CS154

Non-Regular Languages, Minimizing DFAs

CS154

Homework 1 is due!

Homework 2 will appear
this afternoon

The Pumping Lemma: Structure in Regular Languages

Let L be a regular language

Then there is a positive integer P s.t.

for all strings $w \in L$ with $|w| \geq P$
there is a way to write $w = xyz$, where:

1. $|y| > 0$ (that is, $y \neq \varepsilon$)
2. $|xy| \leq P$
3. For *all* $i \geq 0$, $xy^iz \in L$

Why is it called the pumping lemma? The word w gets *pumped* into longer and longer strings…

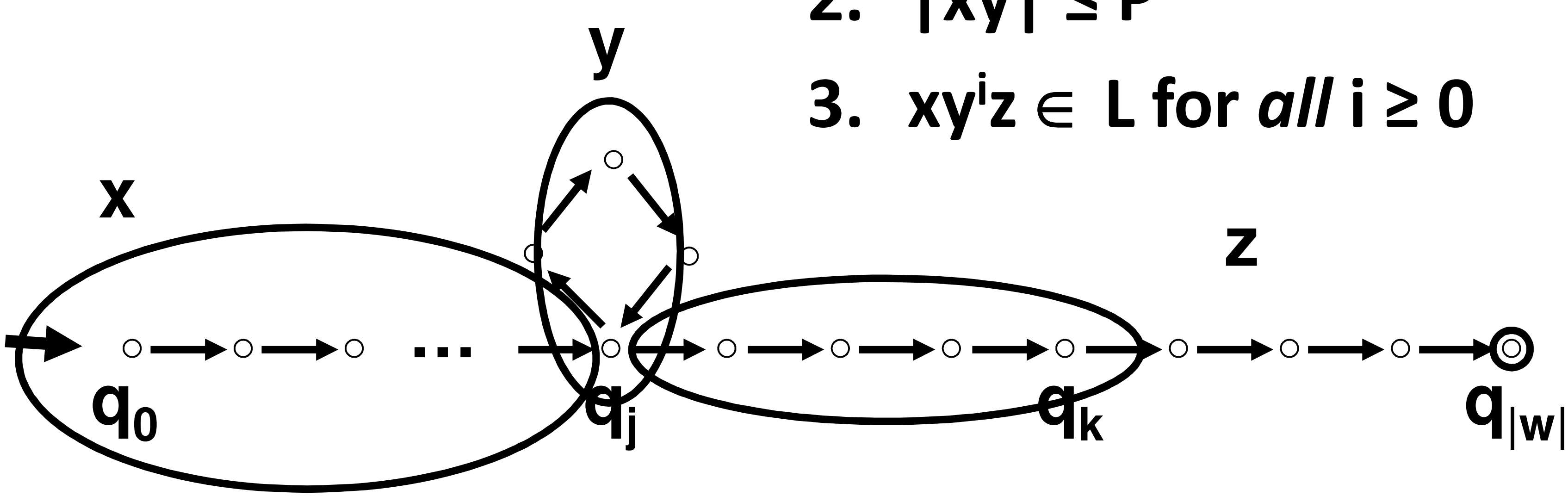
Proof: Let M be a DFA that recognizes L

Let P be the number of states in M

Let w be a string where $w \in L$ and $|w| \geq P$

We show: $w = xyz$

1. $|y| > 0$
2. $|xy| \leq P$
3. $xy^iz \in L$ for *all* $i \geq 0$



Claim: There must exist j and k such that
$0 \leq j < k \leq P$, and $q_j = q_k$

Applying the Pumping Lemma

Let's prove that
EQ = { w | w has equal number of 1s and 0s}
is not regular.

By contradiction. Assume EQ is regular.
Let P be as in pumping lemma. Let $w = 0^P1^P$; note $w \in$ EQ.

If EQ is regular, then there is a way to write w as $w = xyz$, $|y| > 0$, $|xy| \leq P$, and for all $i \geq 0$, xy^iz is *also* in EQ

Claim: The string y must be all zeroes.

Why? Because $|xy| \leq P$ and $w = xyz = 0^P1^P$

But then xyyz has more 0s than 1s *Contradiction!*

Applying the Pumping Lemma

Let's prove that

SQ = $\{0^{n^2} \mid n \geq 0\}$ is not regular

Assume SQ is regular. Let $w = 0^{P^2}$

If SQ is regular, then we can write $w = xyz$, $|y| > 0$, $|xy| \leq P$, and for any $i \geq 0$, xy^iz is *also* in SQ

So $xyyz \in SQ$. Note that $xyyz = 0^{P^2+|y|}$

Note that $0 < |y| \leq P$

So $|xyyz| = P^2 + |y| \leq P^2 + P < P^2 + 2P + 1 = (P+1)^2$

and $P^2 < |xyyz| < (P+1)^2$

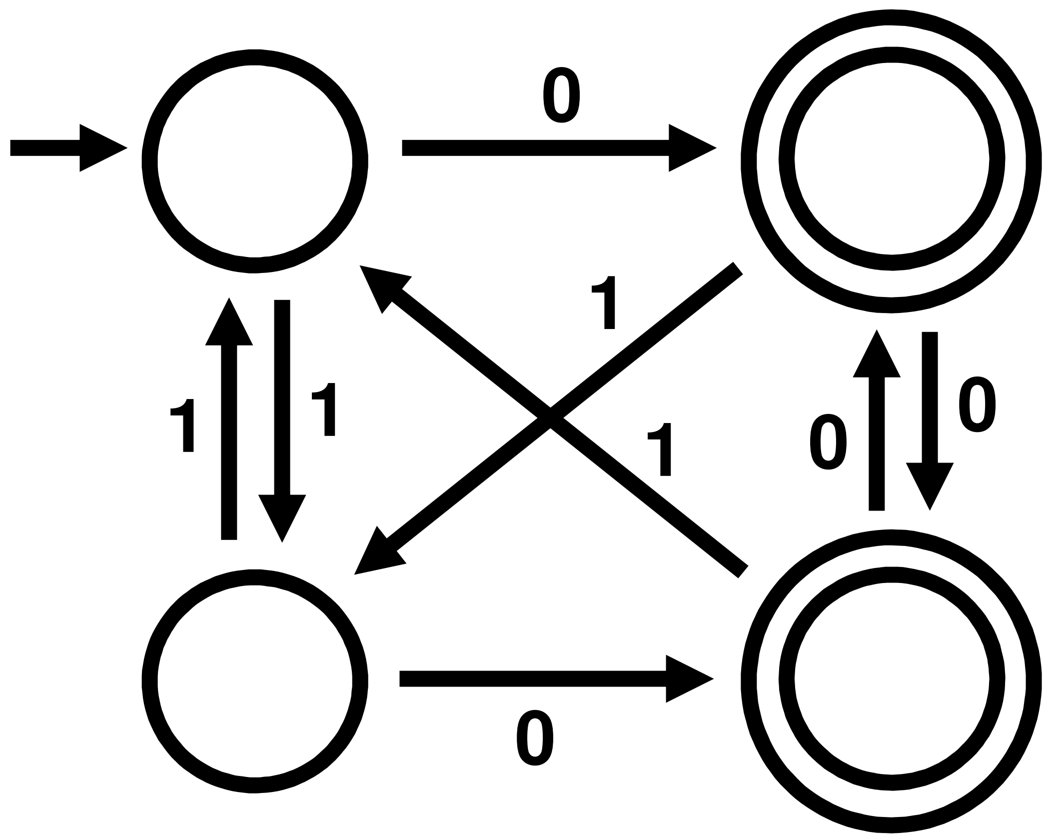
Therefore $|xyyz|$ is *not* a perfect square!

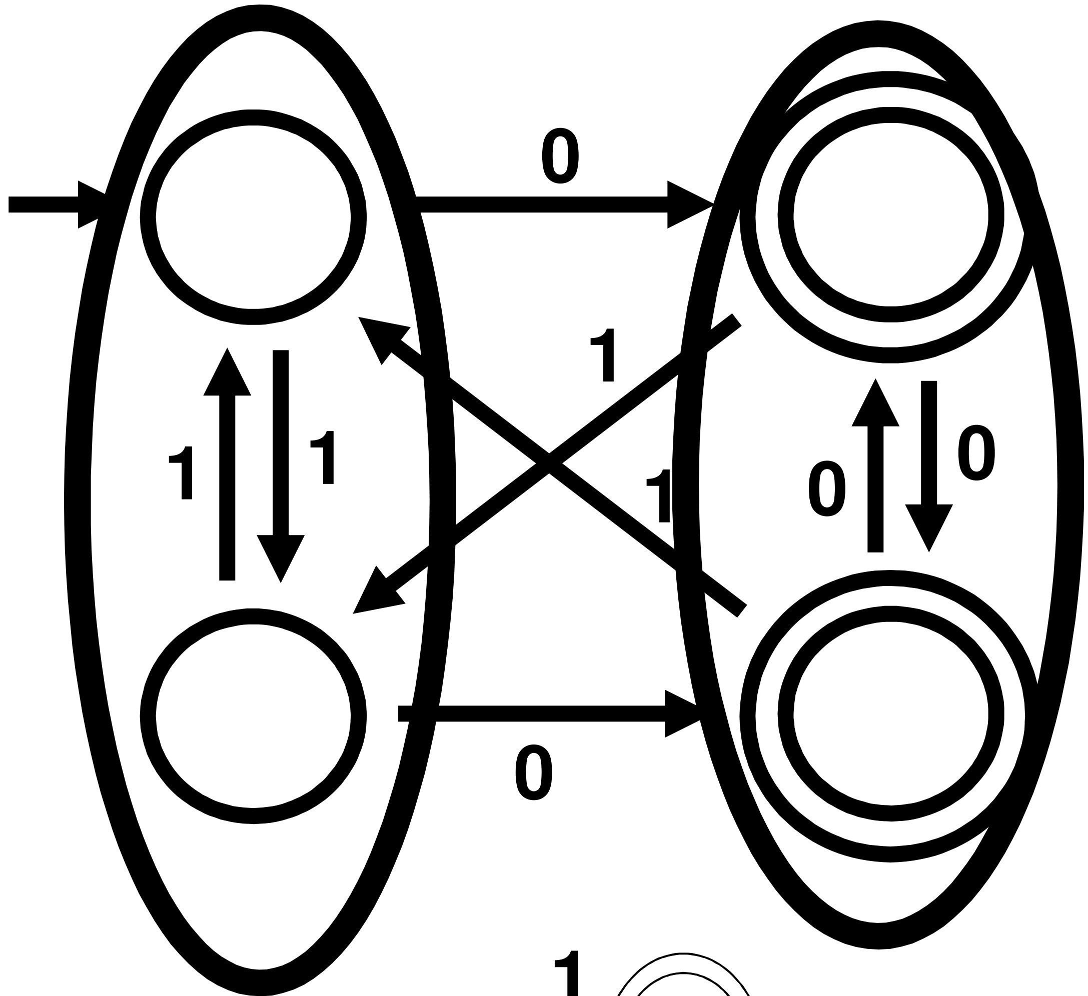
Hence $0^{P^2+|y|} = xyyz \notin SQ$, so our assumption must be false.

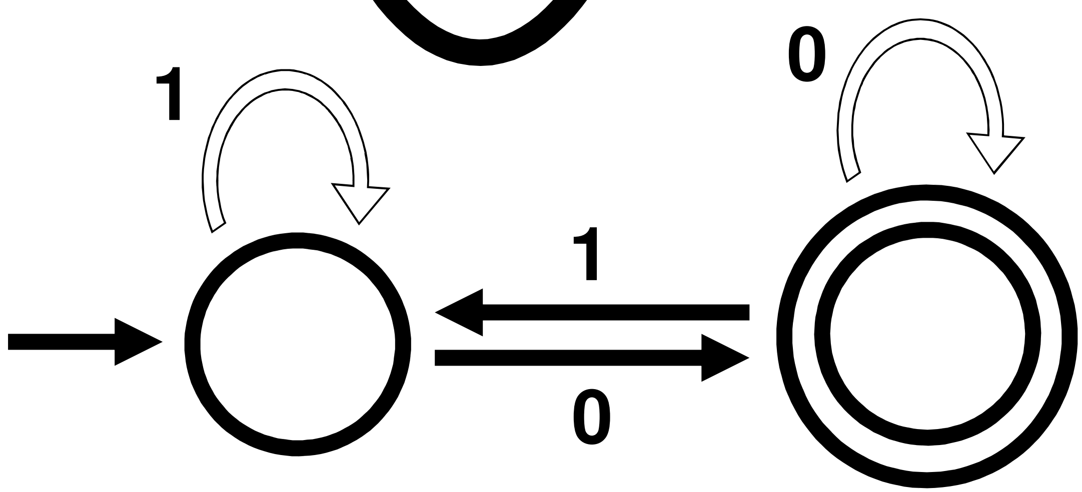
That is, SQ is not regular!

Does this DFA have a minimal number of states?

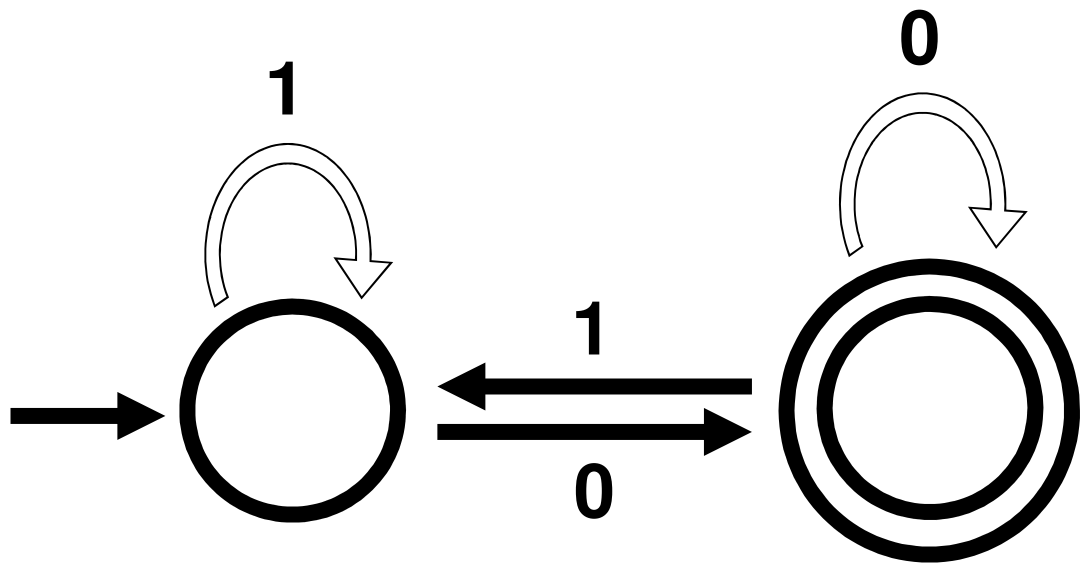
NO




0
1
1
1
1
0
0
0


1
0
1
0

Is this minimal?



How can we tell in general?

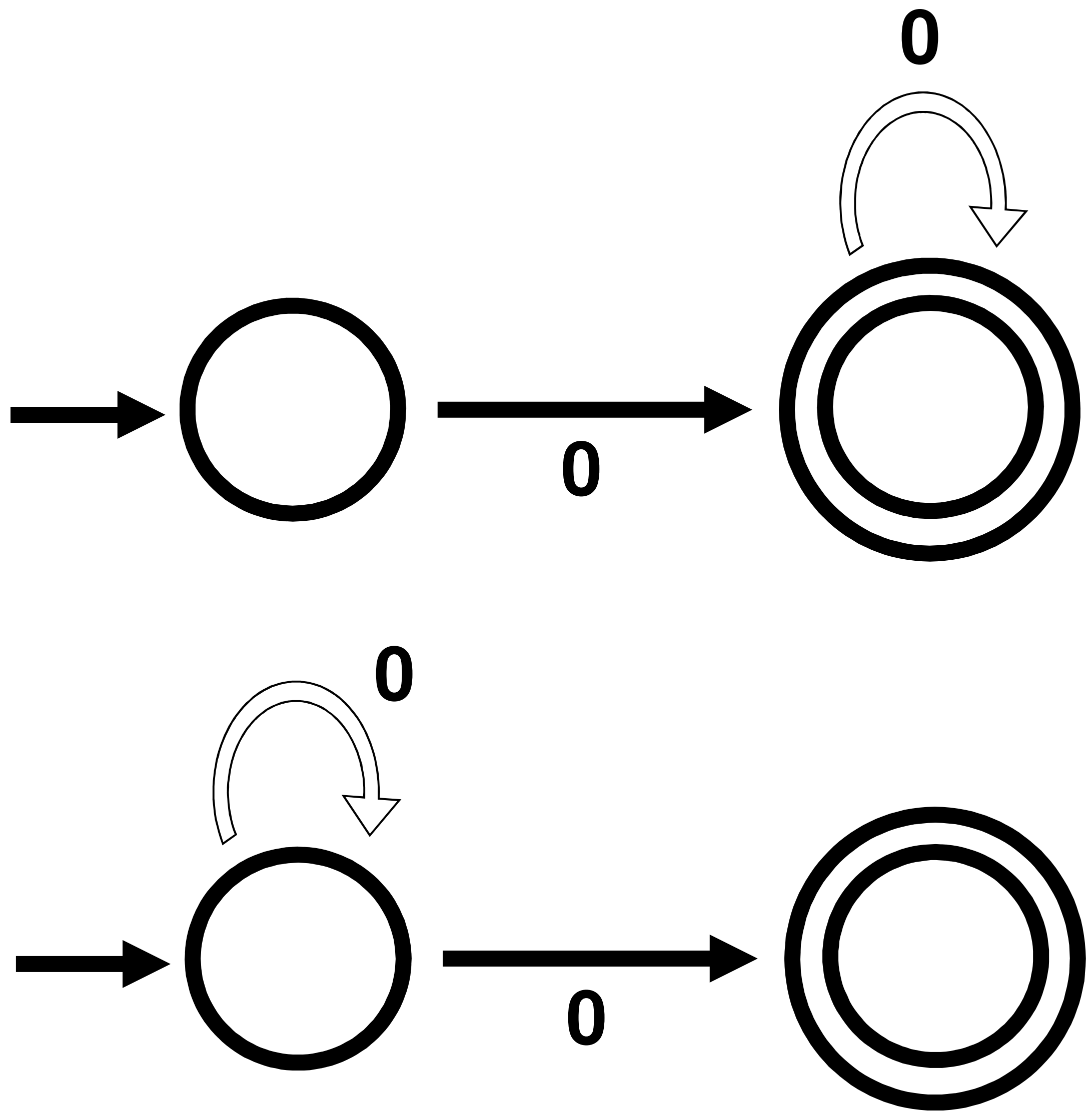
Theorem:

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 this 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 δ to strings

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

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

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

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

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

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

Def. $w \in \Sigma^*$ **distinguishes states q_1 and q_2 iff**

$$\Delta(q_1, w) \in F \Leftrightarrow \Delta(q_2, w) \notin F$$

Extending transition function δ to strings

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

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

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

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

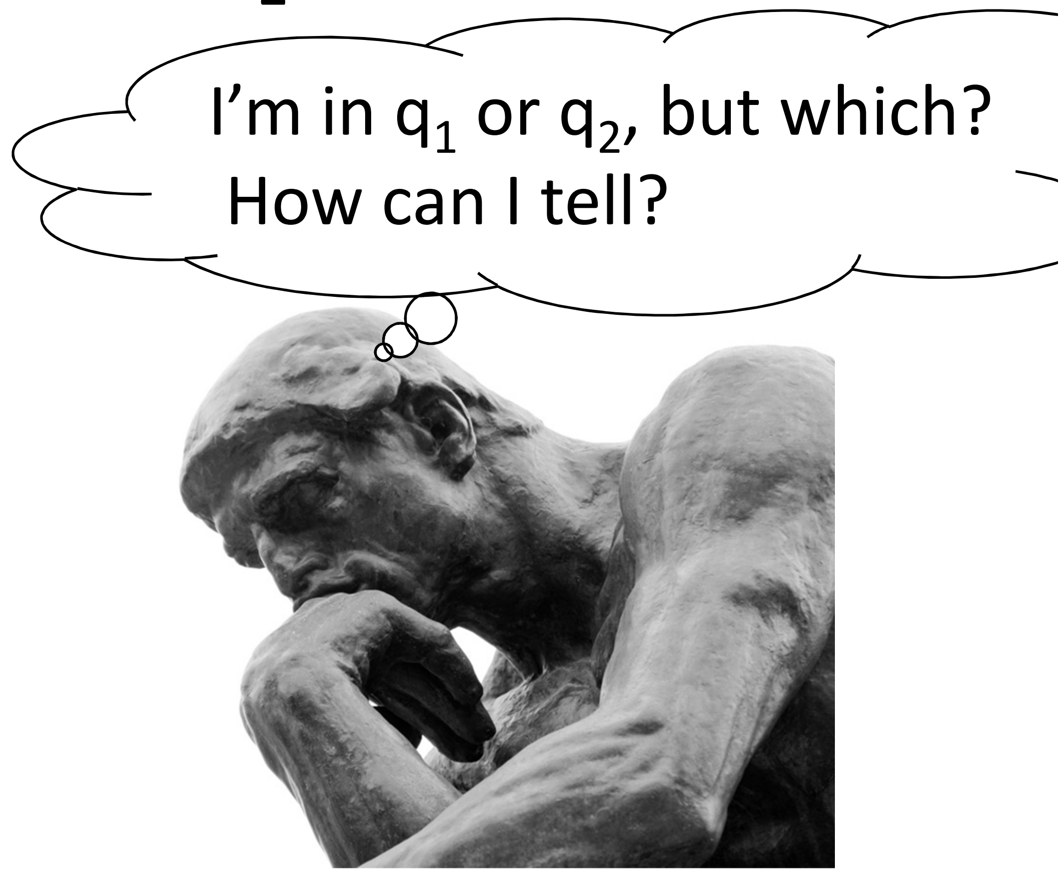
$\Delta(q, w)$ = *the state of M reached after reading in w, starting from state q*

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

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

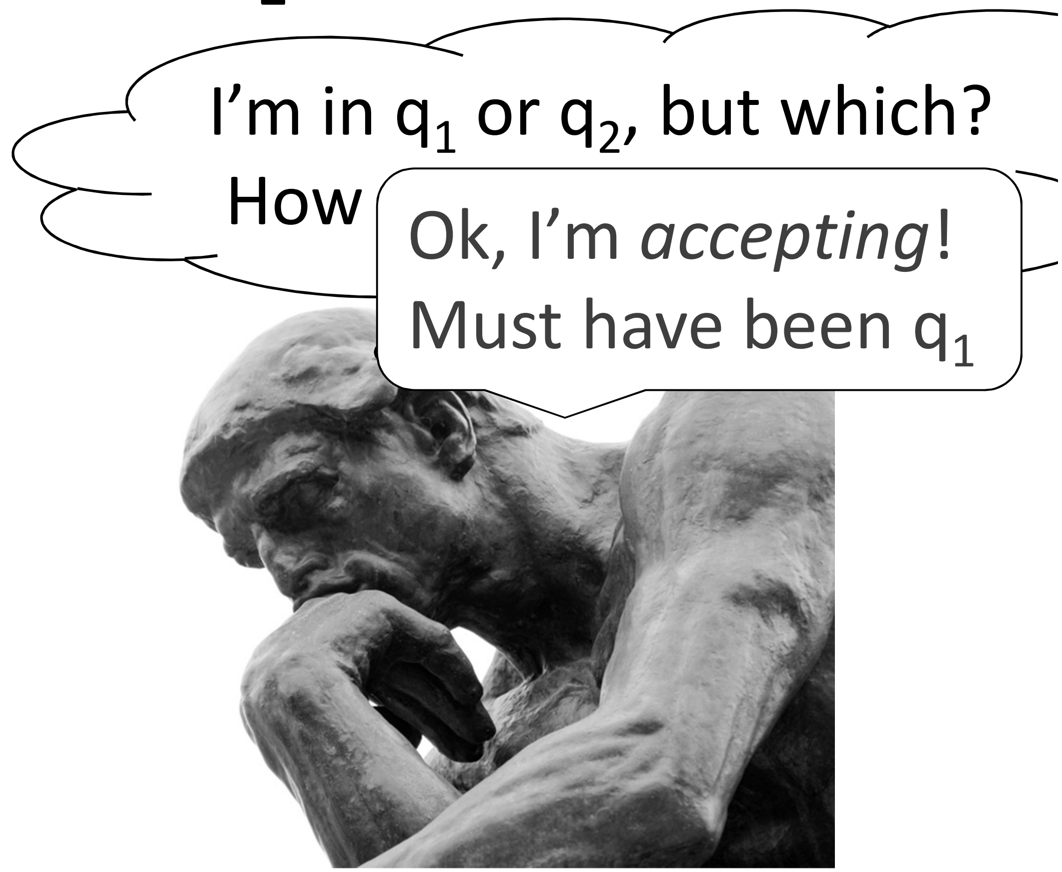
Distinguishing two states

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



Distinguishing two states

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



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

Definition:

State p is *distinguishable* from state q

iff there is $w \in \Sigma^*$ that distinguishes p and q

iff there is $w \in \Sigma^*$ so that

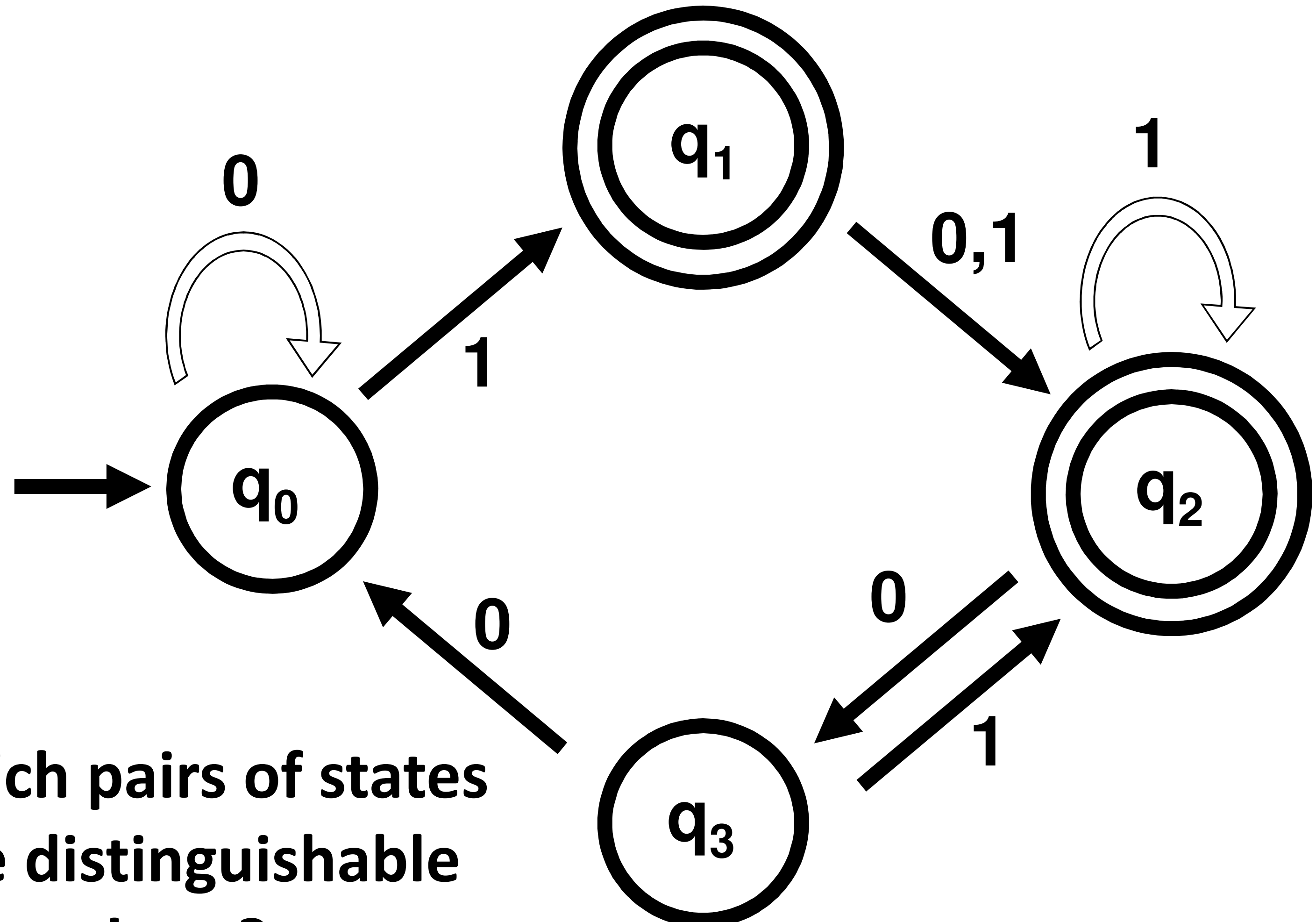
exactly *one* of $\Delta(p, w)$, $\Delta(q, w)$ is a final state

State p is *indistinguishable* from state q

iff p is not distinguishable from q

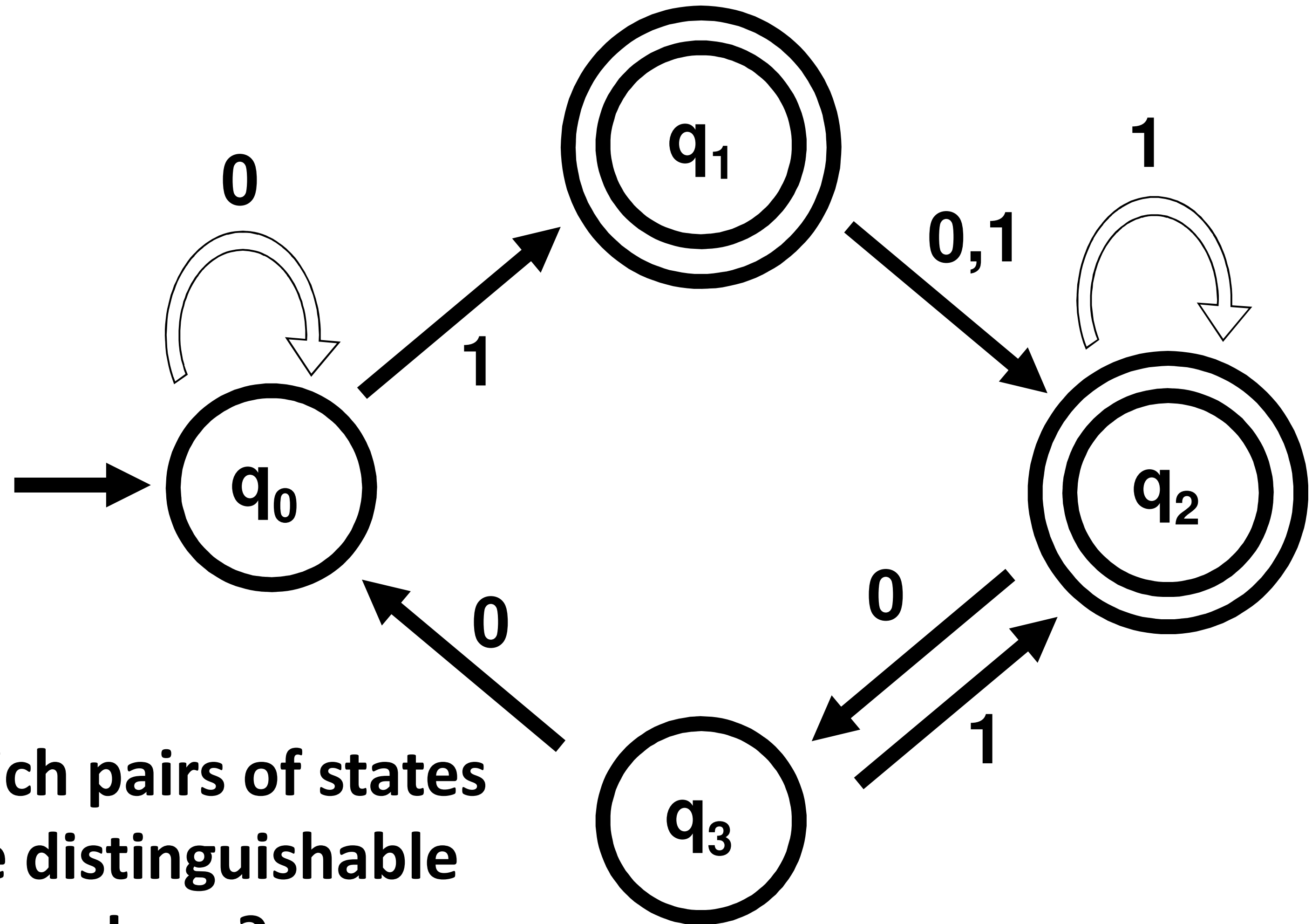
iff for all $w \in \Sigma^*$, $\Delta(p, w) \in F \Leftrightarrow \Delta(q, w) \in F$

Pairs of indistinguishable states are redundant…



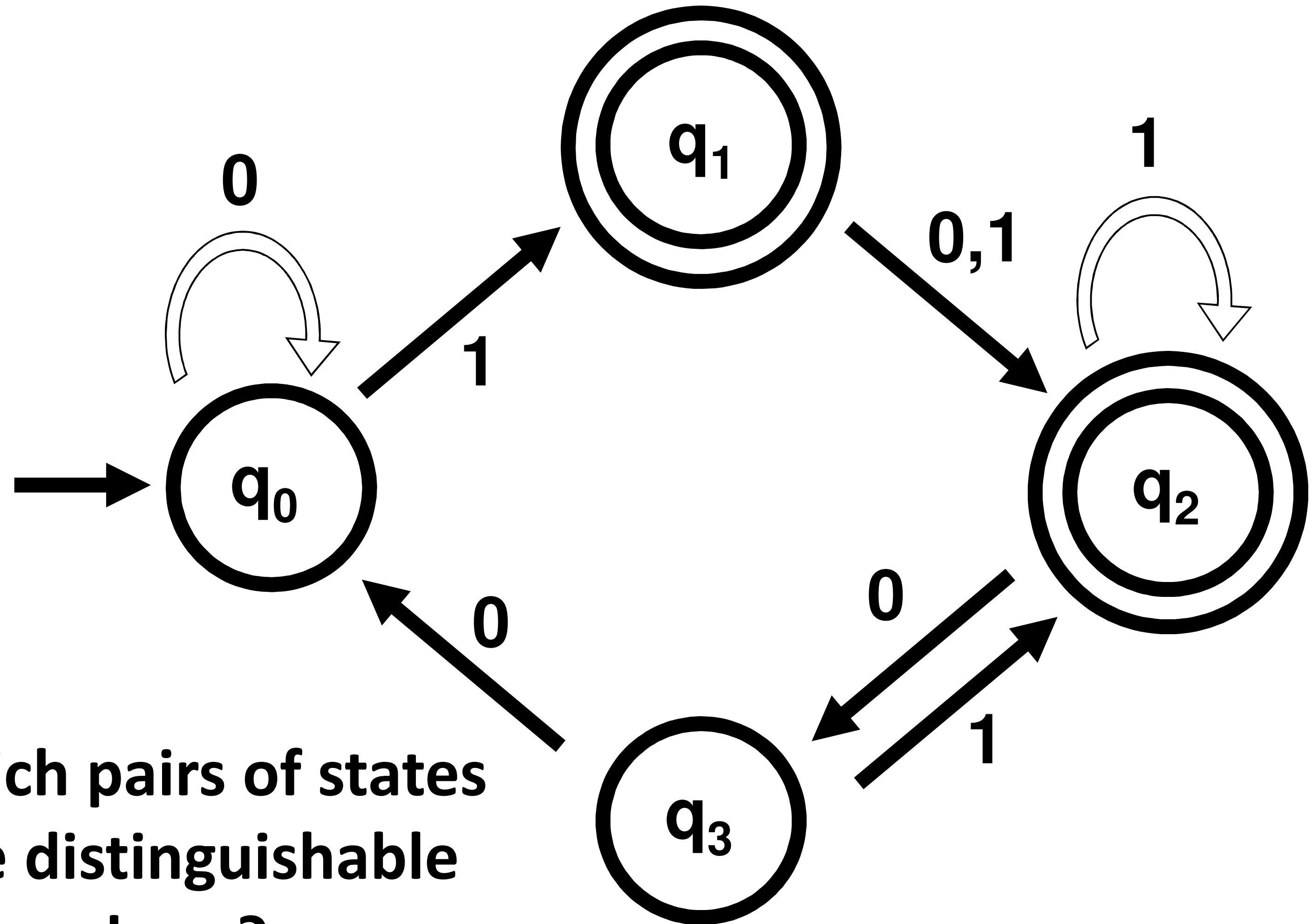
Which pairs of states are distinguishable here?

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

$p \nsim q$ 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)

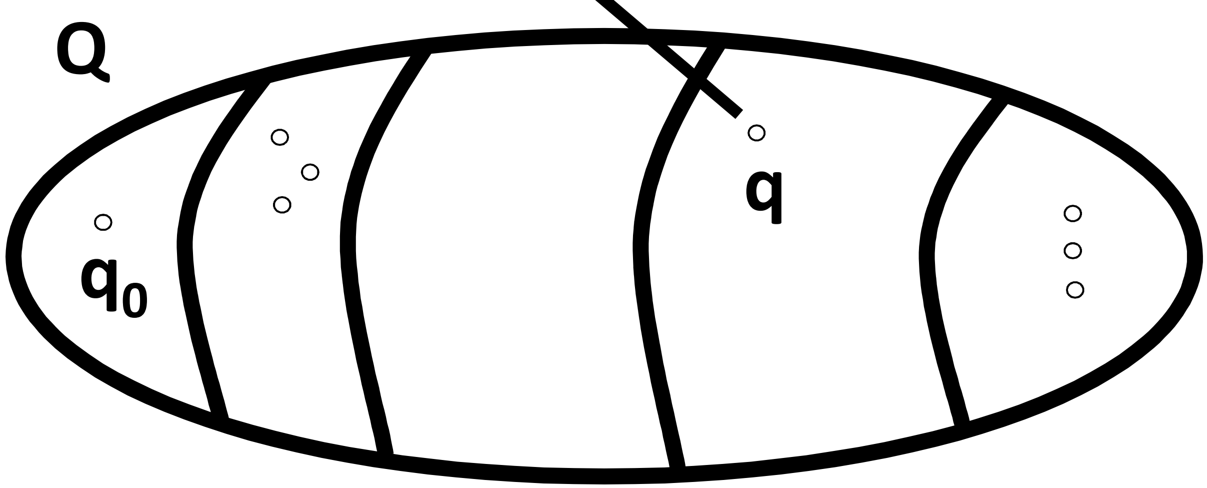
Proof?

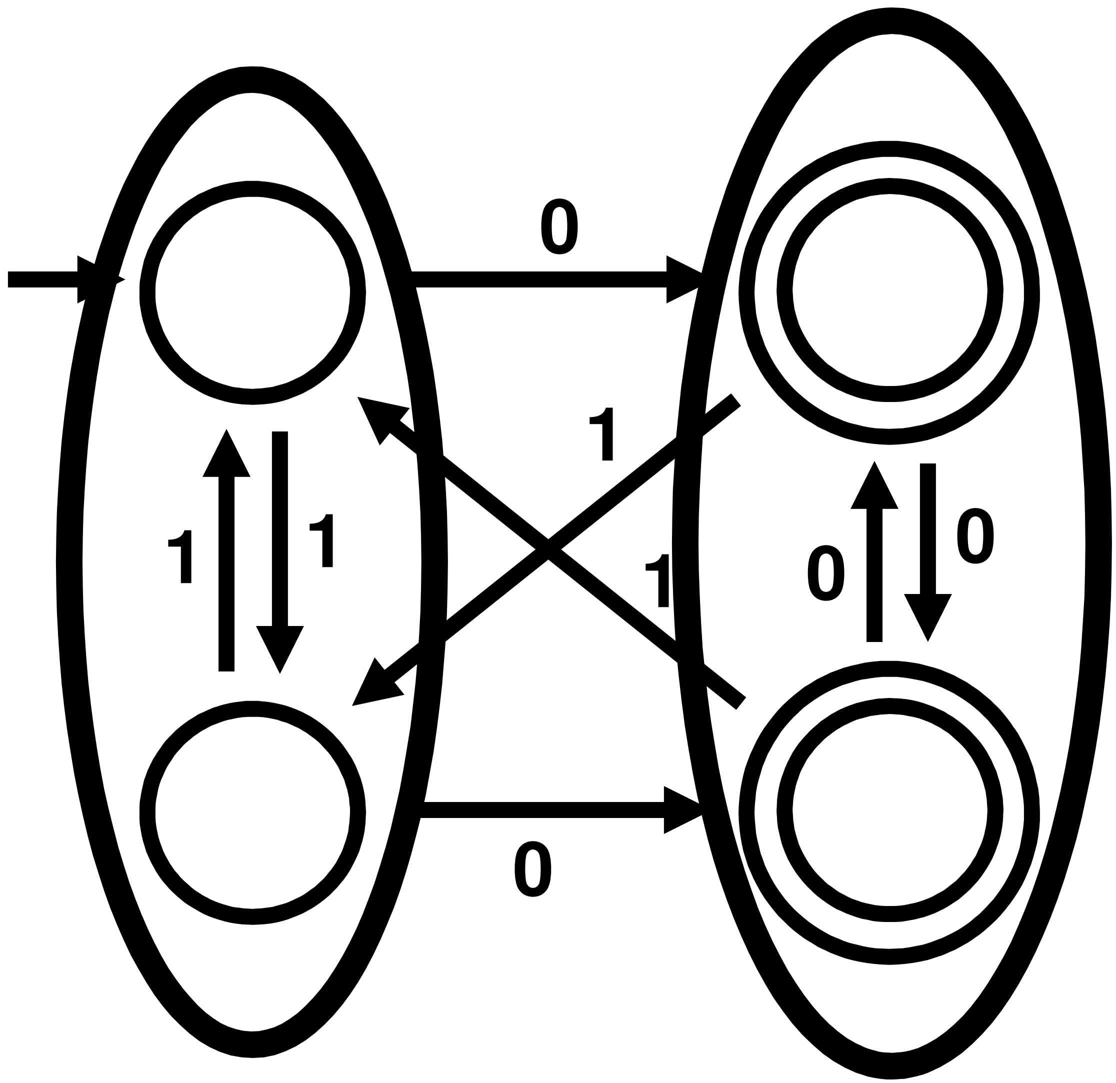
Fix $M = (Q, \Sigma, \delta, q_0, F)$ and let $p, q, r \in Q$

Proposition: $\sim$ is an equivalence relation

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

$[q] := \{ p \mid p \sim q \}$




0
1
1
1
1
0
0
0

Algorithm: MINIMIZE-DFA

Input: DFA M

Output: DFA M_{MIN} such that:

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

M_{MIN} has no *inaccessible* states

M_{MIN} is *irreducible*

||

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

Theorem: M_{MIN} is the unique minimal DFA that is equivalent to M

Intuition:
The states of M_{MIN} will be the *equivalence classes* of states of M

We'll uncover these equivalent states with a *dynamic programming* algorithm

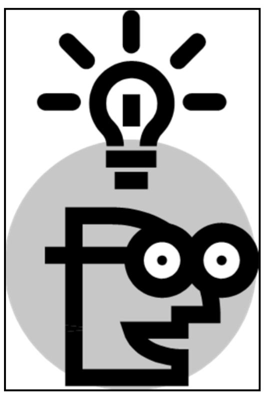
The Table-Filling Algorithm

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

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

(2) $EQUIV_M = \{ [q] \mid q \in Q \}$

High-Level Idea:



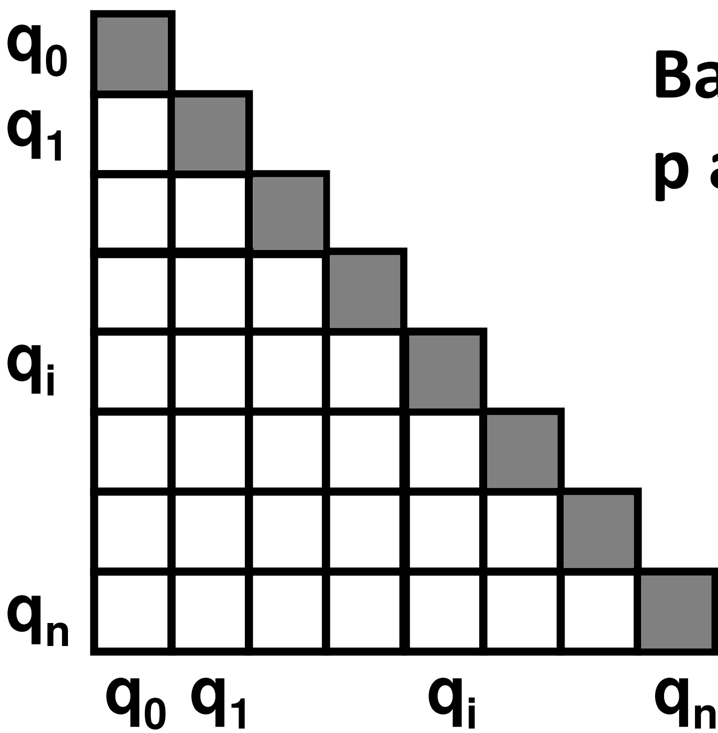
- **We know how to find those pairs of states that the string ε distinguishes…**
- **Use this and *iteration* to find those pairs distinguishable with *longer* strings**
- **The pairs of states left over will be indistinguishable**

The Table-Filling Algorithm

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

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

(2) $EQUIV_M = \{ [q] \mid q \in Q \}$



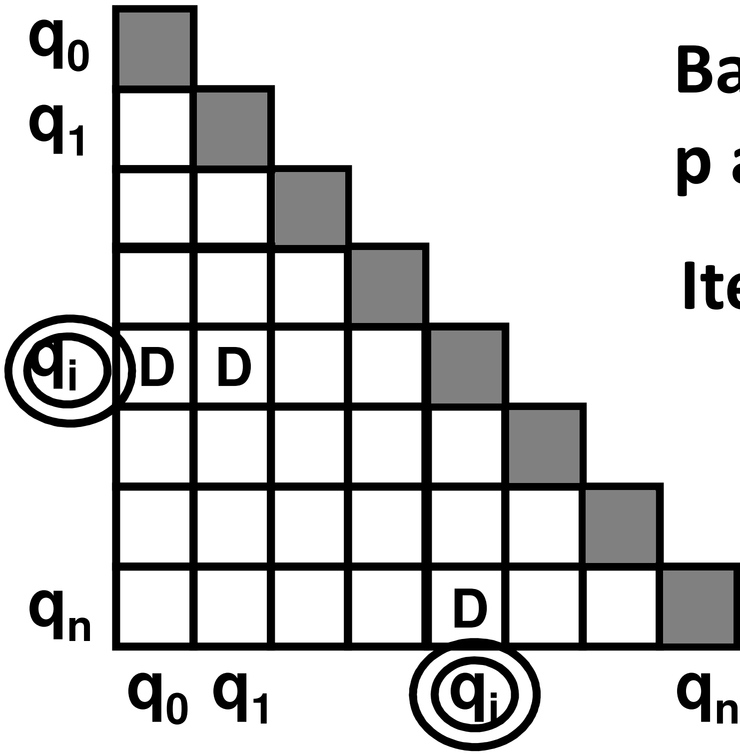
Base Case: For all (p, q) such that p accepts and q rejects $\Rightarrow$ $p \nsim q$

The Table-Filling Algorithm

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

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

(2) $EQUIV_M = \{ [q] \mid q \in Q \}$



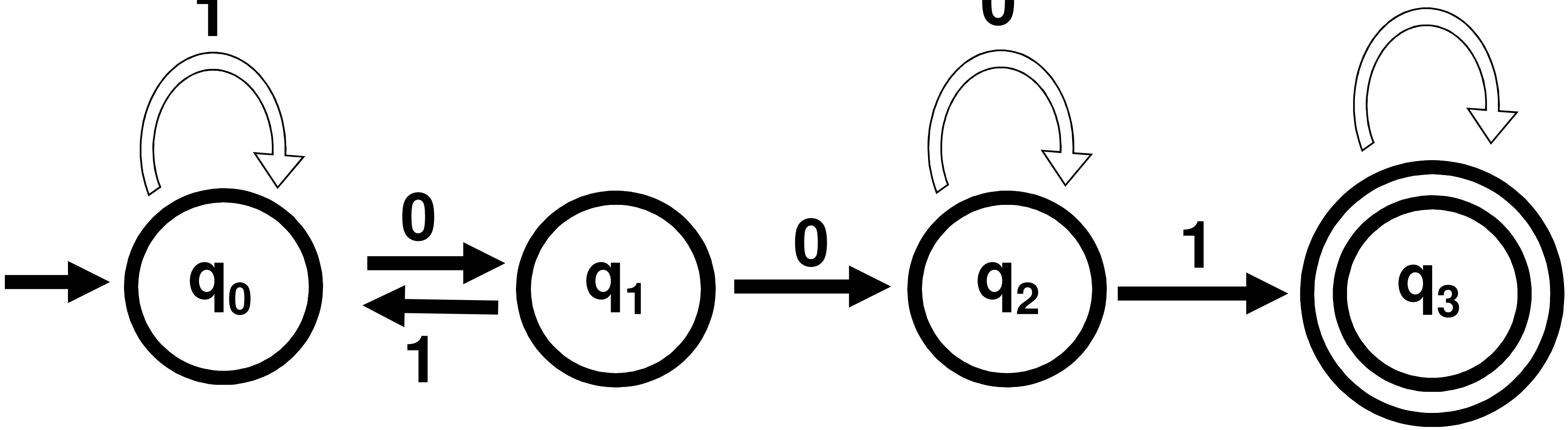
Base Case: For all (p, q) such that p accepts and q rejects $\Rightarrow p \nsim q$

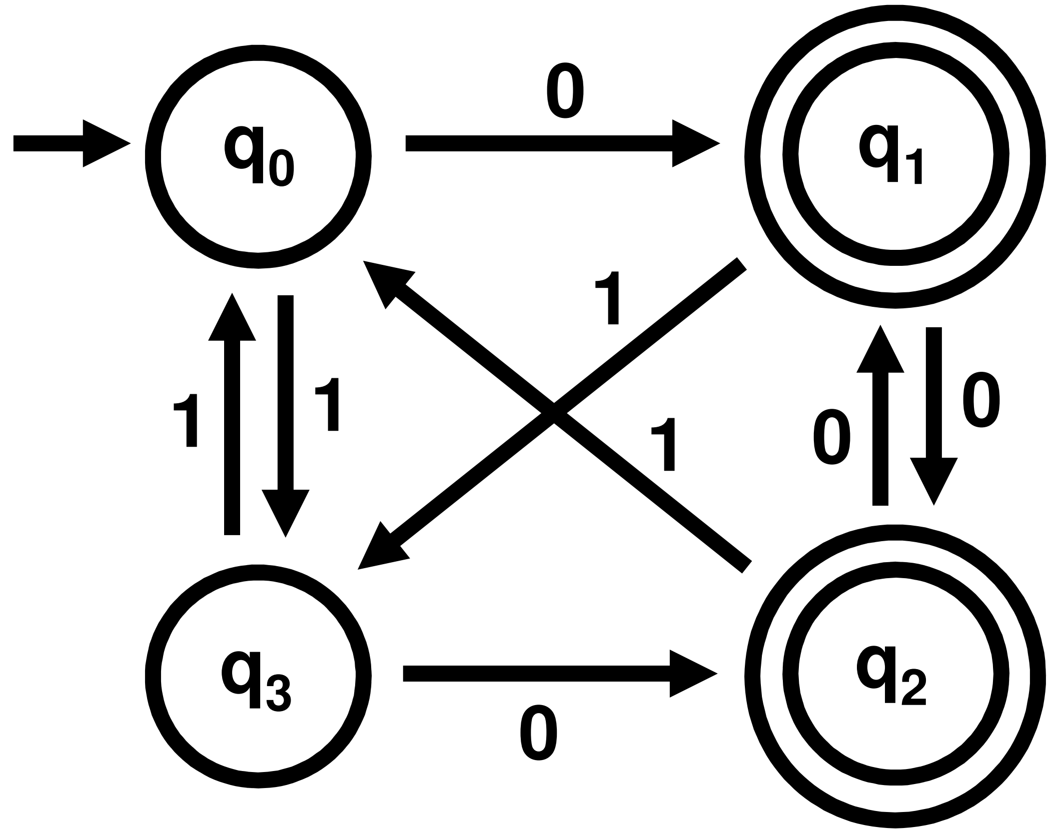
Iterate: If there are states p, q and symbol $\sigma \in \Sigma$ satisfying:

$$\begin{array}{l} \delta(p, \sigma) = p' \\ \qquad\qquad\quad \nsim \\ \delta(q, \sigma) = q' \end{array} \Rightarrow \text{mark } p \nsim q$$

Repeat until no more D's can be added

	q_0	q_1	q_2	q_3
q_0				
q_1	D			
q_2	D	D		
q_3	D	D	D	





	q_0	q_1	q_2	q_3
q_0				
q_1	D			
q_2	D			
q_3		D	D	

Claim: If (p, q) is marked D by the Table-Filling algorithm, then $p \nsim q$

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 ε 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 $\Rightarrow$ $p' \nsim q'$ (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 σw 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 $\nsim$ q (call this a "bad pair")

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

Δ(p, w) $\in$ F and Δ(q, w) $\notin$ F **(Why is |w| > 0?)**

Of all such bad pairs, let p, q be a pair with the *shortest* distinguishing string w

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 $p \nsim q$ (call this a "bad pair")

Of all such bad pairs, let p, q be a pair with the *shortest* distinguishing string w

$\Delta(p, w) \in F$ and $\Delta(q, w) \notin F$ (Why is $|w| > 0$?)

We have $w = \sigma w'$, 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_{MIN}

1. **Remove all inaccessible states from M**

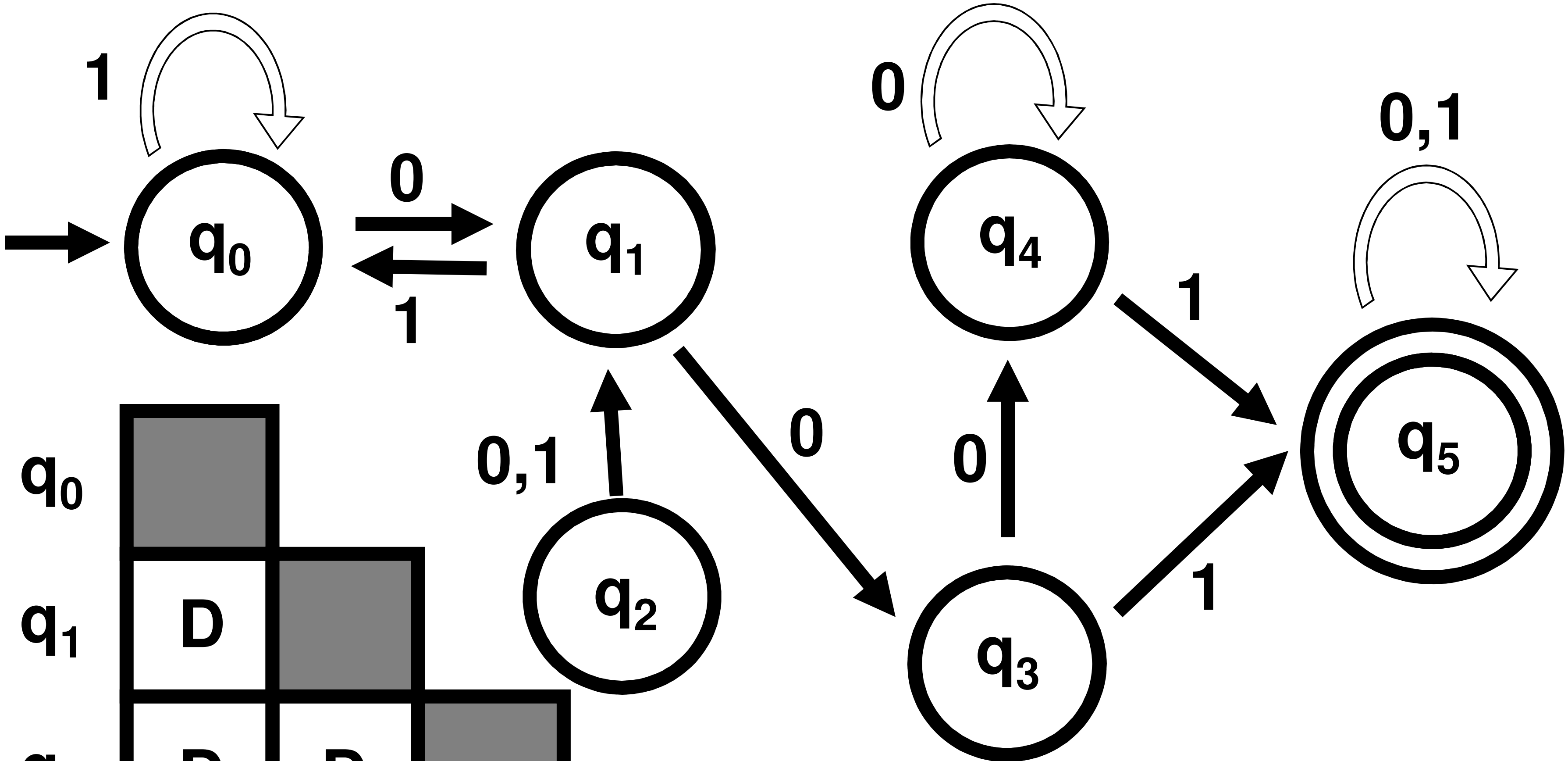
2. **Run Table-Filling algorithm on M to get:**
$EQUIV_M$ = { [q] | q is an accessible state of M }

3. **Define: $M_{MIN} = (Q_{MIN}, \Sigma, \delta_{MIN}, q_{0\,MIN}, F_{MIN})$**

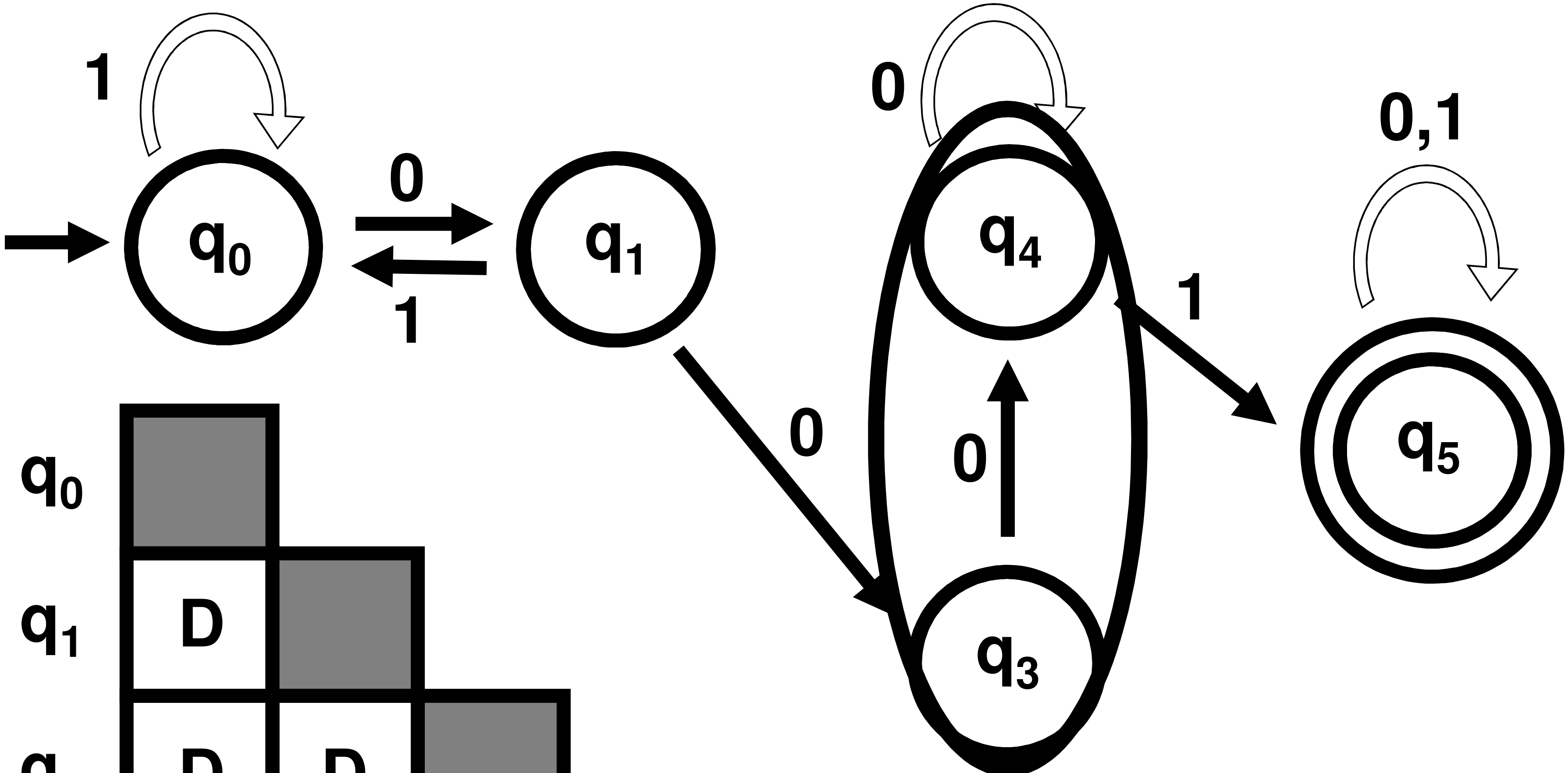
$$Q_{MIN} = EQUIV_M,\ \ q_{0\,MIN} = [q_0],\ \ F_{MIN} = \{ [q] \mid q \in F \}$$

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

Claim: $L(M_{MIN}) = L(M)$



	q_0	q_1	q_3	q_4	q_5
q_0					
q_1	D				
q_3	D	D			
q_4	D	D			
q_5	D	D	D	D	



	q_0	q_1	q_3	q_4	q_5
q_0					
q_1	D				
q_3	D	D			
q_4	D	D			
q_5	D	D	D	D	

Thm: M_{MIN} is the unique minimal DFA equivalent to M

Claim: Suppose for a DFA M', L(M′)=L(M_{MIN}) and M′ has no inaccessible states and M' is irreducible. Then there is an *isomorphism* between M′ and M_{MIN}

If M' is a minimal DFA, then M' has no inaccessible states and is irreducible. So the Claim implies:

If M' is a minimal DFA for M, then there is an isomorphism between M' and M_{MIN}. So the Thm holds!

Corollary: If M has no inaccessible states and is irreducible, then M is minimal.

Proof: Let M^{min} be minimal for M. Then L(M) = L(M^{min}), no inaccessible states in M, and M is irreducible.

By Claim, both M^{min} and M are isomorphic to M_{MIN}!