Aside: Discussion on board games and **states**

- Where can pieces exist on a board?
- How do pieces move? (deterministic)
- When is the game over?

When we consider a “map” of all of the states and where they go, we create a **state diagram**
A finite automaton is a collection of three things:

1. A finite set of states, one of which is designated as the initial state, called the **start state**, and some (maybe none) of which are designated as **final states**.

2. An **alphabet** $\Sigma$ of possible input letters

3. A finite set of **transitions** that tell for each state and for each letter of the input alphabet which state to go to next.
A Brief Example

1. Three states: \( x, y, z \). Of which \( x \) is the starting state and \( z \) is the only final state.
2. \( \Sigma = \{ a, b \} \)
3. Transition Rules:
   1. From state \( x \) and input \( a \), go to state \( y \).
   2. From state \( x \) and input \( b \), go to state \( z \).
   3. From state \( y \) and input \( a \), go to state \( x \).
   4. From state \( y \) and input \( b \), go to state \( z \).
   5. From state \( z \) and input any, go to state \( z \).

This defines a language recognizer. What language does it accept?
Definition of FA as Transition Table

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>z</td>
</tr>
<tr>
<td>y</td>
<td>x</td>
<td>z</td>
</tr>
<tr>
<td>Final</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>z</td>
<td>z</td>
</tr>
</tbody>
</table>

- *states* are listed along the left
- *alphabet characters* are listed along the top
- The “cell” at the intersection of a *state* and *character* indicate which *state* should be transitioned to
A finite set of states $Q = \{q_0, q_1, q_2, \ldots\}$ of which $q_0$ is the start.

A subset of $Q$ called the final states.

An alphabet $\Sigma = \{x_1, x_2, x_3, \ldots\}$.

A transition function mapping each state-letter pair with a state:

$$\delta(q_i, x_j) = x_k$$

**NOTE:** Every state has as many **outgoing edges** as there are letters in the alphabet. It is possible for a state to have no **incoming edges** or to have many.
Transition Diagram
Another Example

Question
What language does this FA accept?

Simplification
Another Example

Question
What language does this FA accept?

Simplification
Examples

Finite Automaton Accepting Everything  $\Sigma = \{a, b\}$
Examples

Finite Automaton Accepting Everything $\Sigma = \{a, b\}$

Finite Automaton Accepting Nothing $\Sigma = \{a, b\}$
Examples

Finite Automaton Accepting Everything $\Sigma = \{a, b\}$

Finite Automaton Accepting Nothing $\Sigma = \{a, b\}$
Accepting an even-length string

Suppose we wanted to define an FA which accepts any string of an even length

- How would we do this *programmatically*?
- How can we represent this with *states*?
Suppose we wanted to define an FA which accepts any string of an even length

- How would we do this *programmatically*?
- How can we represent this with *states*?
Accepting an even-length string

Suppose we wanted to define an FA which accepts any string of an even length

• How would we do this **programmatically**?
• How can we represent this with **states**?

![Diagram of an FA accepting an even-length string]

- States: q₀, q₁
- Transitions: a, b
\(a(a + b)^*\)

\[
\begin{array}{c}
\text{a:} \\
\begin{array}{c}
q_0 \\
q_1
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\text{(a + b)*:} \\
\begin{array}{c}
q_1
\end{array}
\end{array}
\]

What if we encounter a \(b\) in \(q_0\)?
\[ a(a + b)^* \]

**Question**

What if we encounter a \( b \) in \( q_0 \)?
An extension on $a(a + b)^*$
An extension on $a(a + b)^*$

All of these are equivalent!
Matching strings with triple letters (aaa or bbb)

- Sequence of a’s
Matching strings with triple letters (aaa or bbb)

• Sequence of a’s or b’s
Matching strings with triple letters (*aaa* or *bbb*)

- Proper state transitions when sequence broken
Chalkboard Examples

Construct FAs which accept the following:

- only the exact string $\text{baa}$
- all words not ending in $b$
- all words with an odd number of $a$’s
- all words with different first and last letters
- all words with length divisible by 3
Revisiting EVEN-EVEN

Three cases:

1. $aa$
2. $bb$
3. $(ab + ba)(aa + bb)^*(ab + ba)$
Revisiting EVEN-EVEN

Three cases:

1. \textit{aa handled here}
2. \textit{bb}
3. \((ab + ba)(aa + bb)^*(ab + ba)\)
Revisiting EVEN-EVEN

Three cases:

1. **aa**
2. **bb** handled here
3. \((ab + ba)(aa + bb)^*(ab + ba)\) handled here (q3 represents \(ab + ba\))
Revisiting EVEN-EVEN

Three cases:

1. aa
2. bb
3. \((ab + ba)(aa + bb)^*(ab + ba)\)

handled here (q3 represents ab + ba)
Homework 2b

1. Build an FA that accepts only the language of all words with b as the second letter. Show both the picture and the transition table for this machine and find a regular expression for the language.

2. Find two FA’s that satisfy these conditions: Between them they accept all words in \((a + b)^*\), but there is no word accepted by both machines.

3. Describe the languages accepted by the following FA’s:

   (continued on next page)
Homework 2b

3 Describe the languages accepted by the following FA’s:

ii

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

iii

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