Minimizing DFAs
Does this DFA have a minimal number of states?
Is this minimal?

How can we tell in general?
Theorem:

For every regular language $L$, there is a unique (up to re-labeling of the states) minimal-state DFA $M^*$ such that $L = L(M^*)$.

Furthermore, there is an efficient algorithm which, given any DFA $M$, will output this unique $M^*$.

If these were true for more general models of computation, that would be an engineering breakthrough.
Note: There isn’t a uniquely minimal NFA
Extending transition function $\delta$ to strings

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$, we extend $\delta$ to a function $\Delta : Q \times \Sigma^* \rightarrow Q$ as follows:

$$\Delta(q, \varepsilon) = q$$

$$\Delta(q, \sigma) = \delta(q, \sigma)$$

$$\Delta(q, \sigma_1...\sigma_{k+1}) = \delta(\Delta(q, \sigma_1...\sigma_k), \sigma_{k+1})$$

$\Delta(q, w)$ = the state of $M$ reached after reading in $w$, starting from state $q$

Note: $\Delta(q_0, w) \in F \iff M$ accepts $w$

Def. $w \in \Sigma^*$ distinguishes states $q_1$ and $q_2$ if exactly one of $\Delta(q_1, w), \Delta(q_2, w)$ is a final state
Distinguishing two states

Def. \( w \in \Sigma^* \) distinguishes states \( q_1 \) and \( q_2 \) if exactly one of \( \Delta(q_1, w) \), \( \Delta(q_2, w) \) is a final state.

I’m in \( q_1 \) or \( q_2 \), but which? How can I tell?

Ok, I’m accepting! Must have been \( q_1 \)

Here... read this

W
Fix $M = (Q, \Sigma, \delta, q_0, F)$ and let $p, q \in Q$

Definitions:

State $p$ is *distinguishable* from state $q$
if there is $w \in \Sigma^*$ that distinguishes $p$ and $q$
($\iff$ there is $w \in \Sigma^*$ so that
exactly one of $\Delta(p, w), \Delta(q, w)$ is a final state)

State $p$ is *indistinguishable* from state $q$
if $p$ is not distinguishable from $q$
($\iff$ for all $w \in \Sigma^*$, $\Delta(p, w) \in F \iff \Delta(q, w) \in F$)

*Pairs of indistinguishable states are redundant...*
Which pairs of states are distinguishable here?

ε distinguishes all final states from non-final states
Which pairs of states are distinguishable here?

The string 10 distinguishes $q_0$ and $q_3$
Which pairs of states are distinguishable here?

The string 0 distinguishes $q_1$ and $q_2$. 
Fix $M = (Q, \Sigma, \delta, q_0, F)$ and let $p, q, r \in Q$

Define a binary relation $\sim$ on the states of $M$:

$p \sim q$ iff $p$ is indistinguishable from $q$
$p \not\sim q$ iff $p$ is distinguishable from $q$

Proposition: $\sim$ is an equivalence relation

$p \sim p$ (reflexive)
$p \sim q \Rightarrow q \sim p$ (symmetric)
$p \sim q$ and $q \sim r \Rightarrow p \sim r$ (transitive)

Proof?
Fix $M = (Q, \Sigma, \delta, q_0, F)$ and let $p, q, r \in Q$

Therefore, the relation $\sim$ partitions $Q$ into disjoint equivalence classes

Proposition: $\sim$ is an equivalence relation

$[q] := \{ p \mid p \sim q \}$
Algorithm: MINIMIZE-DFA

Input: DFA $M$

Output: DFA $M_{\text{MIN}}$ such that:

$L(M) = L(M_{\text{MIN}})$

$M_{\text{MIN}}$ has no inaccessible states

$M_{\text{MIN}}$ is irreducible

For all states $p \neq q$ of $M_{\text{MIN}}$, $p$ and $q$ are distinguishable

Theorem: $M_{\text{MIN}}$ is the unique minimal DFA that is equivalent to $M$
Intuition:

The states of $M_{\text{MIN}}$ will be the equivalence classes of states of $M$

We’ll uncover these equivalent states with a *dynamic programming* algorithm
The Table-Filling Algorithm

Input: DFA $M = (Q, \Sigma, \delta, q_0, F)$

Output:
1. $D_M = \{(p, q) \mid p, q \in Q \text{ and } p \not\sim q\}$
2. $\text{EQUIV}_M = \{[q] \mid q \in Q\}$

High-Level Idea:
- We know how to find those pairs of states that the string $\varepsilon$ distinguishes...
- Use this and *iteration* to find those pairs distinguishable with *longer* strings
- The pairs of states left over will be indistinguishable
The Table-Filling Algorithm

Input: DFA $M = (Q, \Sigma, \delta, q_0, F)$

Output:  
1. $D_M = \{ (p, q) \mid p, q \in Q \text{ and } p \not\sim q \}$
2. $\text{EQUIV}_M = \{ [q] \mid q \in Q \}$

Base Case: For all $(p, q)$ such that $p$ accepts and $q$ rejects $\implies p \not\sim q$
The Table-Filling Algorithm

Input: DFA \( M = (Q, \Sigma, \delta, q_0, F) \)

Output:  
(1) \( D_M = \{ (p, q) \mid p, q \in Q \text{ and } p \sim q \} \)

(2) \( \text{EQUIV}_M = \{ [q] \mid q \in Q \} \)

Base Case: For all \( (p, q) \) such that \( p \) accepts and \( q \) rejects \( \Rightarrow p \sim q \)

Iterate: If there are states \( p, q \) and symbol \( \sigma \in \Sigma \) satisfying:

\[
\delta (p, \sigma) = p' \quad \text{mark}
\]

\[
\sim \Rightarrow p \sim q
\]

\[
\delta (q, \sigma) = q' \quad \text{mark}
\]

Repeat until no more \( D \)'s can be added
Claim: If \((p, q)\) is marked D by the Table-Filling algorithm, then \(p \not\sim q\)

Proof: By induction on the number of steps in the algorithm before \((p,q)\) is marked D

If \((p, q)\) is marked D at the start, then one state’s in F and the other isn’t, so \(\varepsilon\) distinguishes \(p\) and \(q\)

Suppose \((p, q)\) is marked D at a later point.

Then there are states \(p’, q’\) such that:

1. \((p’, q’)\) are marked D \(\Rightarrow p’ \not\sim q’\) (by induction)

So there’s a string \(w\) s.t. \(\Delta(p’, w) \in F \Leftrightarrow \Delta(q’, w) \notin F\)

2. \(p’ = \delta(p, \sigma)\) and \(q’ = \delta(q, \sigma)\), where \(\sigma \in \Sigma\)

The string \(\sigma w\) distinguishes \(p\) and \(q\)!
Claim: If \((p, q)\) is not marked \(D\) by the Table-Filling algorithm, then \(p \sim q\)

Proof (by contradiction):

Suppose the pair \((p, q)\) is not marked \(D\) by the algorithm, yet \(p \not\sim q\) (call this a “bad pair”)

Then there is a string \(w\) such that \(|w| > 0\) and:

\[ \Delta(p, w) \in F \quad \text{and} \quad \Delta(q, w) \not\in F \]  

(Why is \(|w| > 0\)?)

We have \((p', q')\) for some string \(w'\) and some \(\sigma \in \Sigma\)

Let \(p' = \delta(p, \sigma)\) and \(q' = \delta(q, \sigma)\)

Then \((p', q')\) is also a bad pair, but with a SHORTER distinguishing string, \(w'\)!
Algorithm MINIMIZE

Input: DFA M

Output: Equivalent minimal-state DFA $M_{\text{MIN}}$

1. Remove all inaccessible states from M

2. Run Table-Filling algorithm on M to get:
   $\text{EQUIV}_M = \{ [q] \mid q \text{ is an accessible state of } M \}$

3. Define: $M_{\text{MIN}} = (Q_{\text{MIN}}, \Sigma, \delta_{\text{MIN}}, q_{0 \text{MIN}}, F_{\text{MIN}})$
   
   $Q_{\text{MIN}} = \text{EQUIV}_M$, $q_{0 \text{MIN}} = [q_0]$, $F_{\text{MIN}} = \{ [q] \mid q \in F \}$

   $\delta_{\text{MIN}}([q], \sigma) = [\delta(q, \sigma)]$

Claim: $L(M_{\text{MIN}}) = L(M)$
Suppose for now the Claim is true.

If $M'$ is a minimal DFA, then $M'$ has no inaccessible states and is irreducible (why?) So the Claim implies:

Let $M'$ be a minimal DFA for $M$.

$\Rightarrow$ there is an isomorphism between $M'$ and the DFA $M_{\text{MIN}}$ that is output by $\text{MINIMIZE}(M)$.

$\Rightarrow$ The Thm holds!
Thm: $M_{MIN}$ is the unique minimal DFA equivalent to $M$

Claim: If $L(M')=L(M_{MIN})$ and $M'$ has no inaccessible states and $M'$ is irreducible $\implies$ there is an isomorphism between $M'$ and $M_{MIN}$

Proof: We recursively construct a map from the states of $M_{MIN}$ to the states of $M'$

Base Case: $q_{0\,MIN} \mapsto q_{0\,'}$

Recursive Step: If $p \mapsto p'$

Then $q \mapsto q'$
Base Case: $q_0 \text{MIN} \rightarrow q_0'$

Recursive Step: If $p \rightarrow p'$

Then $q \rightarrow q'$

\[
\begin{array}{c}
\sigma \\
\sigma \\
q \\
q'
\end{array}
\]
Base Case: $q_{0\ MIN} \mapsto q_{0\ '}$

Recursive Step: If $p \mapsto p'$

Then $q \mapsto q'$

We need to prove:

The map is defined everywhere
The map is well defined
The map is a bijection
The map preserves all transitions:
If $p \mapsto p'$ then $\delta_{MIN}(p, \sigma) \mapsto \delta'(p', \sigma)$

(this follows from the definition of the map!)
Base Case: $q_0^{MIN} \mapsto q_0'$

Recursive Step: If $p \mapsto p'$

Then $q \mapsto q'$

Let $q' = \Delta'(q_0',w)$. Then $q \mapsto q'$

The map is defined everywhere

That is, for all states $q$ of $M_{MIN}$

there is a state $q'$ of $M'$ such that $q \mapsto q'$

If $q \in M_{MIN}$, there is a string $w$ such that $\Delta_{MIN}(q_{0 \ MIN},w) = q$
The map is well defined

Proof by contradiction.

Suppose there are states $q'$ and $q''$ such that $q \mapsto q'$ and $q \mapsto q''$

We show that $q'$ and $q''$ are indistinguishable, so it must be that $q' = q''$

Base Case: $q_{0\text{MIN}} \mapsto q_0'$

Recursive Step: If $p \mapsto p'$

\[
\begin{array}{c}
p \\
\sigma \\
q \\
\end{array}
\quad \begin{array}{c}
p' \\
\sigma \\
q' \\
\end{array}
\quad \text{Then } q \mapsto q'
\]
Suppose there are states $q'$ and $q''$ such that $q \mapsto q'$ and $q \mapsto q''$

Suppose $q'$ and $q''$ are distinguishable
Base Case: $q_0^{\text{MIN}} \mapsto q_0'$

Recursive Step: If $p \mapsto p'$
\[ \sigma \downarrow \quad \sigma \downarrow \]
\[ q \mapsto q' \]

Then $q \mapsto q'$

The map is onto

Want to show: For all states $q'$ of $M'$ there is a state $q$ of $M_{\text{MIN}}$ such that $q \mapsto q'$

For every $q'$ there is a string $w$ such that $M'$ reaches state $q'$ after reading in $w$

Let $q$ be the state of $M_{\text{MIN}}$ after reading in $w$

Claim: $q \mapsto q'$
The map is one-to-one

*Proof by contradiction.* Suppose there are states $p \neq q$ such that $p \mapsto q'$ and $q \mapsto q'$

If $p \neq q$, then $p$ and $q$ are distinguishable
How can we prove that two regular expressions are equivalent?
Parting thoughts:
DFAs can be optimized
Later: Can DFAs be learned?