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 \( \epsilon \).

• 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
  - *Deterministic*
  - *Finite*
  - *Automaton*
- DFAs are the simplest type of automaton that we will see in this course.
DFAs

• A DFA is defined relative to some alphabet Σ.

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

• There is a unique start state.

• There are zero or more accepting states.
A Sample DFA

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

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

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

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

\( q_0 \)

\( q_1 \)

\( q_2 \)

\( q_3 \)

\( \Sigma \)

\( 0 \)

\( 1 \)

\( 0 \)

\( 1 \)

\( 0 \)

\( 0 \)

\( 1 \)

\( * q_0 \)

\( q_0 \)

\( q_1 \)

\( q_2 \)

\( q_3 \)

\( * q_3 \)

\( q_3 \)

\( q_3 \)

\( q_3 \)

\( q_3 \)
Tabular DFAs

These stars indicate accepting states.

<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

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

Question to ponder: Why isn’t there a column here for $\Sigma$?
My Turn to Code Things Up!

```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 $\overline{L}$) is the language of all strings in $\Sigma^*$ that aren't in $L$.
- Formally:
  \[ \overline{L} = \Sigma^* - L \]

Good proofwriting exercise: prove $\overline{\overline{L}} = L$ 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

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

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

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

\[ q_3 \xleftarrow{0} q_1 \xleftarrow{0} q_2 \]

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

\[ q_0 \xleftarrow{0, 1} \]

\[ q_0 \xrightarrow{\text{start}} \]

Diagram showing states and transitions.
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 zero or multiple decisions that it can make at one point.
  - 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

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

$start\rightarrow q_0$

$q_0 \xrightarrow{0, 1} q_3$

$q_2 \xrightarrow{0} q_3 \xrightarrow{0, 1}$

$q_3 \xrightarrow{0, 1}$
A Simple NFA

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

0 1 0 1 1
A Simple NFA
A Simple NFA

start

$q_0$ 1 $q_1$

0, 1

$q_1$ 1 $q_2$

q3

0

0, 1

0, 1

0 1 0 1 1
A Simple NFA

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

\[
\begin{array}{cccc}
\rightarrow & 1 & \rightarrow & 1 \\
0, 1 & \rightarrow & 0 & \rightarrow 0, 1 \\
0, 1 & \rightarrow & \end{array}
\]

\[
\begin{array}{c}
0 \\
1 \\
0 \\
1 \\
1
\end{array}
\]
A Simple NFA

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

The diagram shows a nondeterministic finite automaton (NFA) with states labeled as follows:

- $q_0$ (start state)
- $q_1$
- $q_2$
- $q_3$

The transitions are labeled with symbols:

- $0$, $1$

The sequence $010111$ is shown at the bottom, indicating the path the automaton follows.
A Simple NFA

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

Input: 01011
A Simple NFA

0 1 0 1 1
A Simple NFA

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

- **Start状态**: $q_0$
- **状态转接**:
  - $q_0$ 转 $q_1$: $1$
  - $q_1$ 转 $q_2$: $1$
  - $q_2$ 转 $q_3$: $0, 1$
  - $q_3$ 转 $q_2$: $0, 1$

**输入串**：0 1 0 1 1
A Simple NFA

start

$q_0$ 1 $q_1$

$q_1$ 1 $q_2$

$q_3$

$q_2$

0, 1

0, 1

$q_3$

$q_0$

$q_1$

$q_2$

$q_3$

0 1 0 1 1
A Simple NFA
A Simple NFA

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

Input: 0 1 0 1 1
A Simple NFA

start

\[ q_0 \] \rightarrow \begin{array}{c}
q_1 \\
1
\end{array}

\[ q_3 \]

\begin{array}{c}
0, 1 \\
0
\end{array}

\[ q_2 \] \rightarrow 
\begin{array}{c}
0, 1 \\
0, 1
\end{array}

0 1 0 1 1
A Simple NFA
A Simple NFA

Start

$q_0$ 1 $q_1$ 1 $q_2$

$q_3$

0, 1

0, 1

0, 1

0 1 0 1 1
A Simple NFA

![Diagram of a simple NFA]

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

Input string: 010111
A Simple NFA
A Simple NFA
A Simple NFA

![Diagram of a simple NFA with states q0, q1, q2, and q3, transitions labeled 0, 1, and 0, 1, and a string 01011 being processed.]
A Simple NFA
A Simple NFA

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

$q_3 \xrightarrow{0} q_0, 1$

Input: 0 1 0 1 1 1
A Simple NFA

start

$q_0$  1  $q_1$

$q_2$

$q_3$

0, 1

0

0, 1

0, 1

0

1

0, 1

0, 1

0, 1

0

1

0

1

1
A Simple NFA

\[
\begin{array}{cccc}
q_0 & 1 & q_1 & 1 \\
\circ & \circ & \circ & \circ \\
0, 1 & \circ & \circ & \circ \\
\end{array}
\]
A Simple NFA

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

\text{start} \quad 1, 1 \quad 0, 1 \quad 0, 1 \quad 0, 1

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

SEAL
OF APPROVAL

0 1 0 1 1
A More Complex NFA

\[
\begin{align*}
\text{start} & \rightarrow q_0 & 1 & \\
q_0 & \rightarrow q_1 & 1 & \\
q_1 & \rightarrow q_2 & 1 & \\
q_2 & \rightarrow q_0 & 0, 1 & 
\end{align*}
\]
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
A More Complex NFA
A More Complex NFA

start → $q_0$ (0, 1) → $q_1$ (1) → $q_2$ (1)

0 1 0 1 1
A More Complex NFA

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

\[ q_0 \xrightarrow{0,1} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \]
A More Complex NFA
A More Complex NFA

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

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

Start

The NFA starts at state \( q_0 \) and can follow the transitions labeled with 1 to states \( q_1 \) and \( q_2 \). The state \( q_2 \) is a dead-end state, as indicated by the double circle.
A More Complex NFA

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

\[ q_0 \xrightarrow{0, 1} \]

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

start $q_0$ -> $q_1$ -> $q_2$

- From $q_0$ to $q_1$: 1
- From $q_1$ to $q_2$: 1
- From $q_0$: 0, 1

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

![NFA Diagram]

- Start state: $q_0$
- Transitions:
  - $q_0 \xrightarrow{1} q_1$
  - $q_0 \xleftarrow{0, 1} q_1$
  - $q_1 \xrightarrow{1} q_2$
- Accepting state: $q_2$

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

Start

$q_0$

$q_1$

$q_2$

Transition:
- 1 from $q_0$ to $q_1$
- 1 from $q_1$ to $q_2$
- 0, 1 from $q_0$ to itself
A More Complex NFA
A More Complex NFA

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

**SEAL OF APPROVAL**
Hello, NFA!

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

$h \mid i$
Hello, NFA!

start \rightarrow \begin{array}{c}
\text{h} \\
q_0 \\
\text{h} \\
q_1 \\
\text{i} \\
q_2
\end{array}

\begin{array}{c}
\text{h} \\
\text{i}
\end{array}
Hello, NFA!

The diagram represents a non-deterministic finite automaton (NFA) with the following states and transitions:

- **Start state**: $q_0$
- **Transitions**:
  - From $q_0$ to $q_1$ on input 'h'
  - From $q_1$ to $q_2$ on input 'i'

The diagram also shows the input symbols 'h' and 'i' in the state transitions.
Hello, NFA!

- Start state: $q_0$ with transitions:
  - $h$ to $q_1$
  - $i$ to $q_2$

- Transition matrix:
  - $h$: $[h, i]$
Hello, NFA!

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

{h, i}
Hello, NFA!

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

 SEAL

 OF APPROVAL

\[ h \ | \ i \]
Tragedy in Paradise

\[
\begin{align*}
\text{start} & \quad q_0 \quad h \quad q_1 \quad i \quad q_2
\end{align*}
\]

\[
\begin{array}{c}
\text{h} \\
\text{i} \\
\text{p}
\end{array}
\]
Tragedy in Paradise

\[
\begin{align*}
\text{start} &\quad q_0 & h & q_1 & i & q_2
\end{align*}
\]
Tragedy in Paradise

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

start

$q_0$ → $q_1$ → Sad Face

$h$ $i$ $p$
Tragedy in Paradise

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

h i p
The language of an NFA is $\mathcal{L}(N) = \{ w \in \Sigma^* | N \text{ accepts } w \}$. 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?)
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 **\(\varepsilon\)-transition**.
- An NFA may follow any number of \(\varepsilon\)-transitions at any time without consuming any input.

![NFA Diagram](image)
**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 \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.
\(\varepsilon\)-Transitions

- NFAs have a special type of transition called the \textit{\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 \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.

![Diagram of ε-Transitions](image-url)
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.

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.
Time-Out For Announcements!
Midterms Graded

• Midterms have been graded! If you didn’t pick yours up yet, you can grab it from the Gates building.
  • SCPD students – exams have been sent back to the SCPD distribution office. If you haven’t received yours yet, ping the SCPD distribution office.
• We’ve posted a regrade request form on the course website with instructions about how to ask for a regrade. Regrade requests are due next next Wednesday.
Your Questions
“Why do the majority of students here neglect to wear their helmets? I always assumed Stanford people would want to protect their heads”
"In your opinion, as a 20 year old, what is the best reason to be optimistic for the future?"

It’s almost fashionable these days to be fatalistic about the future, and it’s a position I’ve never understood. People are so smart, so clever, and so resourceful that I’m genuinely excited to see what the future has in store for us.

We’ve got a lot of challenges to face, but we have historical precedents we can look to to see how to best resolve them. The particular challenges will change, but we’ll find ways to figure them out. And despite what a lot of folks say, I don’t think we’re facing any crises that are so totally unlike our past ones that we can’t possibly address them. We’ve found lots of creative ways to solve major problems in the past. I’m sure we can do it again.
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

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

\[ \Sigma \]

start
Perfect Positive Guessing

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

Start: \( q_0 \)

Input alphabet: \( \Sigma \)

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

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

Input sequence: \( a \ b \ a \ b \ a \ a \)
Perfect Positive Guessing

- Start at $q_0$
- Move to $q_1$ on $a$
- Move to $q_2$ on $b$
- Move to $q_3$ on $a$

The symbols $a$, $b$, and $\Sigma$ are transitions in the diagram.
Perfect Positive Guessing

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

\[
\Sigma \xrightarrow{\text{start}} \]

Input sequence:

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

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

States:
- $q_0$
- $q_1$
- $q_2$
- $q_3$

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

Input alphabet: $\Sigma$

Strings:
- $a b a b a b a$

Start state: $q_0$

Final state: $q_3$
Perfect Positive Guessing

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

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

\[ \Sigma \]

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

Start
Perfect Positive Guessing

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

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

Input:\[ \Sigma \]

Sample Input:\[ a \ b \ b \ a \ b \ b \ a \]

Diagram:

- Start state: \( q_0 \)
- Transitions:
  - \( q_0 \rightarrow q_1 \) on input \( a \)
  - \( q_1 \rightarrow q_2 \) on input \( b \)
  - \( q_2 \rightarrow q_3 \) on input \( a \)
Perfect Positive Guessing

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

\[ \Sigma \]

a b a b a 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

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

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

Start state: \(q_0\)

Final state: \(q_3\)
Massive Parallelism

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

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

Start: \( q_0 \)

Transitions:
- \( a \rightarrow q_1 \)
- \( b \rightarrow q_2 \)
- \( a \rightarrow q_3 \)

Input alphabet: \( \sum \)

Example input sequence: \( a b a b a b a \)
Massive Parallelism

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

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

\[ \sum \]

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

a b a b a a
Massive Parallelism

$\sum$

start

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

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

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

\[ \Sigma \]

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

Input sequence:

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

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

\[ \Sigma \]

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

\[ \Sigma \]

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

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

\[ \Sigma \]

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

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

\[ \Sigma \]

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

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

\[ \Sigma \]

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

Input: \[ ababaab \]
Massive Parallelism
Massive Parallelism
Massive Parallelism

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

\[ \Sigma \]

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

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

The diagram shows a finite state machine with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are labeled with symbols $a$ and $b$, and the input alphabet is represented by $\Sigma$. The machine starts in state $q_0$ and transitions through $q_1$ and $q_2$ with input symbols $a$ and $b$, respectively. The transition from $q_2$ to $q_3$ is indicated with a dotted arrow, suggesting a parallel or concurrent transition. The arrow below the states indicates the sequence of symbols $a b a b a b a$.
Massive Parallelism

start

$q_0$ \rightarrow $q_1$ \rightarrow $q_2$ \rightarrow $q_3$

$a$ \rightarrow $b$ \rightarrow $a$

$a$ \rightarrow $b$ \rightarrow $a$

$a$ \rightarrow $b$ \rightarrow $a$

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

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

\[
\Sigma \xrightarrow{a} q_0
\]

Input: \(aababa\)
Massive Parallelism

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

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

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

\[ \Sigma \]

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

\[ \Sigma \]

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

\[ \text{Start} \]

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

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

Input: \( \Sigma \)

Transitions:
- \( q_0 \) to \( q_1 \) on input \( a \)
- \( q_1 \) to \( q_2 \) on input \( b \)
- \( q_2 \) to \( q_3 \) on input \( a \)
- \( q_0 \) has a self-loop on input \( \Sigma \)
Massive Parallelism

\[ \sum \]

\[ a \]

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

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

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

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

Input symbols: \[ \Sigma = \{a, b\} \]
Massive Parallelism
Massive Parallelism

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

- \( \Sigma \)
Massive Parallelism

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

\[ \sum \]

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

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

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

\[ \Sigma \]

a b a b b

\[ \text{start} \]
Massive Parallelism

\[ \Sigma \]

start \]

\[ q_0 \]

\[ q_1 \]

\[ q_2 \]

\[ q_3 \]

**Input:**

- a
- b

**Transition:**

- a from q_0 to q_1
- b from q_1 to q_2
- a from q_2 to q_3

**Acceptance:**

- a from q_3
Massive Parallelism

\[ \Sigma \]

Start: \( q_0 \)  
\[ q_1 \]  \( a \)  \[ q_2 \]  \( b \)  \[ a \]  \( q_3 \)  

Input sequence: \( ababab \)
Massive Parallelism

\[ \sum \]

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

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

\[ \]
Massive Parallelism

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

\[\Sigma\]
Massive Parallelism
Massive Parallelism
Massive Parallelism

$a$, $b$, $\sum$

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

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

\[
\begin{align*}
\Sigma & \quad \rightarrow \\
q_0 & \rightarrow q_1 & \rightarrow q_2 & \rightarrow q_3
\end{align*}
\]

Start state: \( q_0 \)

Transitions:
- \( q_0 \rightarrow q_1 \) on input \( a \)
- \( q_1 \rightarrow q_2 \) on input \( b \)
- \( q_2 \rightarrow q_3 \) on input \( a \)
- \( q_3 \rightarrow q_0 \) on any input

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

\[ \Sigma \]

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

| a | b | a | b | a | b | b |
Massive Parallelism

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

\[\Sigma\]

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

\[
\begin{array}{cccc}
\text{start} & q_0 & q_1 & q_2 & q_3 \\
\Sigma & a & b & a & \text{accept}
\end{array}
\]

Input sequence: \( ababab \)
Massive Parallelism

\[
\begin{array}{cccc}
q_0 & a & q_1 & b \\
\Sigma & b & q_2 & a \\
\end{array}
\]
Massive Parallelism

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

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

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

\[\Sigma\]

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

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

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

\[ \sum \]

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

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

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.
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

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

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

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

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

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

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

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

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

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

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

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

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

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

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