Acceptance by DFA
Defining DFAs

• For each description let’s draw a DFA that recognizes the language it describes

• \{ aa, ab, ac \}

• \{ \Lambda, a, abb, abbbbb, \ldots, ab^{2n}, \ldots \} 

• \{ a^m b c^n \mid m, n \in N \}
• Recall a DFA is a quintuple $A = (Q, \Sigma, s, F, \delta)$, where

- $Q$ is a set of states
- $\Sigma$ is the alphabet of input symbols
- $s$ is an element of $Q$ --- the initial state
- $F$ is a subset of $Q$ --- the set of final states
- $\delta: Q \times \Sigma \rightarrow Q$ is the transition function
Example

- In the example below, \( Q = \{ q_0, q_1, q_2 \} \), 
  \( \Sigma = \{ 0, 1 \} \), 
  \( s = q_0 \), 
  \( F = \{ q_2 \} \),

- \( \Delta \) is given by 6 equalities 

\[
\begin{align*}
  T(q_0, 0) &= q_1, \\
  T(q_0, 1) &= q_0, \\
  T(q_0, 1) &= q_0, \\
  T(q_2, 1) &= q_2 \\
  \ldots
\end{align*}
\]
Transition Table
(Hein 11.2.6)

• All the information presenting a TFA can be given by a single thing -- its *transition table*:

<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>$Q_2$</td>
<td>$Q_1$</td>
</tr>
<tr>
<td>$*Q_2$</td>
<td>$Q_2$</td>
<td>$Q_2$</td>
</tr>
</tbody>
</table>

• The initial and final states are denoted by $\rightarrow$ and $*$ respectively.
Extension of $T$ to Strings

• Given a state $q$ an $T$ a string $w$, there is a unique path labeled $w$ that starts at $q$ (why?). The endpoint of that path is denoted $T(q, w)$

• Formally, the function $T : Q \times \Sigma^* \rightarrow Q$
• is defined recursively:
  
  \[-T(q, \varepsilon) = q\]
  \[-T(q, ua) = T(T(q, u), a)\]

• Note that $T(q, a) = T(q, a)$ for every $a \in \Sigma$;
• so $T$ Toes extend $T$. 
Example trace

- Diagrams (when available) make it very easy to compute $T(q, w)$ --- just trace the path labeled $w$ starting at $q$.

- E.g. trace 101 on the Diagram below starting at $q_0$.
Implementation and precise arguments need the formal definition.

\[ T(q_0,101) = T(T(T(q_0,10),1),1) \]
\[ = T(T(T(q_0,1),0),1) \]
\[ = T(T(T(q_0,1),0),1) \]
\[ = T(T(q_0,0),1) \]
\[ = T(q_1,1) \]
\[ = q_1 \]
A DFA \( (Q, \Sigma, s, F, \Gamma) \), accepts a string \( w \) iff \( \Gamma(s, w) \in F \)

The language of the automaton \( A \) is
\[
L(A) = \{ w \mid A \text{ accepts } w \}.
\]
More formally
\[
L(A) = \{ w \mid \Gamma(\text{Start}(A), w) \in \text{Final}(A) \}
\]

**Example:**

Find a DFA whose language is the set of all strings over \( \{a, b, c\} \) that contain \( aaa \) as a substring.
DFA’s as Haskell Programs

```haskell
data DFA q s = DFA { states :: [q],
                    symbols :: [s],
                    delta :: q -> s -> q,
                    start :: q,
                    final :: [q] }
```

Haskell is a functional language that makes it easy to describe formal (or mathematical) objects.
Transition function

\[
\text{trans} :: (q \rightarrow s \rightarrow q) \rightarrow q \rightarrow [s] \rightarrow q
\]

\[
\text{trans} \ T \ q \ [] = q
\]

\[
\text{trans} \ T \ q \ (s:ss) = \text{trans} \ T \ (T \ q \ s) \ ss
\]

\[
\text{accept} :: (Eq q) \Rightarrow \text{TFA} \ q \ s \rightarrow [s] \rightarrow \text{Bool}
\]

\[
\text{accept} \ m@(\text{TFA}\{\varDelta = T, \text{start} = q0, \text{final} = f\}) \ w
= \text{elem} \ (\text{trans} \ T \ q0 \ w) \ f
\]
An Example

ma = DFA {  states = [0,1,2],
            symbols = [0,1],
            delta = \p a ->
                        (2*p+a) `mod` 3,
            start = 0,
            final = [2]
}
Another definition of acceptance

A DFA \( A = (Q, \Sigma, s, F, \tau) \), accepts a string \( x_1x_2..x_n \) (an element of \( \Sigma^* \)) iff
- There exists a sequence of states \( q_1q_2..q_nq_{n+1} \) such that
  1. \( q_1 = s \)
  2. \( q_{i+1} = \tau(q_i, x_i) \)
  3. \( q_{n+1} \) is an element of \( F \)

How does this relate to our previous definition?

\( L(A) = \{ w \mid I(s, w) \in F \} \)