Complex NFA

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

\[
\begin{align*}
&
\text{start} \\ &
q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \\
&
q_0 \xrightarrow{\emptyset, 1} q_0
\end{align*}
\]
A More Complex NFA
A More Complex NFA

\begin{center}
\begin{tikzpicture}
    \node[state,initial] (q0) {$q_0$};
    \node[state,accepting] (q1) [right of=q0] {$q_1$};
    \node[state,accepting] (q2) [right of=q1] {$q_2$};
    \path[->] (q0) edge node {$0, 1$} (q1);
    \path[->] (q1) edge node {$1$} (q2);
    \path[->] (q2) edge[loop above] node {$0, 1$} (q2);
\end{tikzpicture}
\end{center}

\[
\begin{array}{cccc}
0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]
A More Complex NFA

start

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

\[ \emptyset, 1 \]

\[ 0 \, 1 \, 0 \, 1 \, 1 \]
A More Complex NFA
A More Complex NFA
A More Complex NFA

\[
\begin{align*}
\text{start} & \rightarrow q_0 \quad 1 \rightarrow q_1 \quad 1 \rightarrow q_2 \\
q_0 & \xrightarrow{\emptyset, 1} q_1
\end{align*}
\]
A More Complex NFA
A More Complex NFA

\begin{center}
\begin{tikzpicture}
  \node[state, initial] (q0) at (0,0) {$q_0$};
  \node[state] (q1) at (2,0) {$q_1$};
  \node[state, accepting] (q2) at (4,0) {$q_2$};
  \draw (q0) edge[above] node {$1$} (q1);
  \draw (q1) edge[above] node {$1$} (q2);
  \draw (q0) edge[loop below] node {$0, 1$} (q0);
\end{tikzpicture}
\end{center}

\begin{center}
\begin{tabular}{c c c c}
0 & 1 & 0 & 1 & 1
\end{tabular}
\end{center}
A More Complex NFA
Hello, NFA!

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

h i
Hello, NFA!
Hello, NFA!

start \rightarrow q_0 \xrightarrow{h} q_1 \xrightarrow{i} q_2

h | i
Hello, NFA!
Hello, NFA!

![Diagram of an NFA with states $q_0$, $q_1$, and $q_2$. The transitions are labeled with the letters h and i. The start state is $q_0$, and the accepting state is $q_2$.]
Hello, NFA!

Start state $q_0$ transitions to $q_1$ on input $h$, and $q_1$ transitions to the accepting state $q_2$ on input $i$. The seal in the image of approval is a humorous element.
Tragedy in Paradise
Tragedy in Paradise

Diagram:
- Start state: $q_0$
- Transition: $h \rightarrow q_1$
- Transition: $i \rightarrow q_2$
- Transition: $h \rightarrow h i p$
Tragedy in Paradise
Tragedy in Paradise
Tragedy in Paradise

\[
\begin{array}{c}
\text{start} \\
q_0 \xrightarrow{h} q_1 \xrightarrow{i} q_2
\end{array}
\]
Tragedy in Paradise

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

\[ \text{h i p} \]
Tragedy in Paradise

\[
\begin{array}{ccc}
q_0 & \xrightarrow{h} & q_1 \\
\xrightarrow{\text{start}} & & \xrightarrow{i}
\end{array}
\]

\[
\begin{array}{c}
\text{h}
\end{array}
\]

\[
\begin{array}{c}
\text{i}
\end{array}
\]

\[
\begin{array}{c}
\text{p}
\end{array}
\]
Tragedy in Paradise
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 \( \{ w \in \Sigma^* \mid w \text{ ends in } \text{aaa} \} \) not correct?

\[ \{ w \in \Sigma^* \mid w \text{ ends in } \text{aa} \} \]
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.
\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.

![NFA Diagram]

The state transitions are as follows:
- From $q_0$ to $q_1$: $a$.
- From $q_1$ to $q_2$: $a$.
- From $q_3$ to $q_4$: $b, \varepsilon$.
- From $q_4$ to $q_5$: $b$.
- From $q_5$ to $q_5$: $b$. 

Input sequence: **baabb**
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.
\(\varepsilon\)-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.
\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.

---

Start

\[
\begin{align*}
q_0 &\xrightarrow{a} q_1 & q_1 &\xrightarrow{a} q_2 \\
&\xleftarrow{\varepsilon} q_3 & q_3 &\xrightarrow{b, \varepsilon} q_4 & q_4 &\xrightarrow{b} q_5 \\
&\xleftarrow{\varepsilon} & q_4 &\xrightarrow{b} q_5 \\
\end{align*}
\]
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.

Not at all fun or rewarding exercise: what is the language of this NFA?
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.
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

\[
\begin{align*}
\Sigma \\
\rightarrow q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3
\end{align*}
\]
Perfect Positive Guessing

\[ \Sigma \]

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

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

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

\[ \Sigma \]

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

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

\[ \Sigma \]

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

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

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

\[\Sigma\]

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

\[ \Sigma \]

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

Start

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

$\Sigma$

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

a b a b a b a
Perfect Positive Guessing

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

\[ \Sigma \]

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

Σ

\[
\begin{align*}
\text{start} & \quad \rightarrow q_0 \\
q_0 & \quad \rightarrow q_1 \quad \text{a} \\
qu_1 & \quad \rightarrow q_2 \quad \text{b} \\
qu_2 & \quad \rightarrow q_3 \quad \text{a}
\end{align*}
\]
Perfect Positive Guessing

\[ \Sigma \]

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

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

SEAL OF APPROVAL
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

\[ \Sigma \]

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

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

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

Sequence: a b a b a a
Massive Parallelism

Σ

start

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

$ababa$
Massive Parallelism

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

Input: \text{a b a b a a}
Massive Parallelism

\[ \sum \]

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

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

\[ \Sigma \]

start

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

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

\[ \uparrow \]
Massive Parallelism

\[ \Sigma \]

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

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

\[ \sum \]

1. \( q_0 \) (start)
2. \( q_1 \) (a)
3. \( q_2 \) (b)
4. \( q_3 \) (a)

Input: \( \text{a b a b a a} \)
Massive Parallelism

\[ \Sigma \]

State Diagram:
- Start state: \( q_0 \)
- Transitions:
  - From \( q_0 \) to \( q_1 \) on input \( a \)
  - From \( q_1 \) to \( q_2 \) on input \( b \)
  - From \( q_2 \) to \( q_3 \) on input \( a \)
  - Loop from \( q_3 \) to \( q_0 \) on input \( \Sigma \)

Input Sequence: \( a \ b \ a \ b \ a \ a \)

Next State: \( q_3 \)
Massive Parallelism

\[
\Sigma
\]

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

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

\[ \sum \]

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

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

\[ \sum \]

Start

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

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

\[ \Sigma \]

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

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

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

\[ \Sigma \]

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

\[ \sum \]

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

Input: \text{a b a b a a}
Massive Parallelism

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

\[ \Sigma \]

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

\[
\Sigma
\]

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

Input: a b a b a a
Massive Parallelism

\[ \Sigma \]

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

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

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

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

\[
\Sigma
\]

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

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

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

\[ \sum \]

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

\[ \uparrow \]
Massive Parallelism

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

Σ

start

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

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

q₀ \rightarrow a \rightarrow q₁ \rightarrow b \rightarrow q₂ \rightarrow a \rightarrow q₃

Σ

a b a b b a
Massive Parallelism
Massive Parallelism

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

\[ \Sigma \]

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

\[ \begin{array}{c}
a \\
b \\
a \\
\text{start}
\end{array} \]

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

Σ

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

$\text{a b a b a a}$
We're in at least one accepting state, so there's some path that gets us to an accepting state.

```
a b a b a a
```
Massive Parallelism

\[ \Sigma \]

- Start: \( q_0 \) → \( q_1 \) (a) → \( q_2 \) (b) → \( q_3 \) (a)
- Input: a b a b
Massive Parallelism

\[ \Sigma \]

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

Input sequence:

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

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

\[ \Sigma \]

Input sequence: a b a b b
Massive Parallelism

\[ \Sigma \]

start

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

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

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

\[ \Sigma \]

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

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

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

\begin{array}{c}
\begin{array}{c}
a \\
b \\
a \\
b \\
b\\end{array}
\end{array}
Massive Parallelism

\[ \Sigma \]

start

\[ q_0 \] → \[ q_1 \] \[ a \] → \[ q_2 \] \[ b \] → \[ q_3 \] \[ a \] → \[ q_3 \] (loop)

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

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

Input: $\Sigma$

Sequence: $a \ b \ a \ b$
Massive Parallelism

\[ \Sigma \]

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

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

\[ \Sigma \]

start

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

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

Σ

start

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

a b a b b
Massive Parallelism

\[
\Sigma
\]

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

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

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

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

\[ \Sigma \]

\[ \text{start} \]

\[ a \ b \ a \ b \]

\[ \downarrow \]
Massive Parallelism
Massive Parallelism

Σ

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

\[ \sum \]

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

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

\}

Massive Parallelism

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

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

Σ

a b a b

q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3
Massive Parallelism

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

\[
\Sigma
\]

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

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

\[ \Sigma \]

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

\[ a \ b \ a \ b \]

\[ \uparrow \]

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

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

\[\Sigma\]

a b a b
Massive Parallelism

We're not in any accepting state, so no possible path accepts.

\[ a \quad b \quad a \quad b \]
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 ε-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 ε-transitions.
Designing NFAs
Designing NFAs

- *Embrace the nondeterminism!*

- Good model: *Guess-and-check*:
  - Is there some information that you'd really like to have? Have the machine *nondeterministically guess* that information.
  - Then, have the machine *deterministically check* that the choice was correct.

- The *guess* phase corresponds to trying lots of different options.

- The *check* phase corresponds to filtering out bad guesses or wrong options.
Guess-and-Check

$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$
$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$
Guess-and-Check

\[ L = \{ w \in \{0, 1\}^* | w \text{ ends in } 010 \text{ or } 101 \} \]

Nondeterministically **guess** when the end of the string is coming up.

Deterministically **check** whether you were correct.
Guess-and-Check

\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
Guess-and-Check

\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
Guess-and-Check

\[ L = \{ \ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
Guess-and-Check

$L = \{ \ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \ \}$
Guess-and-Check

\[ L = \{ \, w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \, \} \]
Guess-and-Check

\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
Guess-and-Check

\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
Guess-and-Check

$L = \{ \ w \in \{0, 1\}^* \ | \ w \text{ ends in 010 or 101} \ \}$
Guess-and-Check

\[ L = \{ \, w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \, \} \]
Guess-and-Check

\[ L = \{ \; w \in \{0, 1\}^* \; | \; \text{w ends in } 010 \text{ or } 101 \; \} \]
Guess-and-Check

\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]
$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$
Guess-and-Check

\[ L = \{ \ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]

Nondeterministically guess which character is missing.

Deterministically check whether that character is indeed missing.
Guess-and-Check

\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]
Guess-and-Check

\[ L = \{w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]
Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$
Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$
Guess-and-Check

\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]
Guess-and-Check

\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]
Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$
Guess-and-Check

\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]
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 $\Sigma^*$?