Recap from Last Time
NFAs

- An **NFA** is a
  - **N**ondeterministic
  - **F**inite
  - **A**utomaton
- NFAs have no restrictions on how many transitions are allowed per state.
- They can also use ε-transitions.
- An NFA accepts a string $w$ if there is some sequence of choices that leads to an accepting state.
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.

• The NFA accepts if *any* of the states that are active at the end are accepting states. It rejects otherwise.
New Stuff!
Just how powerful are NFAs?
NFAs and DFAs

● Any language that can be accepted by a DFA can be accepted by an NFA.

● Why?
  ● Every DFA essentially already is an NFA!

● **Question**: Can any language accepted by an NFA also be accepted by a DFA?

● Surprisingly, the answer is **yes**!
**Thought Experiment:**
How would you simulate an NFA in software?
\[
\begin{align*}
\Sigma &
\end{align*}
\]

<table>
<thead>
<tr>
<th>State Set</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>{q_0}</td>
<td>a</td>
<td>{q_0, q_1}</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>
\[
\begin{array}{|c|c|}
\hline
q_0 & q_1 & q_2 & q_3 \\
\hline
\{q_0\} & \{q_0, q_1\} & & \\
\hline
\hline
\end{array}
\]
\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Transition table:

<table>
<thead>
<tr>
<th>State</th>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>({q_0})</td>
<td>({q_0, q_1})</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>---</td>
</tr>
</tbody>
</table>
\[
\begin{array}{ccc}
\{q_0\} & \{q_0, q_1\} & \{q_0\} \\
\end{array}
\]
<table>
<thead>
<tr>
<th>State Set</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>{q_0}</td>
<td>{q_0, q_1}</td>
<td>{q_0}</td>
</tr>
<tr>
<td>{q_0, q_1}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
$\Sigma$

<table>
<thead>
<tr>
<th>State</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[
\begin{array}{c|c|c}
\Sigma & a & b \\
\hline
\{ q_0 \} & \{ q_0, q_1 \} & \{ q_0 \} \\
\{ q_0, q_1 \} & & \\
\end{array}
\]
State transitions:
- Start state: $q_0$
- From $q_0$ on $a$: $q_1$
- From $q_1$ on $b$: $q_2$
- From $q_2$ on $a$: $q_3$

Transitions for symbols $a$ and $b$:

<table>
<thead>
<tr>
<th>State</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The given DFA has the following states and transitions:

- States: $q_0, q_1, q_2, q_3$
- Start state: $q_0$
- Accept state: $q_3$

Transitions:
- From $q_0$: $a$ goes to $q_1$
- From $q_1$: $b$ goes to $q_2$
- From $q_2$: $a$ goes to $q_3$

The table represents the transition function:

<table>
<thead>
<tr>
<th>Current State</th>
<th>a Transition</th>
<th>b Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td></td>
</tr>
</tbody>
</table>

The diagram shows a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are as follows:
- From $q_0$ to $q_1$ on input $a$.
- From $q_1$ to $q_2$ on input $b$.
- From $q_2$ to $q_1$ on input $a$.
- From $q_2$ to $q_3$ on input $a$.
- The state $q_3$ is a sink state, meaning it loops back to itself on any input.

The table represents the transition function of the automaton.
<table>
<thead>
<tr>
<th>States</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
</tbody>
</table>
\begin{align*}
\begin{array}{|c|c|c|}
\hline
 & a & b \\
\hline
\{ q_0 \} & \{ q_0, q_1 \} & \{ q_0 \} \\
\{ q_0, q_1 \} & \{ q_0, q_1 \} & \{ q_0, q_2 \} \\
\hline
\end{array}
\end{align*}
\[
\begin{align*}
\Sigma & \quad a \quad b \\
\{q_0\} & \{q_0 \, q_1\} & \{q_0\} \\
\{q_0 \, q_1\} & \{q_0 \, q_1\} & \{q_0 \, q_2\} \\
\{q_0 \, q_2\} & & \\
\end{align*}
\]
<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>{q_0}</td>
<td>{q_0, q_1}</td>
<td>{q_0}</td>
</tr>
<tr>
<td>{q_0, q_1}</td>
<td>{q_0, q_1}</td>
<td>{q_0, q_2}</td>
</tr>
<tr>
<td>{q_0, q_2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The automaton starts in state $q_0$. From $q_0$, on input $a$, it transitions to $q_1$, and on input $b$, it transitions to $q_2$. From $q_2$, on input $a$, it loops back to $q_3$. The table below shows the transitions for inputs $a$ and $b$.

<table>
<thead>
<tr>
<th>Current State</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>{q_0}</td>
<td>{q_0, q_1}</td>
<td>{q_0}</td>
</tr>
<tr>
<td>{q_0, q_1}</td>
<td>{q_0, q_1}</td>
<td>{q_0, q_2}</td>
</tr>
<tr>
<td>{q_0, q_2}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \(q_0\) is the start state.
- \(q_3\) is the accepting state.
- Transitions:
  - \(\Sigma\) from \(q_0\) to \(q_1\) on any input.
  - \(a\) from \(q_0\) to \(q_1\) and from \(q_2\) to \(q_3\).
  - \(b\) from \(q_1\) to \(q_2\).

The table shows the transitions for states \(q_0, q_1, q_2, q_3\) and inputs \(a, b\).
The diagram represents a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are as follows:

1. From $q_0$ to $q_1$ on input $a$.
2. From $q_1$ to $q_2$ on input $b$.
3. From $q_2$ to $q_3$ on input $a$.
4. The loop from $q_3$ back to $q_0$ on any input symbol $\Sigma$.

The table below shows the transition function for states $q_0$, $q_1$, $q_2$, and $q_3$:

<table>
<thead>
<tr>
<th>State</th>
<th>Input $a$</th>
<th>Input $b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td>${q_0, q_1, q_3}$</td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td>${q_0, q_1, q_3}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>{q_0}</td>
<td>{q_0, q_1}</td>
<td>{q_0}</td>
</tr>
<tr>
<td>{q_0, q_1}</td>
<td>{q_0, q_1}</td>
<td>{q_0, q_2}</td>
</tr>
<tr>
<td>{q_0, q_2}</td>
<td>{q_0, q_1, q_3}</td>
<td></td>
</tr>
</tbody>
</table>

The diagram shows a DFA with states $q_0, q_1, q_2, q_3$ and transitions labeled with $a$ and $b$. The table represents the transition function $\delta$. For example, $\delta(q_0, a) = \{q_0, q_1\}$.
A DFA with states \( q_0, q_1, q_2, q_3 \), transitions for \( a \) and \( b \), and \( \Sigma \) as the alphabet.

- \( q_0 \) is the start state.
- \( q_3 \) is a final state.

Transition table:

<table>
<thead>
<tr>
<th>State</th>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {q_0} )</td>
<td>( {q_0, q_1} )</td>
<td>( {q_0} )</td>
</tr>
<tr>
<td>( {q_0, q_1} )</td>
<td>( {q_0, q_1} )</td>
<td>( {q_0, q_2} )</td>
</tr>
<tr>
<td>( {q_0, q_2} )</td>
<td>( {q_0, q_1, q_3} )</td>
<td></td>
</tr>
</tbody>
</table>
\[
\begin{array}{c|c|c}
\text{State} & a & b \\
\hline
\{q_0\} & \{q_0, q_1\} & \{q_0\} \\
\{q_0, q_1\} & \{q_0, q_1\} & \{q_0, q_2\} \\
\{q_0, q_2\} & \{q_0, q_1, q_3\} & \\
\end{array}
\]
<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>{q_0}</td>
<td>{q_0, q_1}</td>
<td>{q_0}</td>
</tr>
<tr>
<td>{q_0, q_1}</td>
<td>{q_0, q_1}</td>
<td>{q_0, q_2}</td>
</tr>
<tr>
<td>{q_0, q_2}</td>
<td>{q_0, q_1, q_3}</td>
<td></td>
</tr>
</tbody>
</table>

The diagram shows a Finite State Machine (FSM) with states labeled with \(q_0, q_1, q_2, q_3\). The transitions are as follows:

- From \(q_0\) on input \(a\) to \(q_1\)
- From \(q_1\) on input \(b\) to \(q_2\)
- From \(q_2\) on input \(a\) to \(q_3\)
- \(q_3\) is an accepting state.
<table>
<thead>
<tr>
<th>State</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td>${q_0, q_1, q_3}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1, q_3}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[
\begin{array}{c|cc|c}
\Sigma & a & b \\
\hline 
\{q_0\} & \{q_0, q_1\} & \{q_0\} \\
\{q_0, q_1\} & \{q_0, q_1\} & \{q_0, q_2\} \\
\{q_0, q_2\} & \{q_0, q_1, q_3\} & \{q_0\} \\
\{q_0, q_1, q_3\} & & \\
\end{array}
\]

Fill in this row.

Answer at [https://cs103.stanford.edu/pollev](https://cs103.stanford.edu/pollev)
<table>
<thead>
<tr>
<th>State</th>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {q_0} )</td>
<td>( {q_0, q_1} )</td>
<td>( {q_0} )</td>
</tr>
<tr>
<td>( {q_0, q_1} )</td>
<td>( {q_0, q_1} )</td>
<td>( {q_0, q_2} )</td>
</tr>
<tr>
<td>( {q_0, q_2} )</td>
<td>( {q_0, q_1, q_3} )</td>
<td>( {q_0} )</td>
</tr>
<tr>
<td>( {q_0, q_1, q_3} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Set</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>{q_0}</td>
<td>{q_0, q_1}</td>
<td>{q_0}</td>
</tr>
<tr>
<td>{q_0, q_1}</td>
<td>{q_0, q_1}</td>
<td>{q_0, q_2}</td>
</tr>
<tr>
<td>{q_0, q_2}</td>
<td>{q_0, q_1, q_3}</td>
<td>{q_0}</td>
</tr>
<tr>
<td>{q_0, q_1, q_3}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- States: \( q_0, q_1, q_2, q_3 \)
- Transitions:
  - \( q_0 \xrightarrow{a} q_1 \)
  - \( q_1 \xrightarrow{b} q_2 \)
  - \( q_2 \xrightarrow{a} q_3 \)
  - \( q_0 \xrightarrow{\Sigma} q_0 \)

The diagram shows a transition graph with the given state set and transition rules.
The diagram represents a finite automaton with states $q_0, q_1, q_2,$ and $q_3$. The table below describes the transitions:

<table>
<thead>
<tr>
<th>State Set</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td>${q_0, q_1, q_3}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1, q_3}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A finite automaton with the following transitions:

- Start state: $q_0$
- On input $a$:
  - From $q_0$ to $q_1$
  - From $q_1$ to $q_2$
- On input $b$:
  - From $q_0$ to $q_1$
- The language is defined by the following table:

<table>
<thead>
<tr>
<th>State</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td>${q_0, q_1, q_3}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1, q_3}$</td>
<td>${q_0, q_1}$</td>
<td></td>
</tr>
</tbody>
</table>
The given automaton is described by the following transition table:

<table>
<thead>
<tr>
<th>Current State(s)</th>
<th>On 'a'</th>
<th>On 'b'</th>
</tr>
</thead>
<tbody>
<tr>
<td>{q_0}</td>
<td>{q_0, q_1}</td>
<td>{q_0}</td>
</tr>
<tr>
<td>{q_0, q_1}</td>
<td>{q_0, q_1}</td>
<td>{q_0, q_2}</td>
</tr>
<tr>
<td>{q_0, q_2}</td>
<td>{q_0, q_1, q_3}</td>
<td>{q_0}</td>
</tr>
<tr>
<td>{q_0, q_1, q_3}</td>
<td>{q_0, q_1}</td>
<td></td>
</tr>
</tbody>
</table>

The initial state is \(q_0\) and the alphabet \(\Sigma\) includes 'a' and 'b'.
<table>
<thead>
<tr>
<th>State Set</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td>${q_0, q_1, q_3}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1, q_3}$</td>
<td>${q_0, q_1}$</td>
<td></td>
</tr>
</tbody>
</table>
\[ \begin{array}{c|cc}
\{ q_0 \} & \{ q_0, q_1 \} & \{ q_0 \} \\
\{ q_0, q_1 \} & \{ q_0, q_1 \} & \{ q_0, q_2 \} \\
\{ q_0, q_2 \} & \{ q_0, q_1, q_3 \} & \{ q_0 \} \\
\{ q_0, q_1, q_3 \} & \{ q_0, q_1 \} & \end{array} \]
\[ \begin{array}{|c|c|c|} \hline
\text{State} & \text{ } & \text{ } \\
\{q_0\} & a & \{q_0, q_1\} \\
\{q_0, q_1\} & a & \{q_0\} \\
\{q_0, q_1\} & b & \{q_0, q_2\} \\
\{q_0, q_2\} & a & \{q_0\} \\
\{q_0, q_1, q_3\} & b & \{q_0\} \\
\{q_0, q_1, q_3\} & b & \{q_0, q_2\} \\
\hline
\end{array} \]
The given DFA has the following states:

- Start state: $q_0$
- Transitions:
  - $a$: $q_0 \rightarrow q_1$, $q_1 \rightarrow q_2$, $q_2 \rightarrow q_3$
  - $b$: $q_0 \rightarrow \{q_0\}$, $q_1 \rightarrow \{q_0, q_1\}$, $q_2 \rightarrow \{q_0, q_2\}$, $q_3 \rightarrow \{q_0, q_1, q_2, q_3\}$

The DFA accepts the following strings:

- $\{q_0\}$
- $\{q_0, q_1\}$
- $\{q_0, q_2\}$
- $\{q_0, q_1, q_2\}$
- $\{q_0, q_1, q_3\}$
- $\{q_0, q_2, q_3\}$
- $\{q_0, q_1, q_2, q_3\}$
The diagram illustrates a finite automaton with states $q_0, q_1, q_2, q_3$ and transitions labeled with symbols $a$ and $b$. The start state is $q_0$. The table below shows the transitions:

<table>
<thead>
<tr>
<th>State</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${q_0}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>${q_0, q_1}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
<tr>
<td>${q_0, q_2}$</td>
<td>${q_0, q_1, q_3}$</td>
<td>${q_0}$</td>
</tr>
<tr>
<td>$\ast{q_0, q_1, q_3}$</td>
<td>${q_0, q_1}$</td>
<td>${q_0, q_2}$</td>
</tr>
</tbody>
</table>
\[ \begin{align*}
\{q_0, q_1, q_3\} & \\
\{q_0, q_2\} & \\
\{q_0, q_1\} & \\
\{q_0\} & \\
\end{align*} \]
\[ \Sigma \]

Start in \( q_0 \).

1. \( q_0 \) to \( q_1 \) on \( a \).
2. \( q_1 \) to \( q_2 \) on \( b \).
3. \( q_2 \) to \( q_3 \) on \( a \).

Input string: \( a b a a a b a a \)

State transitions:
- \( q_0 \) to \( q_1 \) on \( a \)
- \( q_1 \) to \( q_2 \) on \( b \)
- \( q_2 \) to \( q_3 \) on \( a \)

Start state: \( \{ q_0 \} \)

Transitions:
- \( a \) from \( q_0 \) to \( q_1 \)
- \( b \) from \( q_1 \) to \( q_2 \)
- \( a \) from \( q_2 \) to \( q_3 \)

Final state: \( \{ q_0, q_1, q_3 \} \)
\[ \Sigma \]

- Start state: \( q_0 \)
- Transition: a -> \( q_1 \)
- Transition: b -> \( q_2 \)
- Transition: a -> \( q_3 \)

Input string: ababaaba

- Start state: \( \{ q_0 \} \)
- Transition: a -> \( \{ q_0, q_1 \} \)
- Transition: b -> \( \{ q_0, q_2 \} \)
- Transition: a -> \( \{ q_0, q_1, q_3 \} \)
The Subset Construction

- This procedure for turning an NFA for a language $L$ into a DFA for a language $L$ is called the *subset construction*.
  - It’s sometimes called the *powerset construction*; it’s different names for the same thing!

- Intuitively:
  - Each state in the DFA corresponds to a set of states from the NFA.
  - Each transition in the DFA corresponds to what transitions would be taken in the NFA when using the massive parallel intuition.
  - The accepting states in the DFA correspond to which sets of states would be considered accepting in the NFA when using the massive parallel intuition.

- There’s an online *Guide to the Subset Construction* with a more elaborate example involving $\epsilon$-transitions and cases where the NFA dies; check that for more details.
The Subset Construction

- In converting an NFA to a DFA, the DFA's states correspond to sets of NFA states.

- **Useful fact:** \( |\wp(S)| = 2^{|S|} \) for any finite set \( S \).

- In the worst-case, the construction can result in a DFA that is *exponentially larger* than the original NFA.

- **Question to ponder:** Can you find a family of languages that have NFAs of size \( n \), but no DFAs of size less than \( 2^n \)?
The Regular Languages
Regular Languages

• Let \( L \subseteq \Sigma^* \) be a language.

• We say that \( L \) is a regular language if there is a DFA \( D \) where \( \mathcal{L}(D) = L \).

• Equivalently, \( L \) is a regular language if there is an NFA \( N \) where \( \mathcal{L}(N) = L \).

• Key questions:
  • What do the regular languages “feel” like?
  • What properties do they have?
  • What languages aren’t regular?
Closure Under Union

- If $L_1$ and $L_2$ are languages over the alphabet $\Sigma$, the language $L_1 \cup L_2$ is the language of all strings in at least one of the two languages.
- If $L_1$ and $L_2$ are regular languages, is $L_1 \cup L_2$?
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1 \cup L_2$. 
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1 \cup L_2$. 
\( L_1 = \{ w \in \{a, b\}^* | \text{w has even length} \} \)

\( L_2 = \{ w \in \{a, b\}^* | \text{w has length exactly three} \} \)

Construct an NFA for \( L_1 \cup L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* | w \text{ has even length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* | w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
$L_1 = \{ w \in \{a, b\}^* | w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* | w \text{ has length exactly three} \}$

Construct an NFA for $L_1 \cup L_2$. 
$L_1 = \{ \ w \in \{a, b\}^* \mid \text{w has even length} \ \}$

$L_2 = \{ \ w \in \{a, b\}^* \mid \text{w has length exactly three} \ \}$

Construct an NFA for $L_1 \cup L_2$. 
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1 \cup L_2$. 
\[ L_1 = \{ w \in \{a, b}\}^* \mid w \text{ has even length} \} \]
\[ L_2 = \{ w \in \{a, b}\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
Construct an NFA for $L_1 \cup L_2$. 

$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1 \cup L_2$. 
$L_1 = \{ w \in \{a, b\}^* | w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* | w \text{ has length exactly three} \}$

Construct an NFA for $L_1 \cup L_2$. 
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \} \]

\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \} \]

\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
Construct an NFA for $L_1 \cup L_2$. 

$L_1 = \{ w \in \{a, b\}^* | w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* | w \text{ has length exactly three} \}$
Construct an NFA for $L_1 \cup L_2$.

$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
Construct an NFA for $L_1 \cup L_2$.

$L_1 = \{ w \in \{a, b\}^* | w \text{ has even length} \}$
$L_2 = \{ w \in \{a, b\}^* | w \text{ has length exactly three} \}$
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 \cup L_2 \).
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1 \cup L_2$. 
Construct an NFA for $L_1 \cup L_2$. 

$L_1 = \{ w \in \{a, b\}^* | w \text{ has even length} \}$

$L_2 = \{ w \in \{a, b\}^* | w \text{ has length exactly three} \}$
Closure Under Intersection

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
Closure Under Intersection

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
Closure Under Intersection

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?
Closure Under Intersection

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?

\[
\overline{L_1} \cup \overline{L_2}
\]
Closure Under Intersection

- If $L_1$ and $L_2$ are languages over $\Sigma$, then $L_1 \cap L_2$ is the language of strings in both $L_1$ and $L_2$.

- Question: If $L_1$ and $L_2$ are regular, is $L_1 \cap L_2$ regular as well?

Hey, it's De Morgan's laws!

Hey, it's De Morgan's laws!
Concatenation
String Concatenation

- If \( w \in \Sigma^* \) and \( x \in \Sigma^* \), the *concatenation* of \( w \) and \( x \), denoted \( wx \), is the string formed by tacking all the characters of \( x \) onto the end of \( w \).

- Example: if \( w = \text{quo} \) and \( x = \text{kka} \), the concatenation \( wx = \text{quokka} \).

- This is analogous to the + operator for strings in many programming languages.

- Some facts about concatenation:
  - The empty string \( \varepsilon \) is the *identity element* for concatenation:
    \[
    w\varepsilon = \varepsilon w = w
    \]
  - Concatenation is *associative*:
    \[
    wxy = w(xy) = (wx)y
    \]
Concatenation

- The **concatenation** of two languages $L_1$ and $L_2$ over the alphabet $\Sigma$ is the language
  \[ L_1L_2 = \{ x \mid \exists w_1 \in L_1. \exists w_2 \in L_2. x = w_1w_2 \} \]

- Let $L_1 = \{ ab, ba \}$ and $L_2 = \{ aa, bb \}$. What is $L_1L_2$?

Answer at https://cs103.stanford.edu/pollev
Concatenation Example

- Let $\Sigma = \{ a, b, ..., z, A, B, ..., Z \}$ and consider these languages over $\Sigma$:
  - $Noun = \{ \text{Puppy, Rainbow, Whale, ... } \}$
  - $Verb = \{ \text{Hugs, Juggles, Loves, ... } \}$
  - $The = \{ \text{The} \}$
  - The language $TheNounVerbTheNoun$ is
    - $\{ \text{ThePuppyHugsTheWhale, TheWhaleLovesTheRainbow, TheRainbowJugglesTheRainbow, ... } \}$
Concatenation

- The *concatenation* of two languages $L_1$ and $L_2$ over the alphabet $\Sigma$ is the language

  $$L_1L_2 = \{ \, x \mid \exists w_1 \in L_1. \exists w_2 \in L_2. \, x = w_1w_2 \, \}$$

- Two views of $L_1L_2$:  
  - The set of all strings that can be made by concatenating a string in $L_1$ with a string in $L_2$.
  - The set of strings that can be split into two pieces: a piece from $L_1$ and a piece from $L_2$. 

\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1L_2 \).
\[ L_1 = \{ \ w \in \{a, b\}^* \ | \ w \text{ has odd length} \ \} \]

\[ L_2 = \{ \ w \in \{a, b\}^* \ | \ w \text{ has length exactly three} \ \} \]

Construct an NFA for \( L_1L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1L_2 \).
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1L_2$. 

$L_1 = \{ \ w \in \{a, b\}^* \mid w \text{ has odd length} \ \}$  
$L_2 = \{ \ w \in \{a, b\}^* \mid w \text{ has length exactly three} \ \}$

Construct an NFA for $L_1L_2$. 
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1L_2$. 
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1L_2$. 

DFA for $L_1$

NFA for $L_2$
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 L_2 \).
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1L_2$. 
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1L_2$. 
Construct an NFA for \( L_1 \) \( L_2 \).

\[
L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}
\]

\[
L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}
\]
Let
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \} \]
and
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 L_2 \).
\( L_1 = \{ \ w \in \{a, b\}^* \mid w \text{ has odd length} \ \} \)
\( L_2 = \{ \ w \in \{a, b\}^* \mid w \text{ has length exactly three} \ \} \)

Construct an NFA for \( L_1L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1L_2 \).
$L_1 = \{ \ w \in \{a, b\}^* \mid w \text{ has odd length} \ \}$

$L_2 = \{ \ w \in \{a, b\}^* \mid w \text{ has length exactly three} \ \}$

Construct an NFA for $L_1L_2$. 
$L_1 = \{ \ w \in \{a, b\}^* \mid w \text{ has odd length} \ \}$

$L_2 = \{ \ w \in \{a, b\}^* \mid w \text{ has length exactly three} \ \}$

Construct an NFA for $L_1L_2$. 
$L_1 = \{ w \in \{a, b\}* | w \text{ has odd length} \}$

$L_2 = \{ w \in \{a, b\}* | w \text{ has length exactly three} \}$

Construct an NFA for $L_1L_2$. 
Consider the languages $L_1$ and $L_2$ defined as follows:

$L_1 = \{ \ w \in \{a, b\}^* \ | \ w \text{ has odd length} \ \}$

$L_2 = \{ \ w \in \{a, b\}^* \ | \ w \text{ has length exactly three} \ \}$

Construct an NFA for $L_1L_2$. 
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \} \]
\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1L_2 \).
\[ L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \} \]

\[ L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \} \]

Construct an NFA for \( L_1 L_2 \).
$L_1 = \{ \ w \in \{a, b\}^* \ | \ w \text{ has odd length} \ \}$

$L_2 = \{ \ w \in \{a, b\}^* \ | \ w \text{ has length exactly three} \ \}$

Construct an NFA for $L_1L_2$. 
Construct an NFA for $L_1 L_2$.

$L_1 = \{ \ w \in \{a, b\}^* \mid w \text{ has odd length} \ \}$

$L_2 = \{ \ w \in \{a, b\}^* \mid w \text{ has length exactly three} \ \}$

DFA for $L_1$

NFA for $L_2$

$a \ b \ a \ b \ a \ b \ b \ b \ b \ b$
Let $L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$

Let $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for $L_1L_2$. 
Construct an NFA for $L_1L_2$.

$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$

$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$
The Kleene Star
Lots and Lots of Concatenation

- Consider the language $L = \{ \text{aa, b} \}$
- $LL$ is the set of strings formed by concatenating pairs of strings in $L$.
  \[ \{ \text{aaaa, aab, baa, bb} \} \]
- $LLL$ is the set of strings formed by concatenating triples of strings in $L$.
  \[ \{ \text{aaaaaa, aaaaab, aabaa, aabb, baaaaa, baab, bbaa, bbb} \} \]
- $LLLLL$ is the set of strings formed by concatenating quadruples of strings in $L$.
  \[ \{ \text{aaaaaaaaa, aaaaaaab, aaaaaabaa, aabaaaab, aabaaa, aabaab, aabbaa, aabbb, baaaaaaa, baaaaab, baabaa, baabb, bbbaaa, bbaab, bbb} \} \]
Language Exponentiation

- We can define what it means to “exponentiate” a language as follows:
  - $L^0 = \{ \varepsilon \}$
    - Intuition: The only string you can form by gluing no strings together is the empty string.
    - Notice that $\{ \varepsilon \} \neq \emptyset$. Can you explain why?
  - $L^{n+1} = LL^n$
    - Idea: Concatenating $(n+1)$ strings together works by concatenating $n$ strings, then concatenating one more.
- **Question to ponder:** Why define $L^0 = \{ \varepsilon \}$?
- **Question to ponder:** What is $\emptyset^0$?
The Kleene Star
The Kleene Closure

• An important operation on languages is the **Kleene closure**, or **Kleene star**, which is defined as

\[ L^* = \{ w \in \Sigma^* \mid \exists n \in \mathbb{N}. w \in L^n \} \]

• Mathematically:

\[ w \in L^* \iff \exists n \in \mathbb{N}. w \in L^n \]

• Intuitively, \( L^* \) is the language all possible ways of concatenating zero or more strings in \( L \) together, possibly with repetition.

• **Question to ponder:** What is \( \emptyset^* \)?
The Kleene Closure

If \( L = \{ \text{a, bb} \} \), then \( L^* = \{ \)

\( \varepsilon, \)

\( \text{a, bb,} \)

\( \text{aa, abb, bba, bbbb,} \)

\( \text{aaa, aabb, abba, abbbb, bbbaa, bbabb, bbbba, bbbbbbb,} \)

\( \ldots \)

\( \} \)

Think of \( L^* \) as the set of strings you can make if you have a collection of stamps – one for each string in \( L \) – and you form every possible string that can be made from those stamps.
Idea: Can we convert an NFA for a language $L$ to an NFA for language $L^*$?
$L = \{ w \in \{ a, b \}^* | w \text{ has an odd number of } a \text{'s and an even number of } b \text{'s } \}$

Construct an NFA for $L^*$. 
Construct an NFA for $L^*$. 

$L = \{ w \in \{a, b\}^* | w \text{ has an odd number of a's and an even number of b's } \}$

Construct an NFA for $L^*$. 

DFA for $L$
\[ L = \{ w \in \{a, b\}^* | w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \} \]

Construct an NFA for \( L^* \).
\[ L = \{ w \in \{a, b\}^* | \text{w has an odd number of a's and an even number of b's} \} \]

Construct an NFA for \( L^* \).
$L = \{ \ w \in \{a, b\}^* \ | \ w \text{ has an odd number of a's and an even number of b's } \}$

Construct an NFA for $L^*$. 
\[ L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a \text{'s and an even number of } b \text{'s } \} \]

Construct an NFA for \( L^* \).
$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}$

Construct an NFA for $L^*$. 
$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}$

Construct an NFA for $L^*$.
Let $L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}$

Construct an NFA for $L^*$. 

DFA for $L$
Construct an NFA for $L^*$. 

$L = \{ w \in \{a, b\}^* \mid w$ has an odd number of $a$’s and an even number of $b$’s $\}$ 

Construct an NFA for $L^*$. 

DFA for $L$
$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a \text{'s and an even number of } b \text{'s} \}$

Construct an NFA for $L^*$.
\[ L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \} \]

Construct an NFA for \( L^* \).
$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of a's and an even number of b's} \}$

Construct an NFA for $L^*$. 
Construct an NFA for $L^*$.

$L = \{ \ w \in \{a, b\}^* \mid w \text{ has an odd number of } a's \text{ and an even number of } b's \ \}$

Construct an NFA for $L^*$.
$L = \{ w \in \{a, b\}^* | w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}$

Construct an NFA for $L^*$. 
$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for $L^*$. 
Let \( L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a \text{'s and an even number of } b \text{'s} \} \).

Construct an NFA for \( L^* \).
$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}$

Construct an NFA for $L^*$. 
Let $L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}$

Construct an NFA for $L^*$. 
Construct an NFA for \( L^* \).
$L = \{ w \in \{a, b\}^* | w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}$

Construct an NFA for $L^*$. 
\[ L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a's \text{ and an even number of } b's \} \]

Construct an NFA for \( L^* \).
\[ L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}\]

Construct an NFA for \( L^* \).
$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}$

Construct an NFA for $L^*$. 

DFA for $L$
$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s } \}$

Construct an NFA for $L^*$. 

Question: Why add the new state out front? Why not just make the old start state accepting?
Closure Properties

- **Theorem:** If $L_1$ and $L_2$ are regular languages over an alphabet $\Sigma$, then so are the following languages:
  - $L_1 \cup L_2$
  - $L_1 \cap L_2$
  - $L_1L_2$
  - $L_1^*$

- These properties are called **closure properties of the regular languages**.
Next Time

• **Regular Expressions**
  • Building languages from the ground up!

• **Thompson’s Algorithm**
  • A UNIX Programmer in Theoryland.

• **Kleene’s Theorem**
  • From machines to programs!