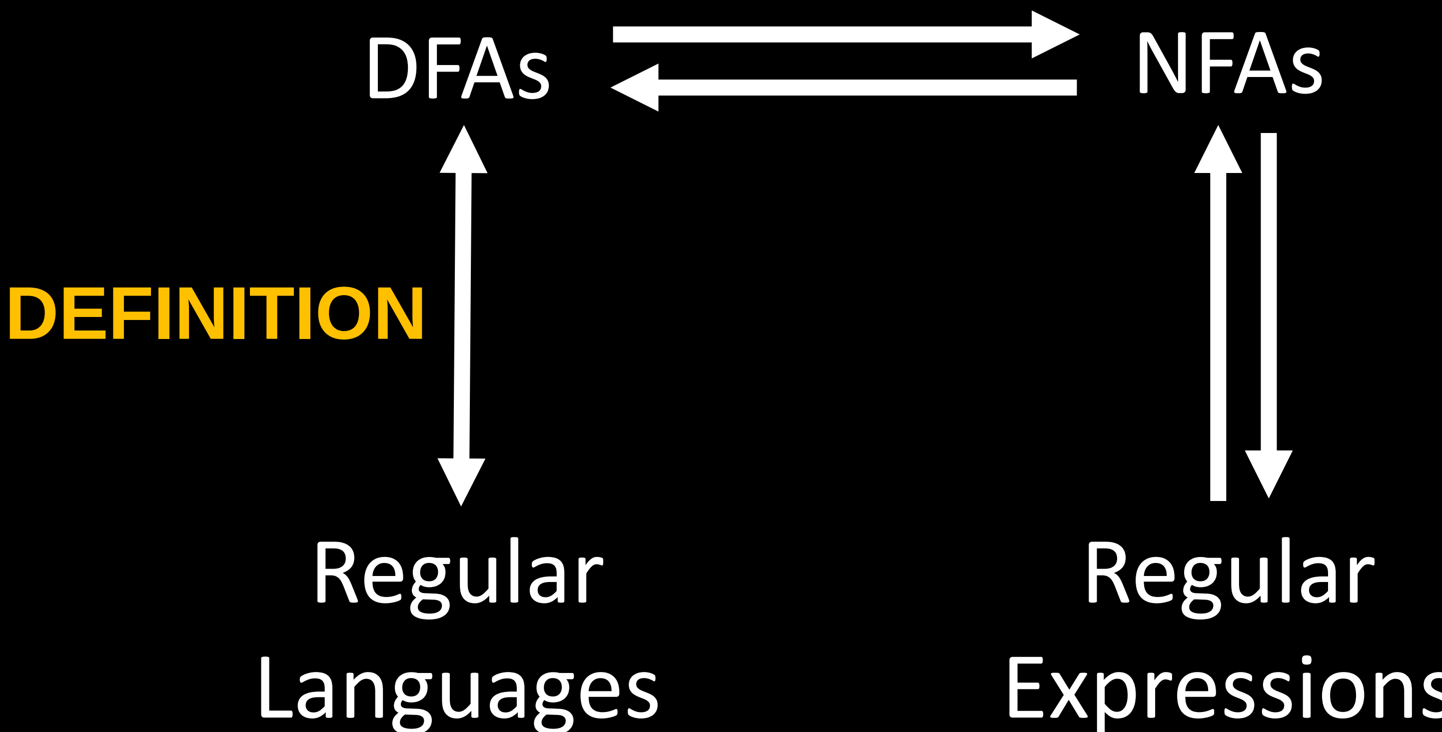
6.045

Lecture 5: Minimizing DFAs

6.045

Announcements:

- Pset 2 is up (as of last night)
  - *Dylan says:* `"It's fire."`
- How was Pset 1?

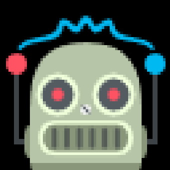

DFAs
NFAs
DEFINITION
Regular Languages
Regular Expressions

Some Languages Are Not Regular:

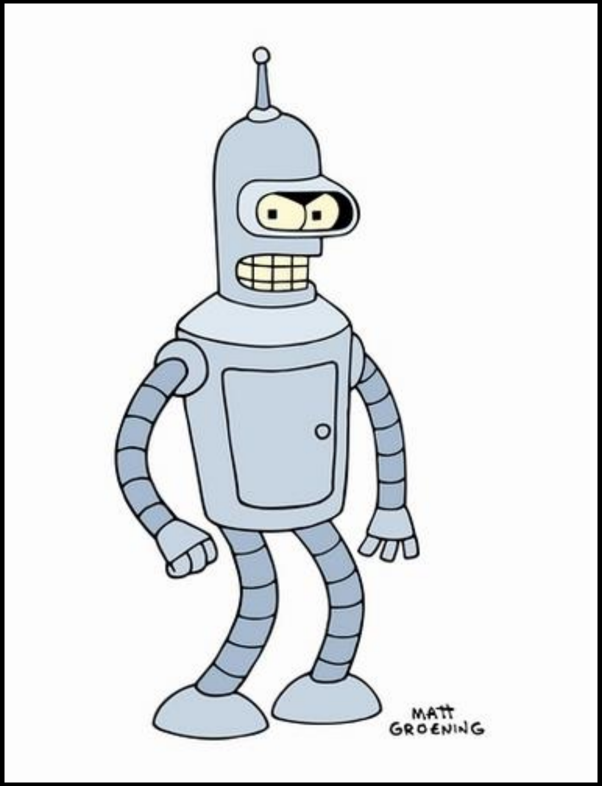
Limitations on DFAs/NFAs

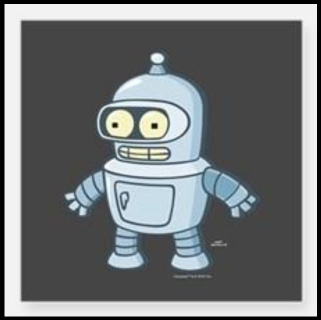
a.k.a.

"Lower Bounds" on DFAs/NFAs

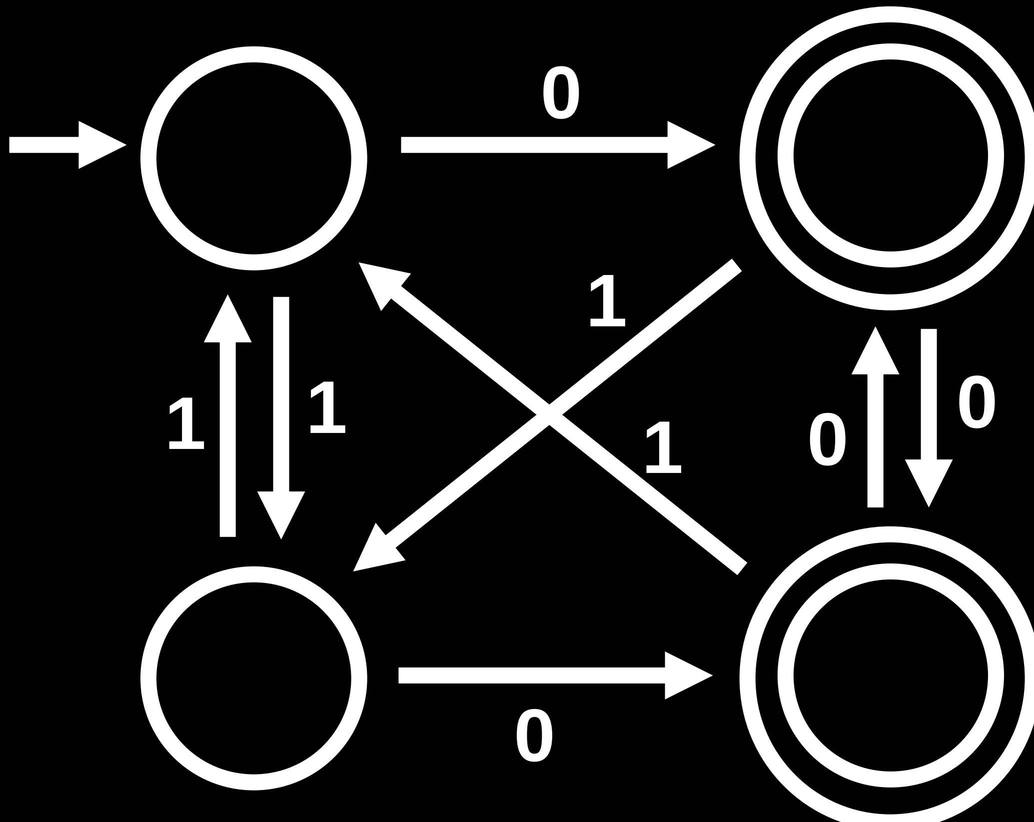
 **How to Make a DFA Lose Its Mind**

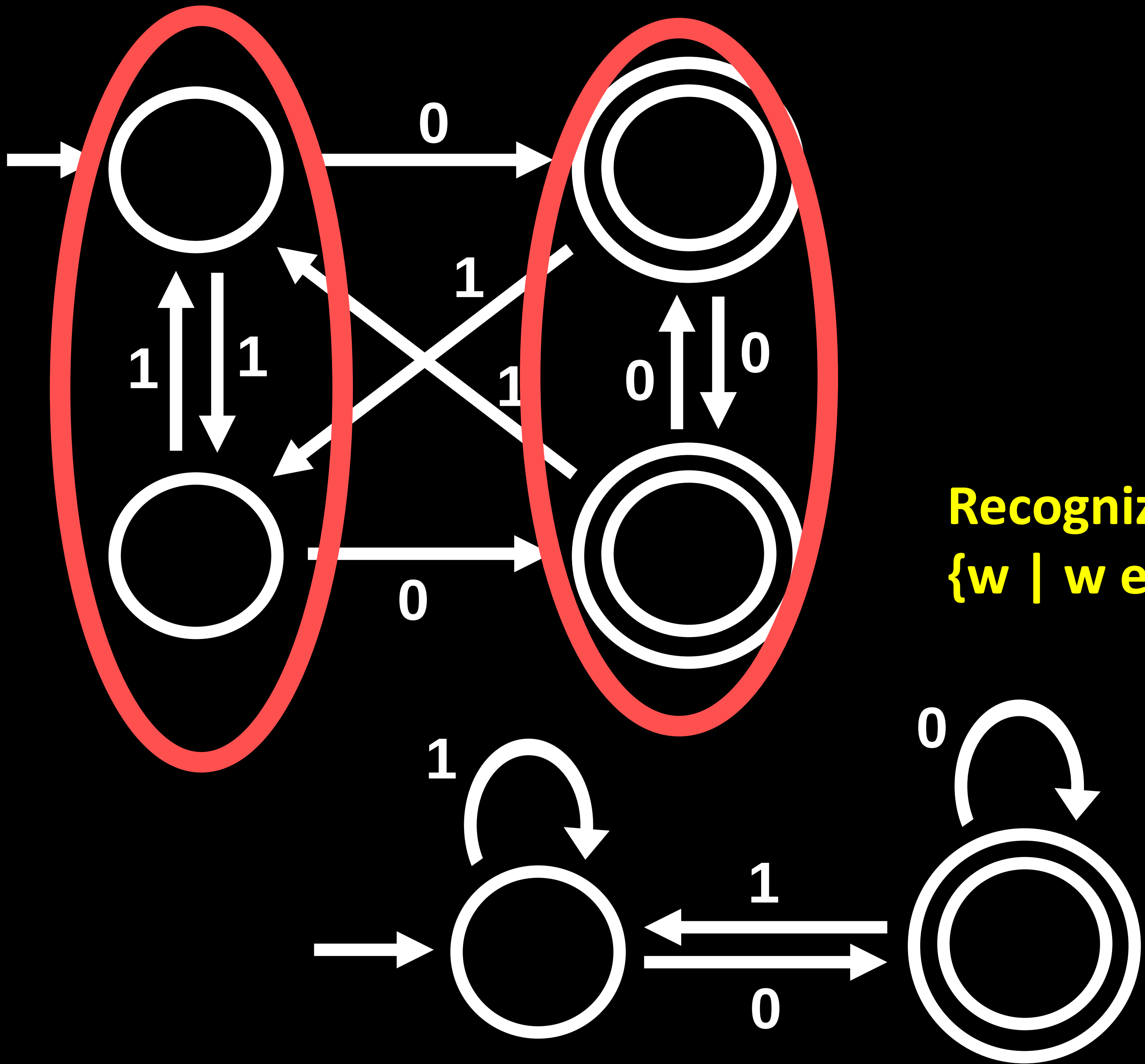
Minimizing DFAs





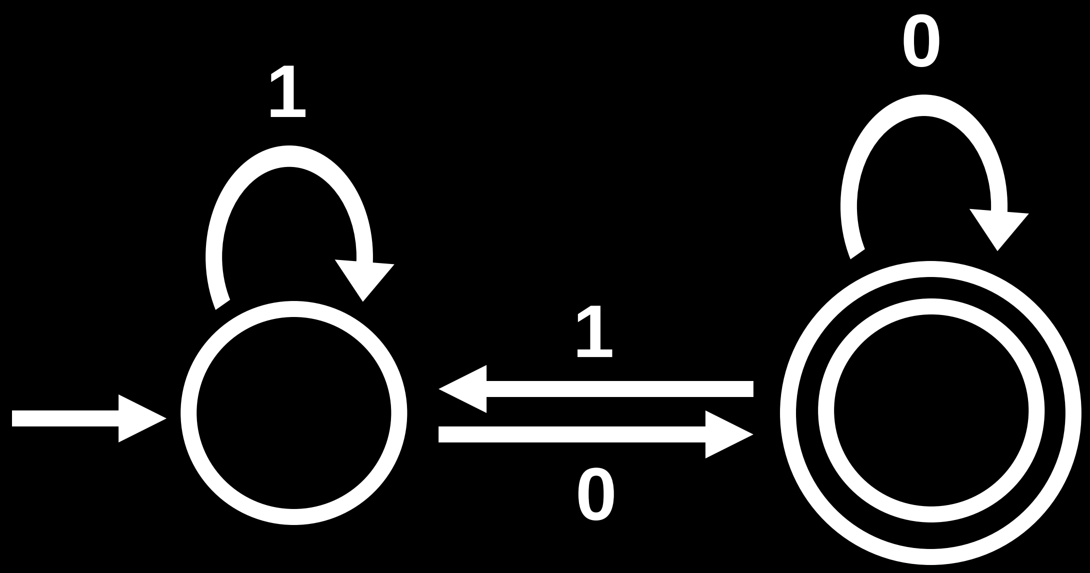
Does this DFA have a minimal number of states?





Recognizes
{w | w ends in 0}

Is this minimal?



How can we tell in general?

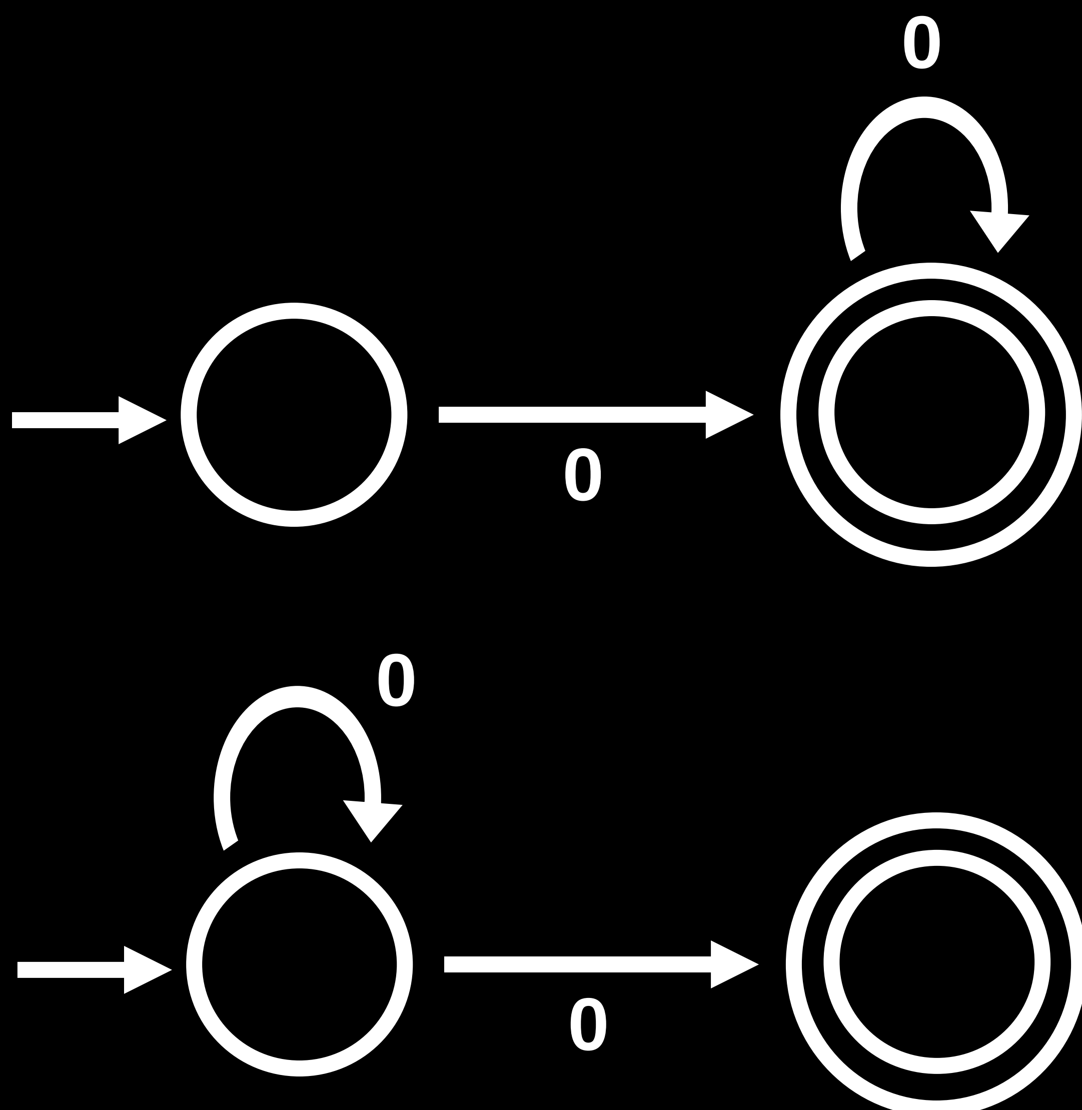
DFA Minimization Theorem:

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

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

If such algorithms existed for more general models of computation, that would be an engineering breakthrough!!

In general, there isn't a uniquely minimal NFA

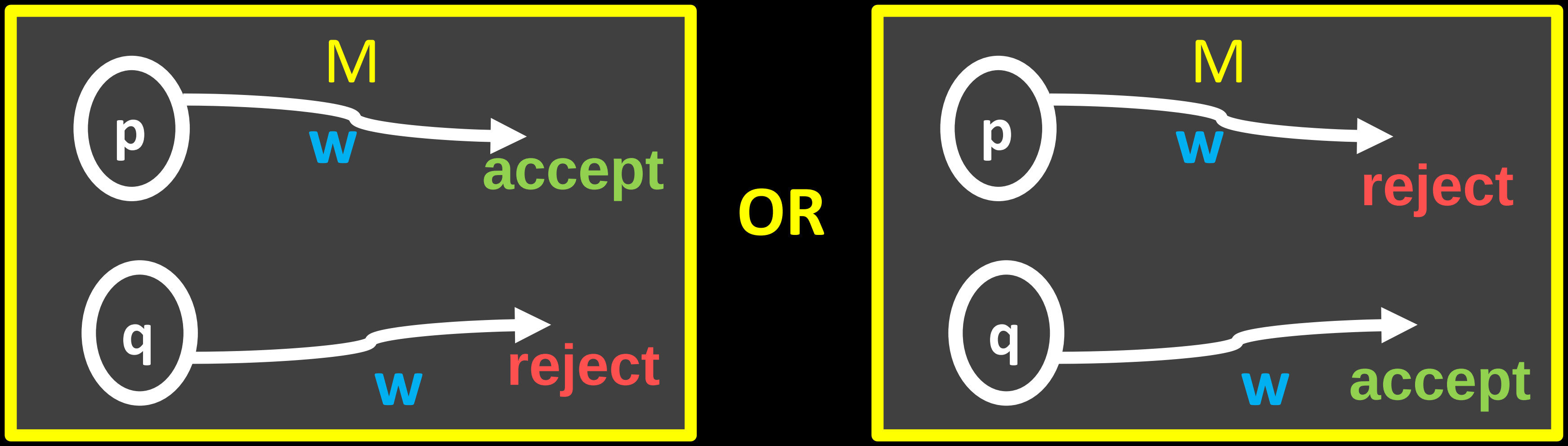


Distinguishing states with strings

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and $q \in Q$, let M_q be the DFA equal to $(Q, \Sigma, \delta, q, F)$

Def. $w \in \Sigma^*$ ***distinguishes*** states p and q if:

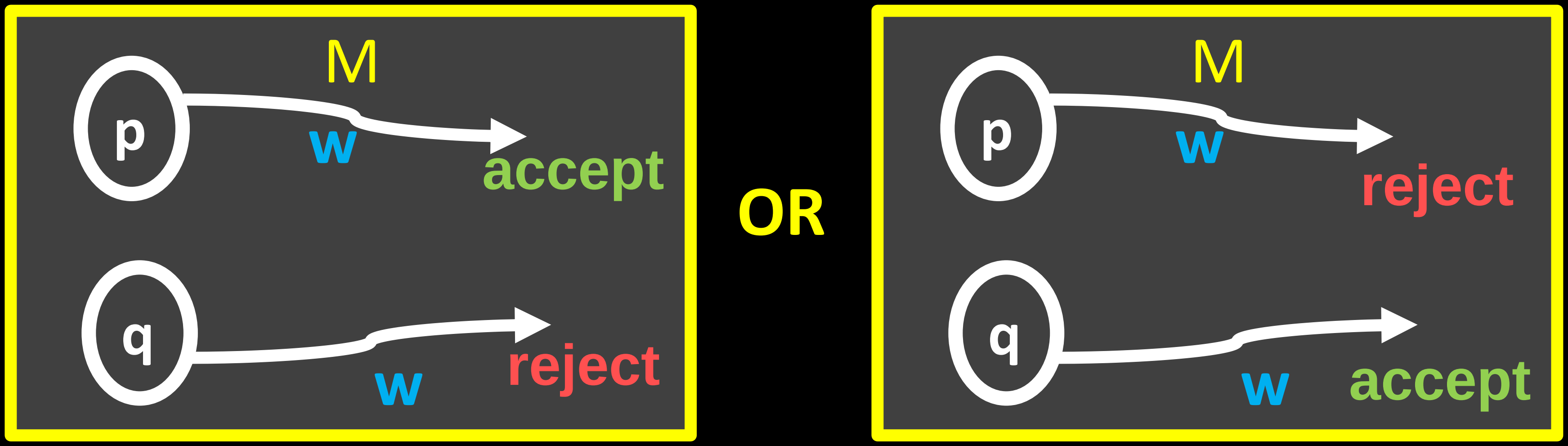
M_p accepts $w \Leftrightarrow M_q$ rejects w



Distinguishing states with strings

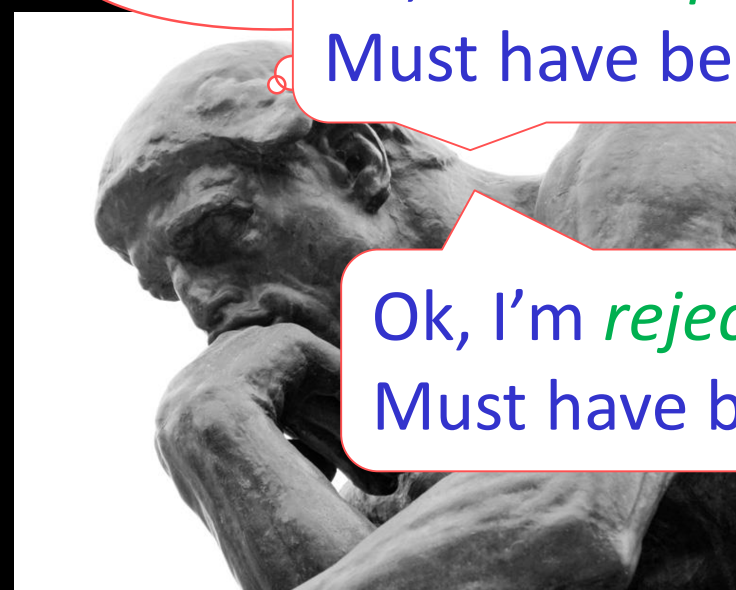
For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and $q \in Q$,
let M_q be the DFA equal to $(Q, \Sigma, \delta, q, F)$

Def. $w \in \Sigma^*$ *distinguishes* states p and q if:
M_p and M_q have *different outputs* on input w



Distinguishing two states

Def. w $\in \Sigma^*$ *distinguishes* states p and q iff M_p and M_q have *different outputs* on w



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

Let $M_p = (Q, \Sigma, \delta, p, F)$ and $M_q = (Q, \Sigma, \delta, q, F)$

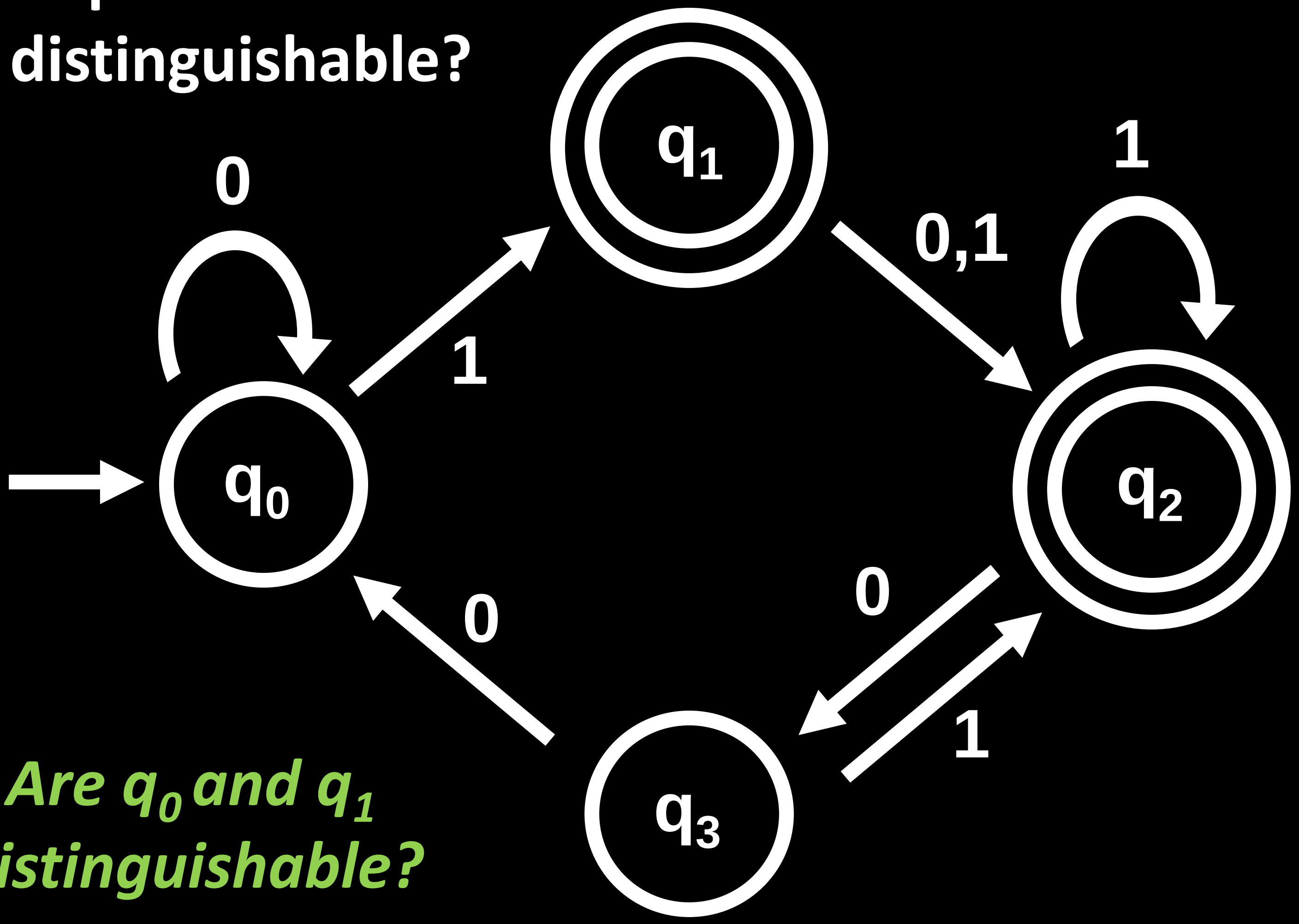
Definition(s):

State p is *distinguishable* from state q
iff there is a $w \in \Sigma^*$ that distinguishes p and q
iff there is a $w \in \Sigma^*$ so that
M_p accepts w $\Leftrightarrow$ M_q rejects w

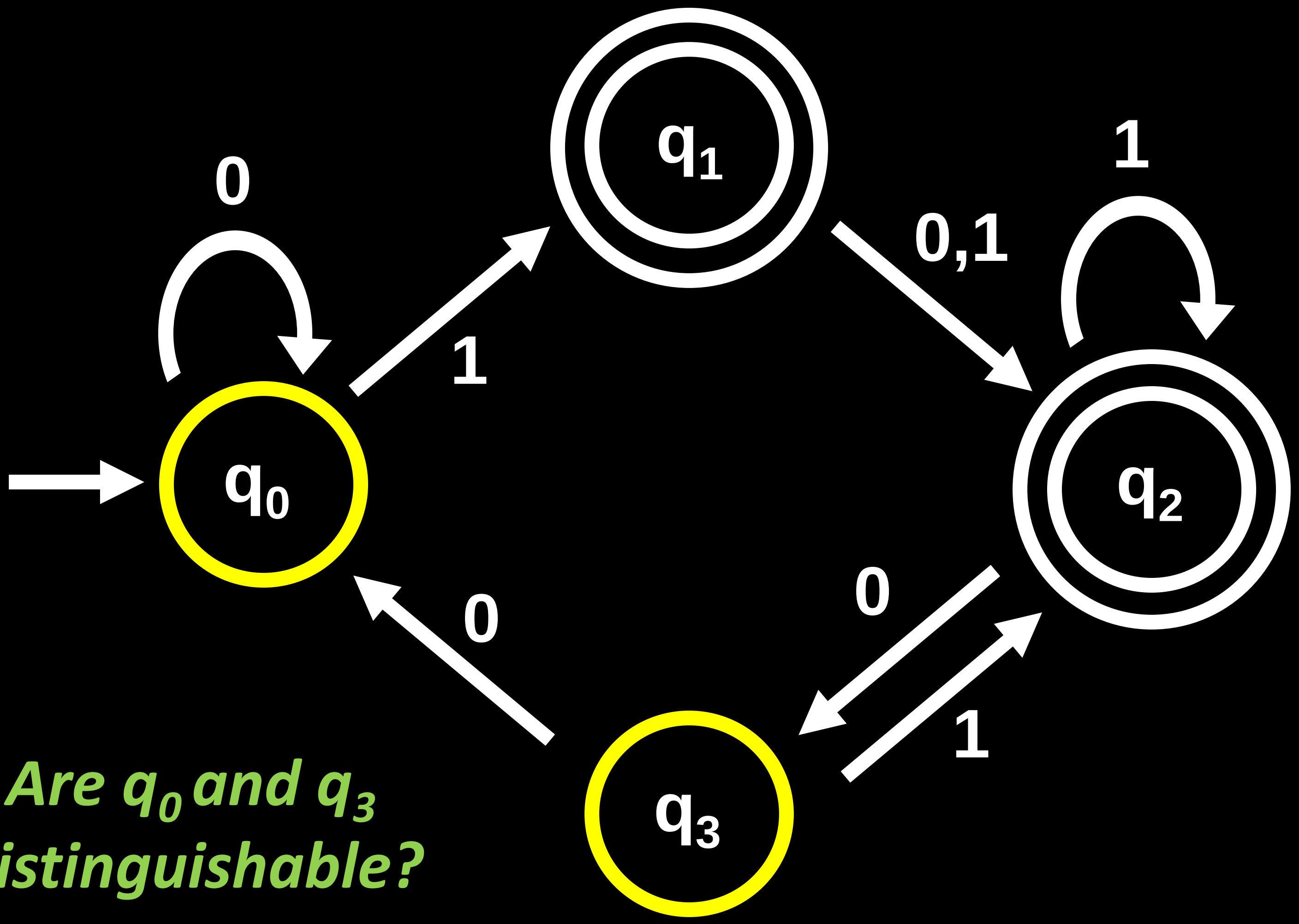
State p is *indistinguishable* from state q
iff p is not distinguishable from q
iff for all $w \in \Sigma^*$, M_p accepts w $\Leftrightarrow$ M_q accepts w

Big Idea: Pairs of indistinguishable states are redundant!
From p or q, M has exactly the same output behavior

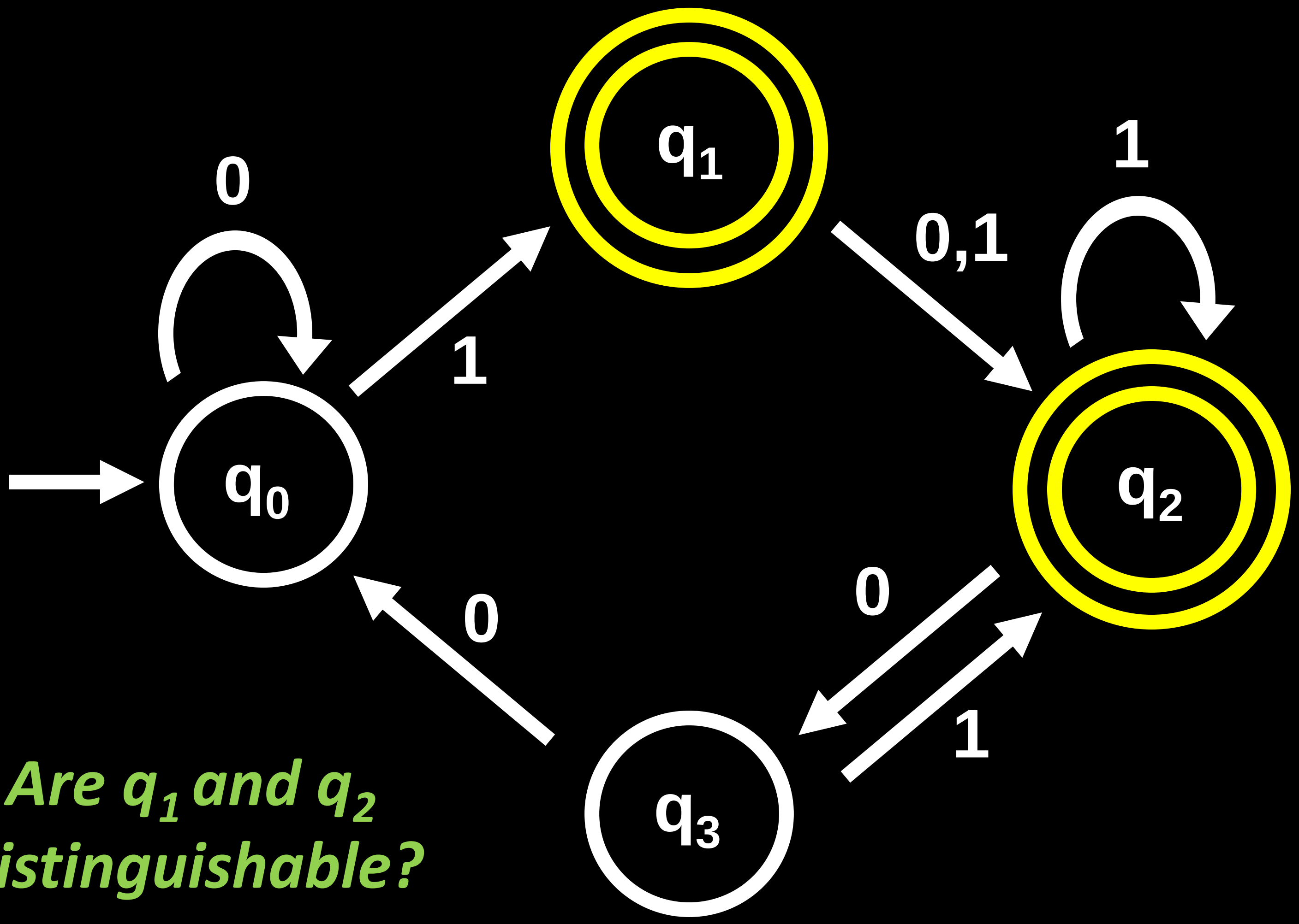
Which pairs of states are distinguishable?



Are q_0 and q_1 distinguishable?



Are q_0 and q_3 distinguishable?



Are q_1 and q_2 distinguishable?

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)

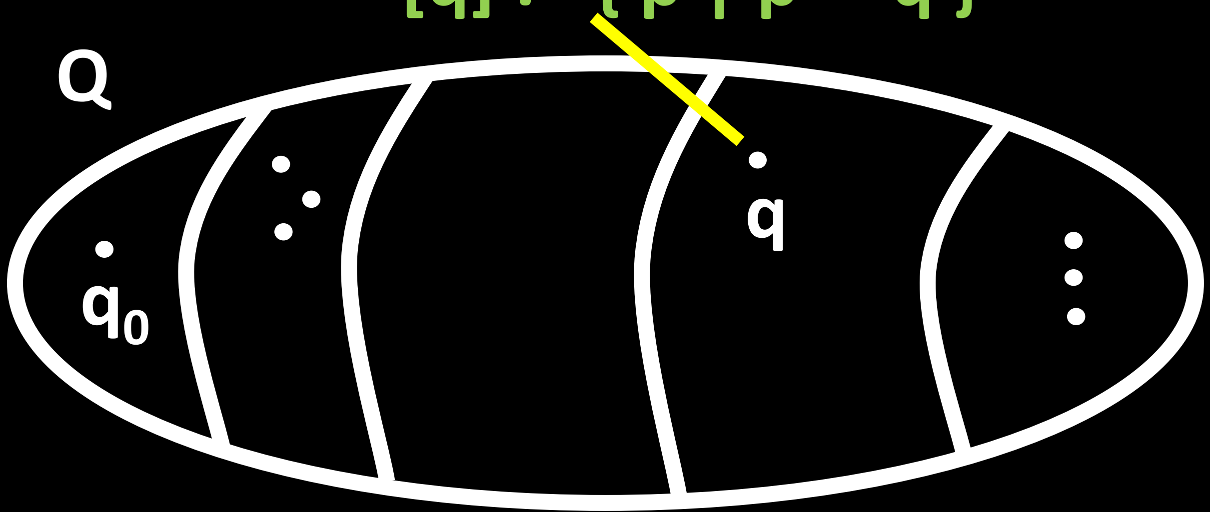
Proof? Just look at the definition! $p \sim q$ means for all w, M_p accepts w $\Leftrightarrow$ M_q accepts w

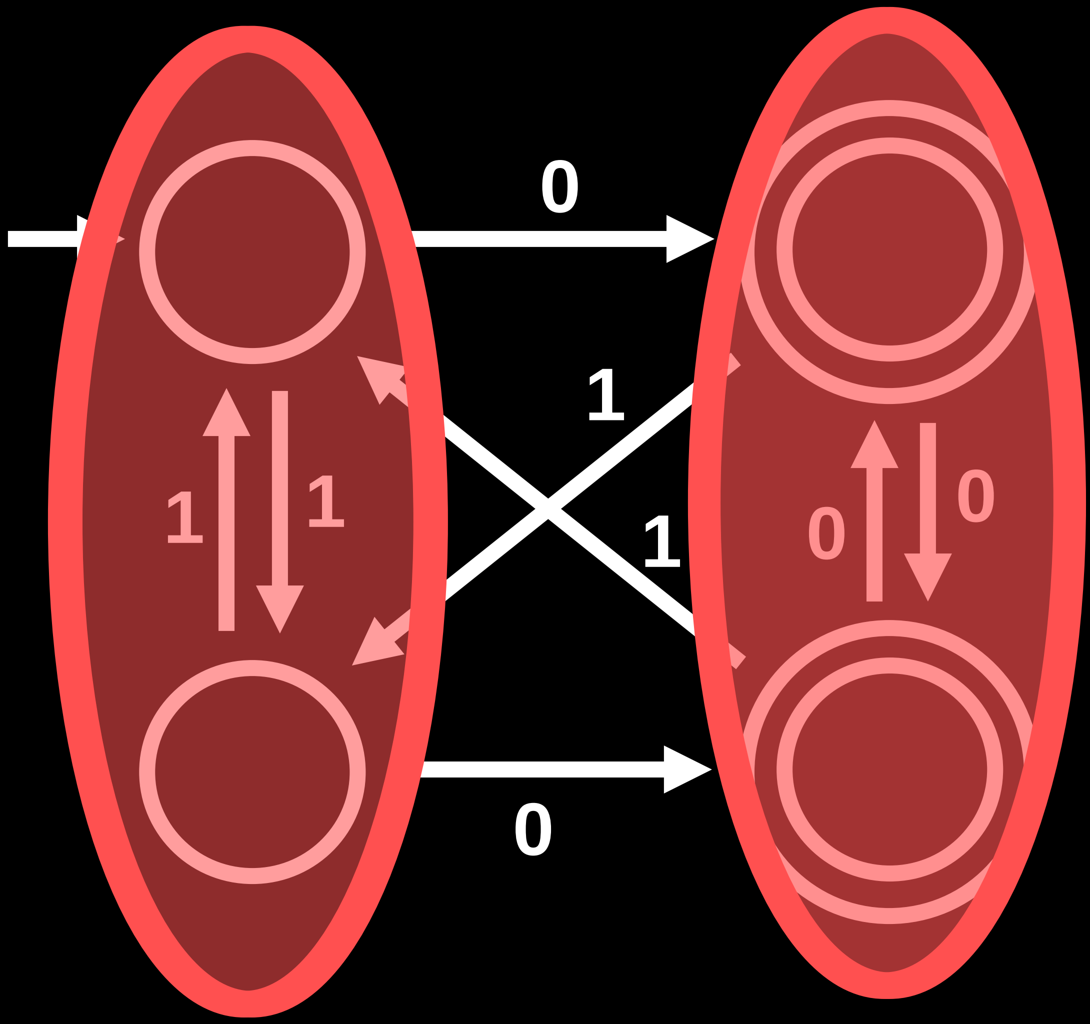
Fix $M = (Q, \Sigma, \delta, q_0, F)$ and let $p, q, r \in Q$

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

Proposition: $\sim$ is an equivalence relation

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




0
1
1
1
1
0
0
0

Algorithm: **MINIMIZE-DFA**

Input: DFA M

Output: DFA M_{MIN} such that:

unreachable from start state

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

2. M_{MIN} has no *inaccessible* states

3. M_{MIN} is *irreducible*

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

Theorem: Every M_{MIN} satisfying 1,2,3 is the unique minimal DFA equivalent to M

Intuition:

States of M_{MIN} = *Equivalence classes* of states of M

We'll discover the equivalent states with a *dynamic programming* algorithm

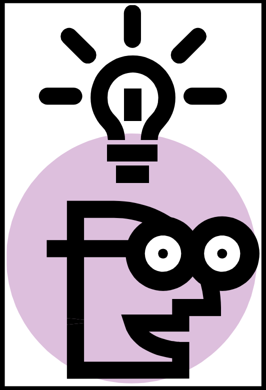
The Table-Filling Algorithm

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

Idea:



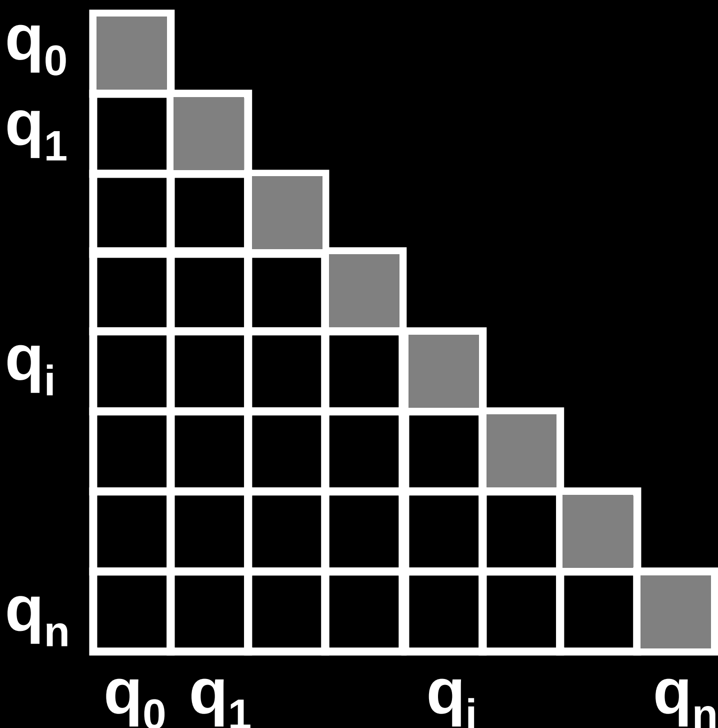
- We know how to find those pairs of states that the string ε distinguishes…
- Use this and *iteration* to find those pairs distinguishable with *longer* strings
- The pairs of states left over will be indistinguishable

The Table-Filling Algorithm

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

Output: (1) $D_M = \{ (p, q) \mid p, q \in Q \text{ and } p \not\sim q \}$

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



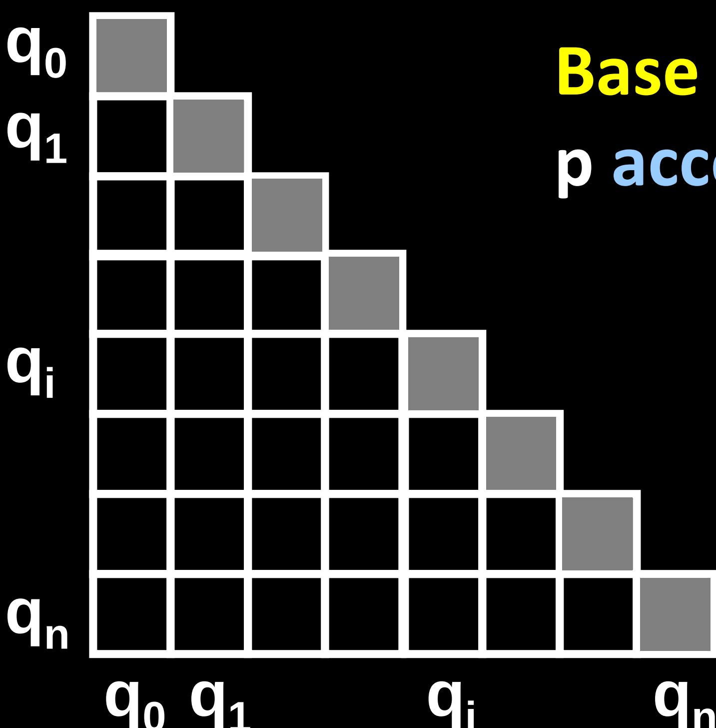
Suppose $|Q|=n+1$.
Start by making a table of cells, with ½ of all possible state pairs.
We want to fill in which pairs are distinguishable.

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



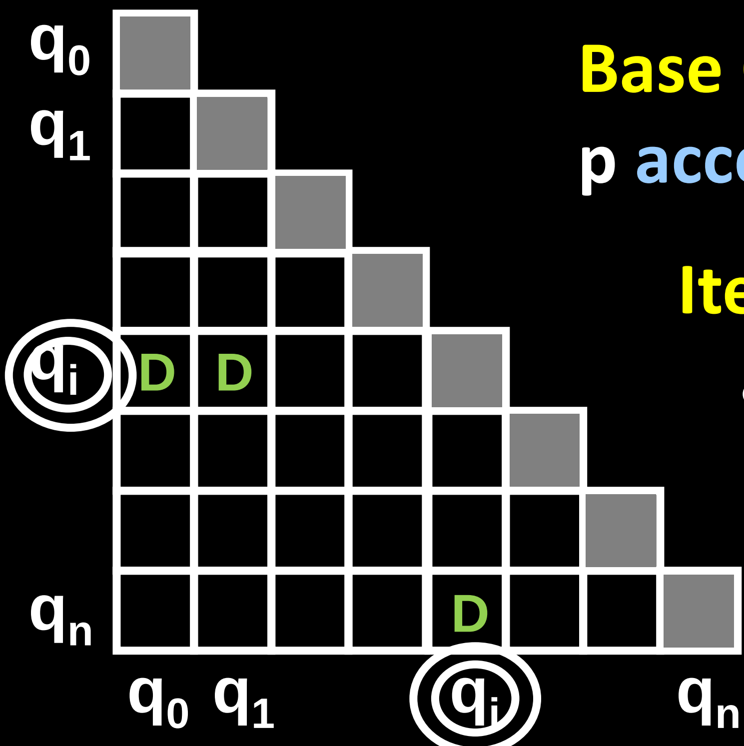
Base Case: For all (p, q) such that p accepts and q rejects $\Rightarrow$ mark $p \not\sim q$

The Table-Filling Algorithm

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

Output: (1) $D_M = \{ (p, q) \mid p, q \in Q \text{ and } p \not\sim q \}$

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

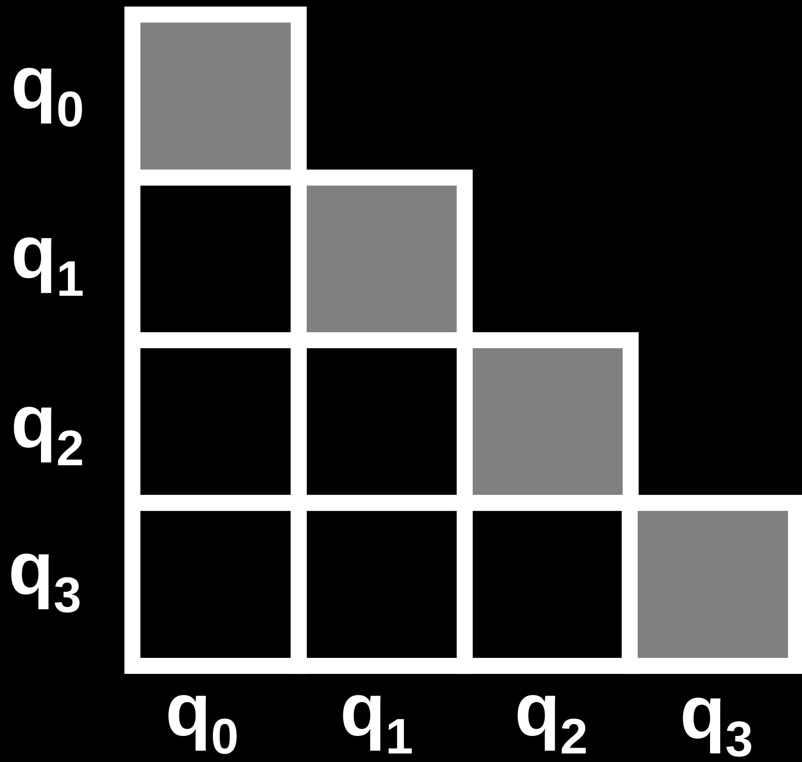


Base Case: For all (p, q) such that p **accepts** and q **rejects** $\Rightarrow$ mark $p \not\sim q$

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

$$\begin{matrix} \delta(p, \sigma) = p' \\ \not\sim \\ \delta(q, \sigma) = q' \end{matrix} \quad \Rightarrow \quad \text{mark } p \not\sim q$$

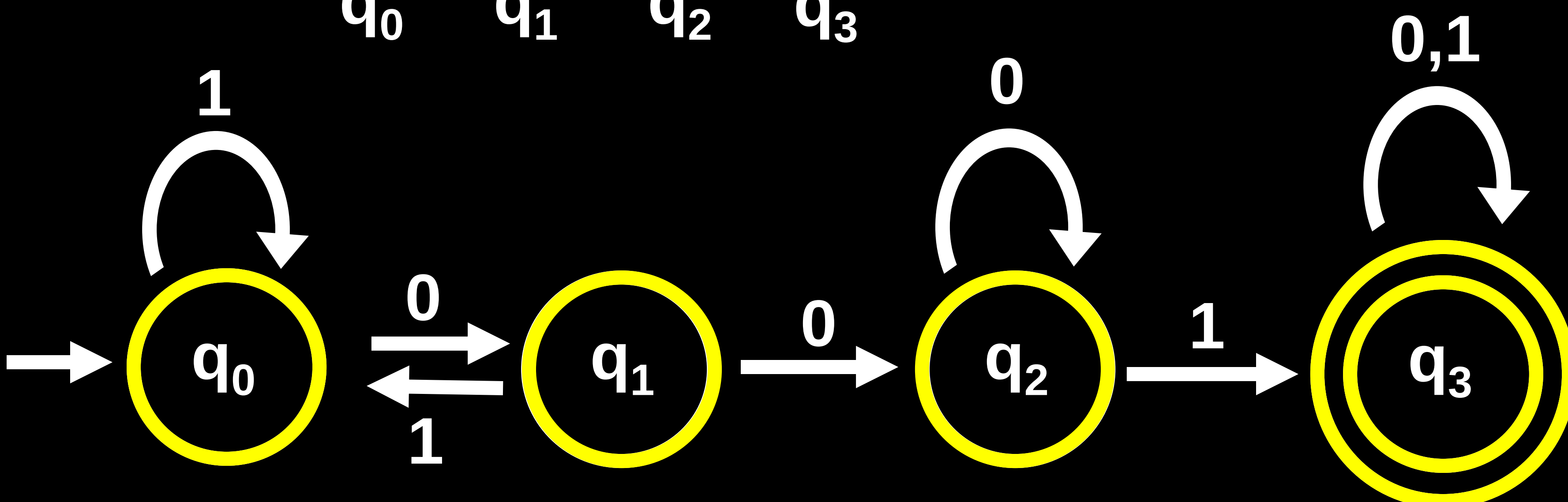
Repeat until the rule doesn't apply

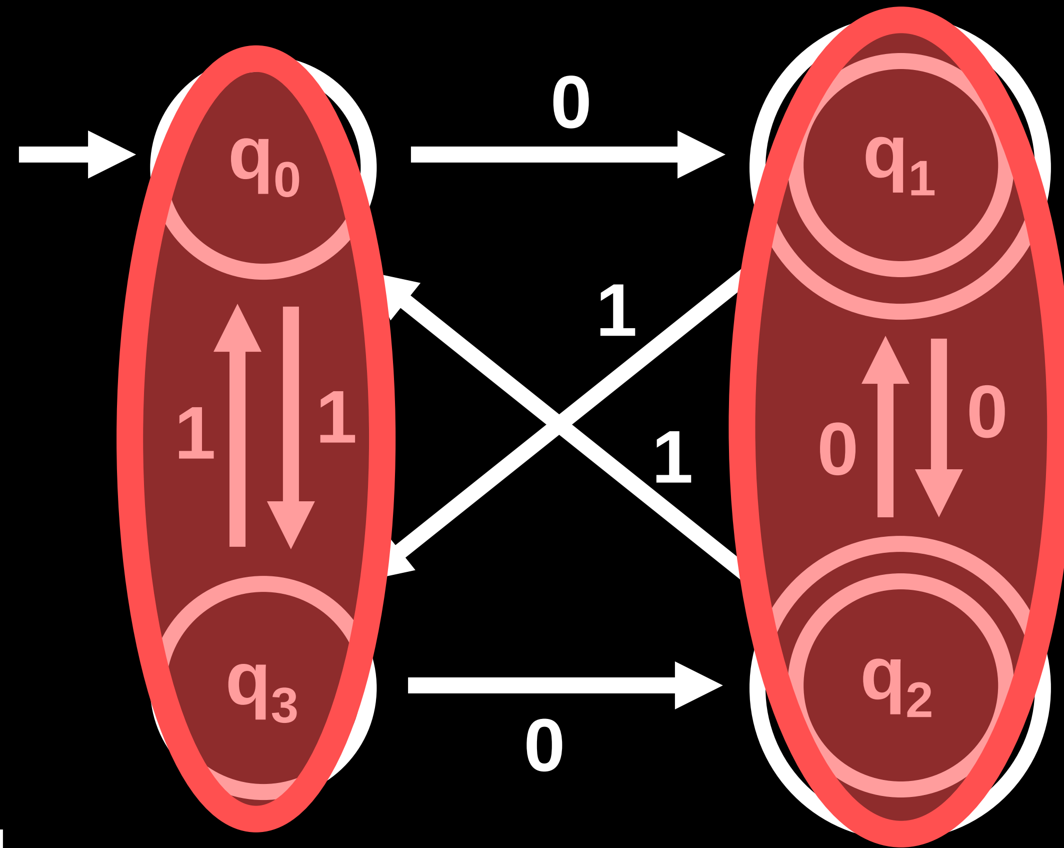


Are q_1 and q_2 distinguishable?

Are q_0 and q_1 distinguishable?

Are q_0 and q_2 distinguishable?





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

Recognizes
{w | w ends in 0}

Claim: If (p, q) is marked D by the algorithm, then p $\not\sim$ q

Proof: Induction on the number of iterations n in the algorithm when (p, q) is marked D

$n = 0$: If (p, q) is marked D in the base case, then exactly one of them is final, so ε distinguishes p and q

I.H. For all (p', q') marked D in the first n iterations, p′ $\not\sim$ q′

Suppose (p, q) is marked D in the $(n+1)$th iteration.

To be marked, there must be states p′, q′ such that:

1. p′ = δ(p,σ) and q′ = δ(q,σ), for some $\sigma \in \Sigma$
2. (p′, q′) is marked D $\Rightarrow$ p′ $\not\sim$ q′ (by induction)

So there's a w s.t. w distinguishes p' and q'

Then, the string σw distinguishes p and q!

Claim: If (p, q) is **not marked D** by the algorithm, then $p \sim q$

Proof (by contradiction):

Suppose there is a pair (p, q) 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:

M_p and M_q have different outputs on w (Why is $|w| > 0$?)

Of all such bad pairs, let (p, q) be a pair with a *minimum-length* distinguishing string w

Claim: If (p, q) is **not marked D** by the algorithm, then $p \sim q$

Proof (by contradiction):

Suppose there is a pair (p, q) 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 a *minimum-length* distinguishing string w

M_p and M_q have different outputs on w (Why is $|w| > 0$?)

We have $w = \sigma w'$, for some string w' and some $\sigma \in \Sigma$

Let $p' = \delta(p,\sigma)$ and $q' = \delta(q,\sigma)$. (p',q') distinguished by w'

Then (p', q') is also a bad pair! (It must be not marked D)

But then (p', q') has a SHORTER distinguishing string, w'

Contradiction!

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

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

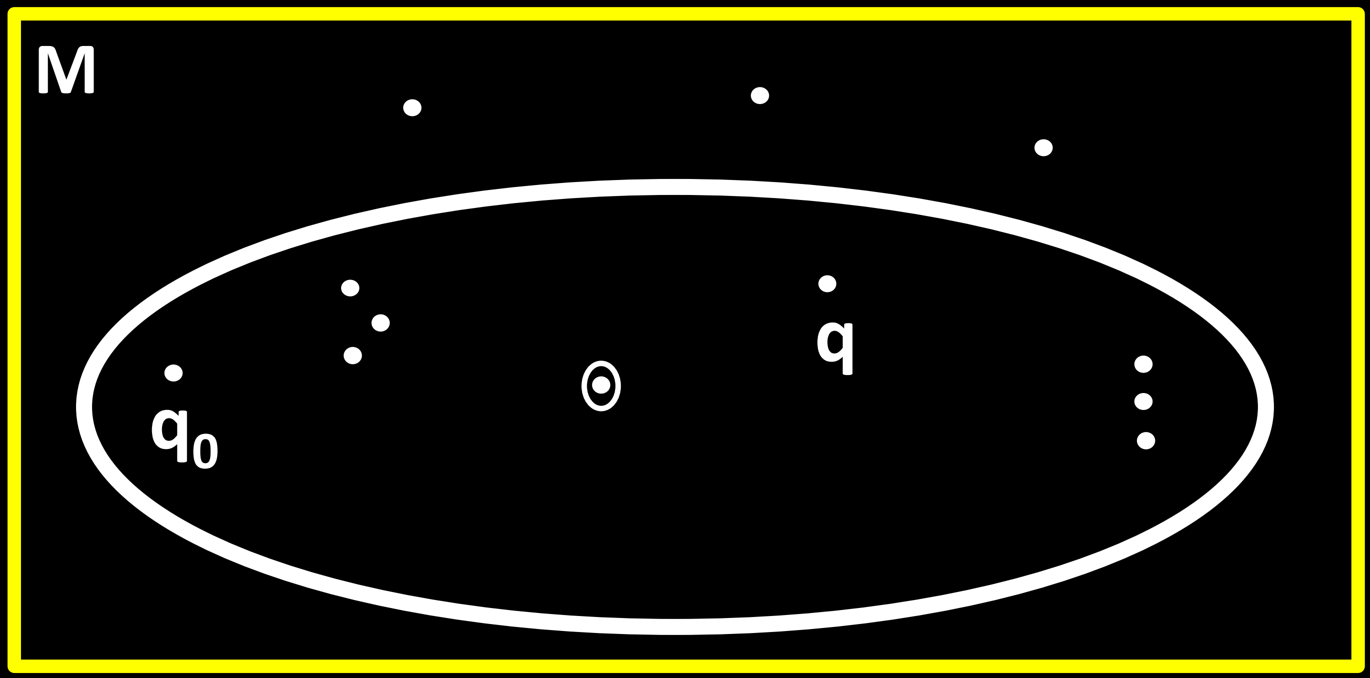
$\delta_{MIN}([q], \sigma) = [\delta(q, \sigma)]$

(well-defined??)

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

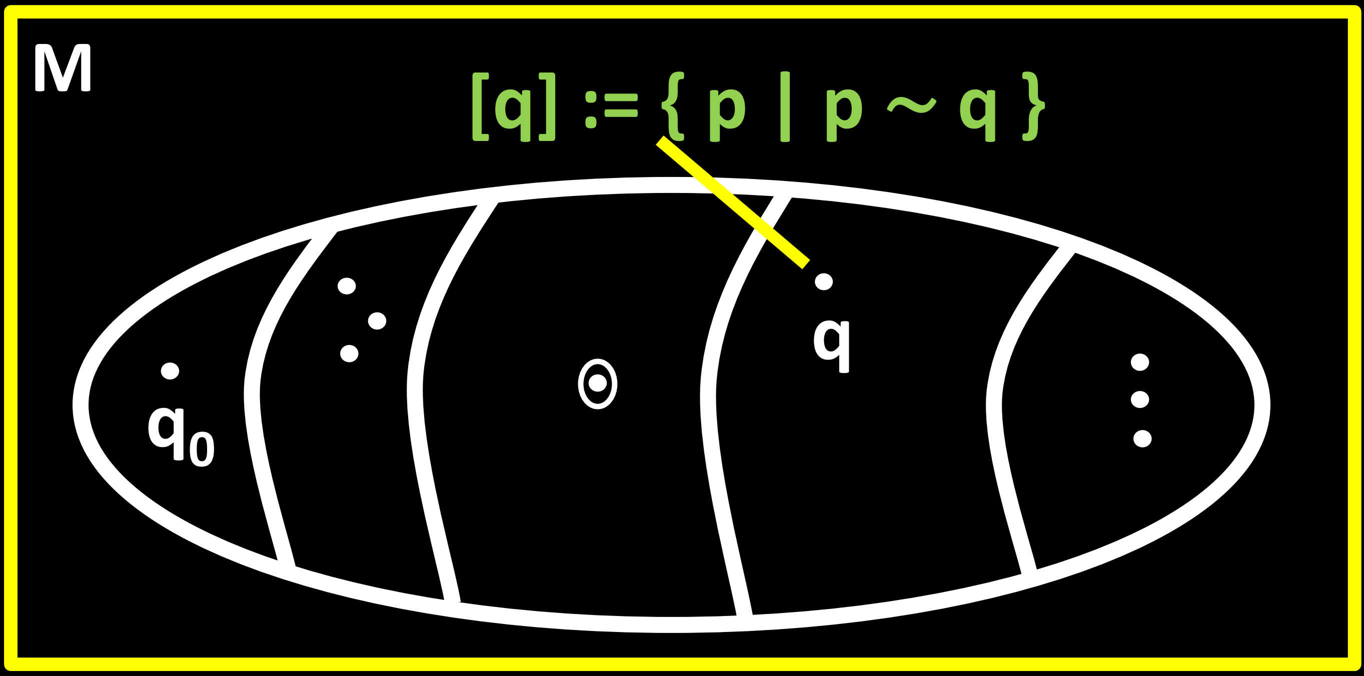
The MINIMIZE Algorithm in Pictures

1. Remove all inaccessible states



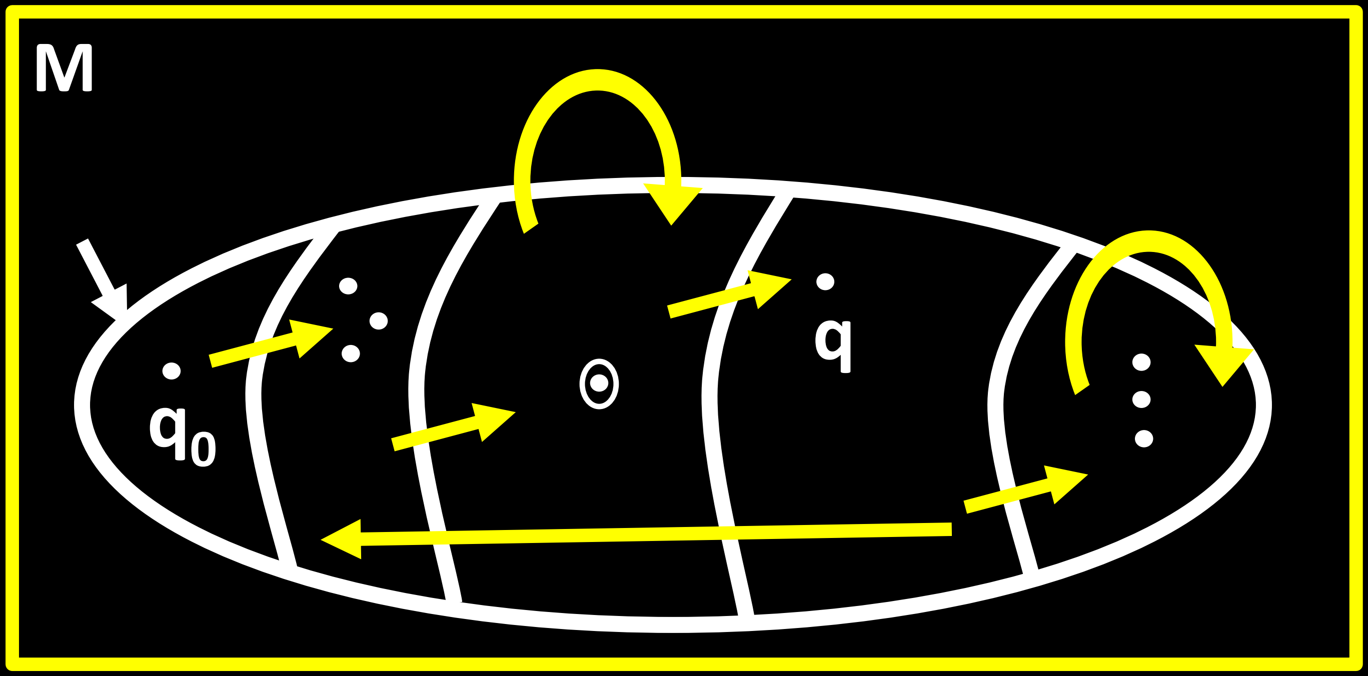
The MINIMIZE Algorithm in Pictures

2. Run Table-Filling to get equiv classes



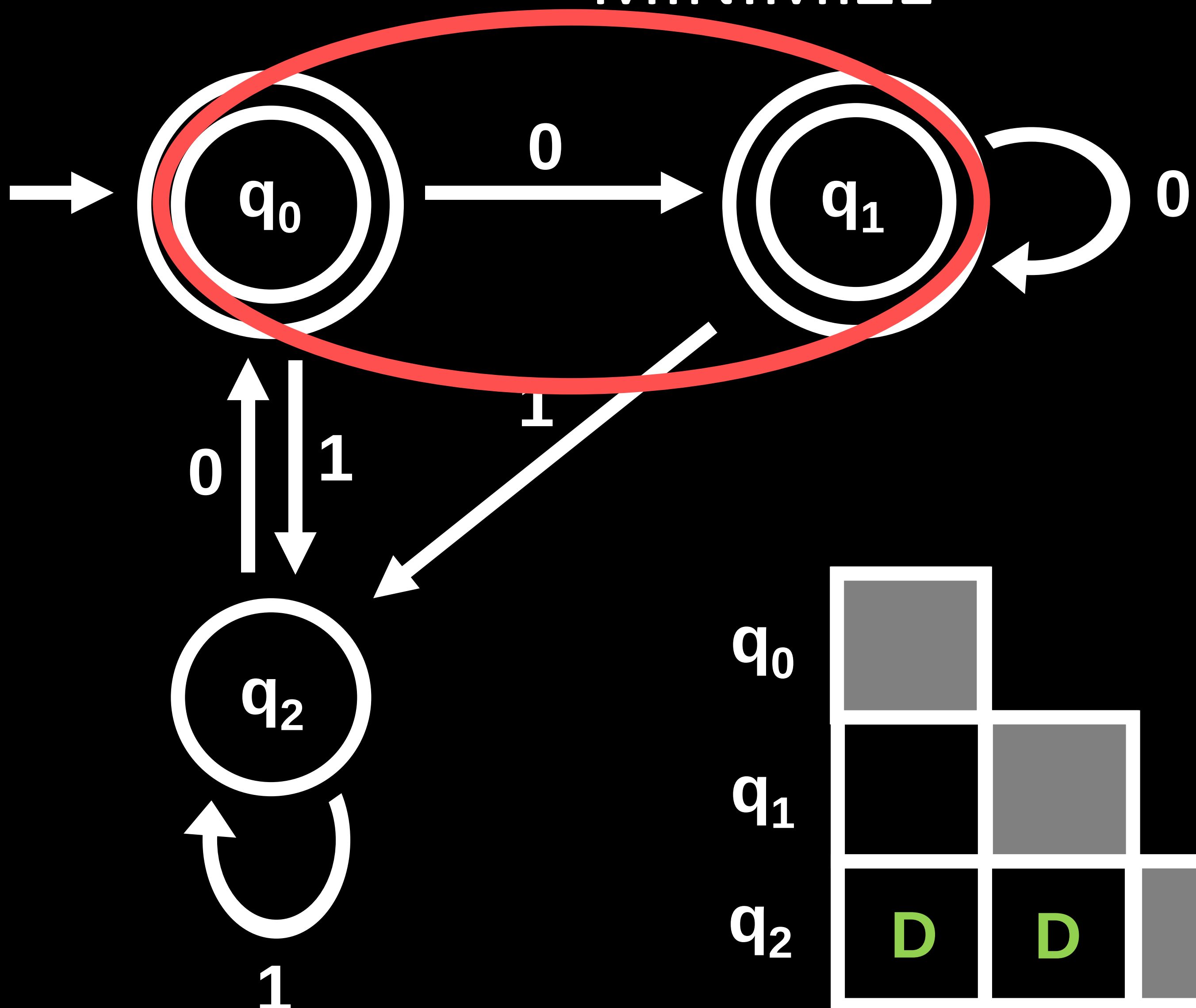
The MINIMIZE Algorithm in Pictures

🙌 **3. Define M_{MIN} with states = equiv classes**



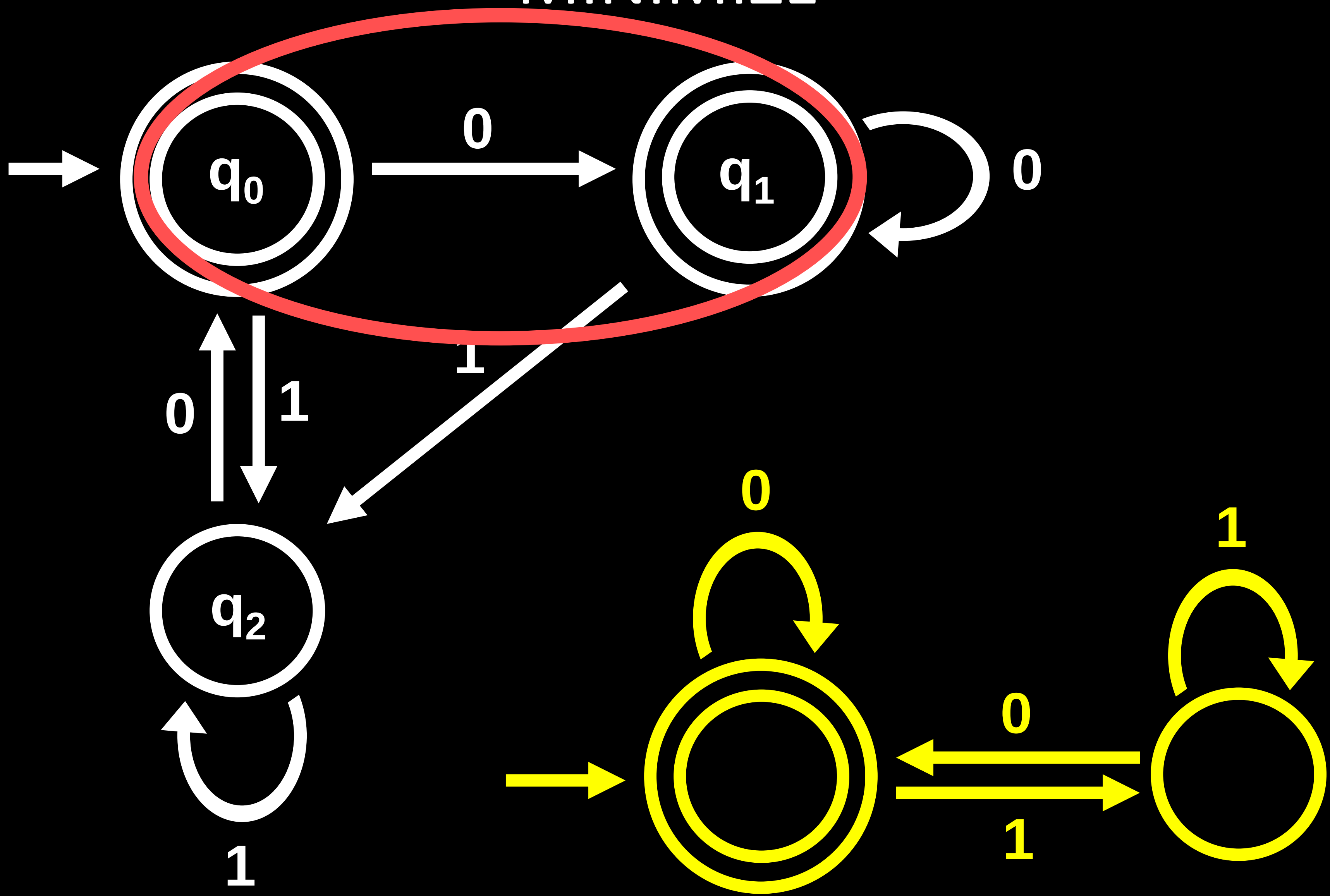
States of M_{MIN} = *Equivalence classes* of states of M

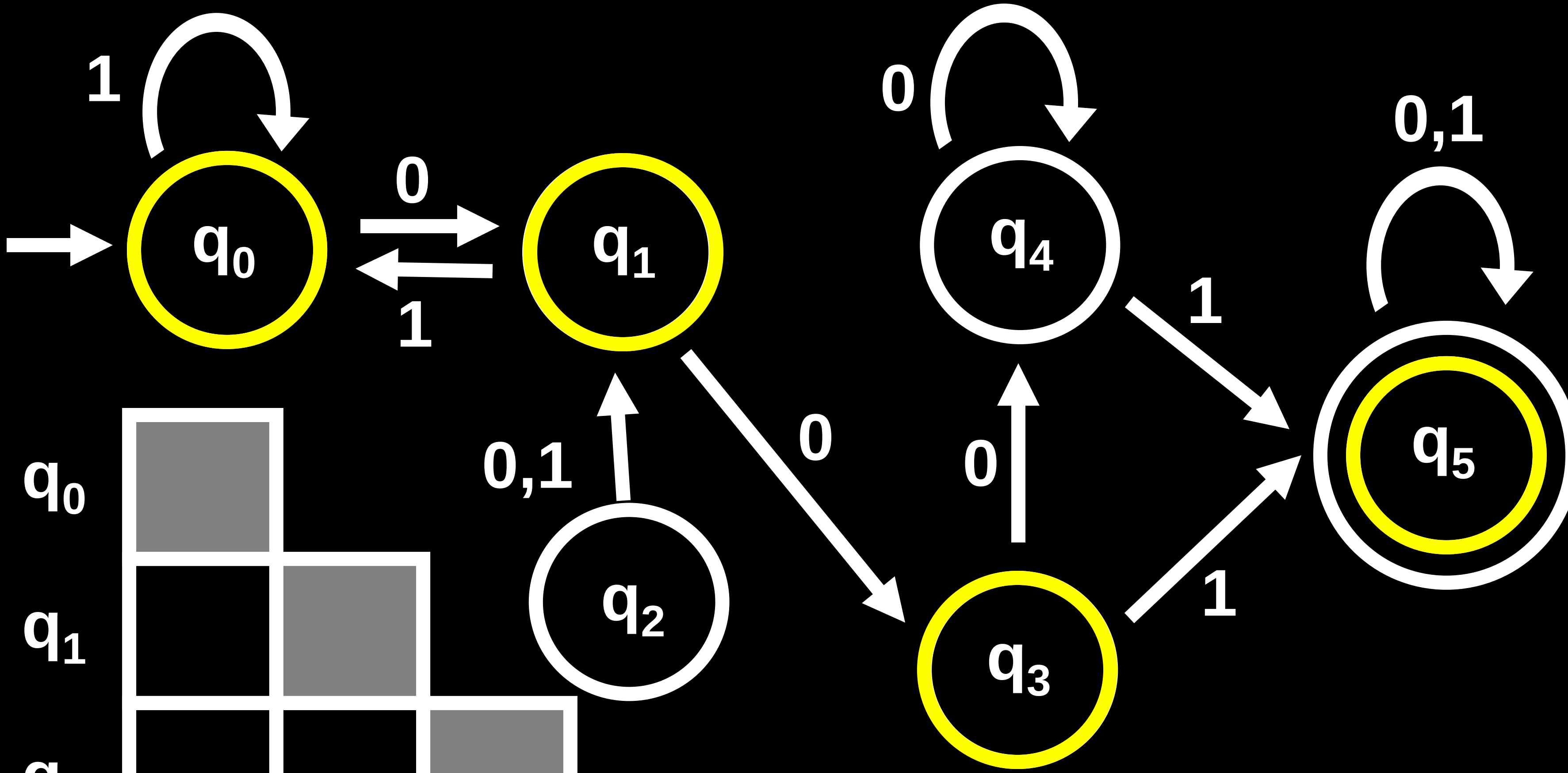
MINIMIZE

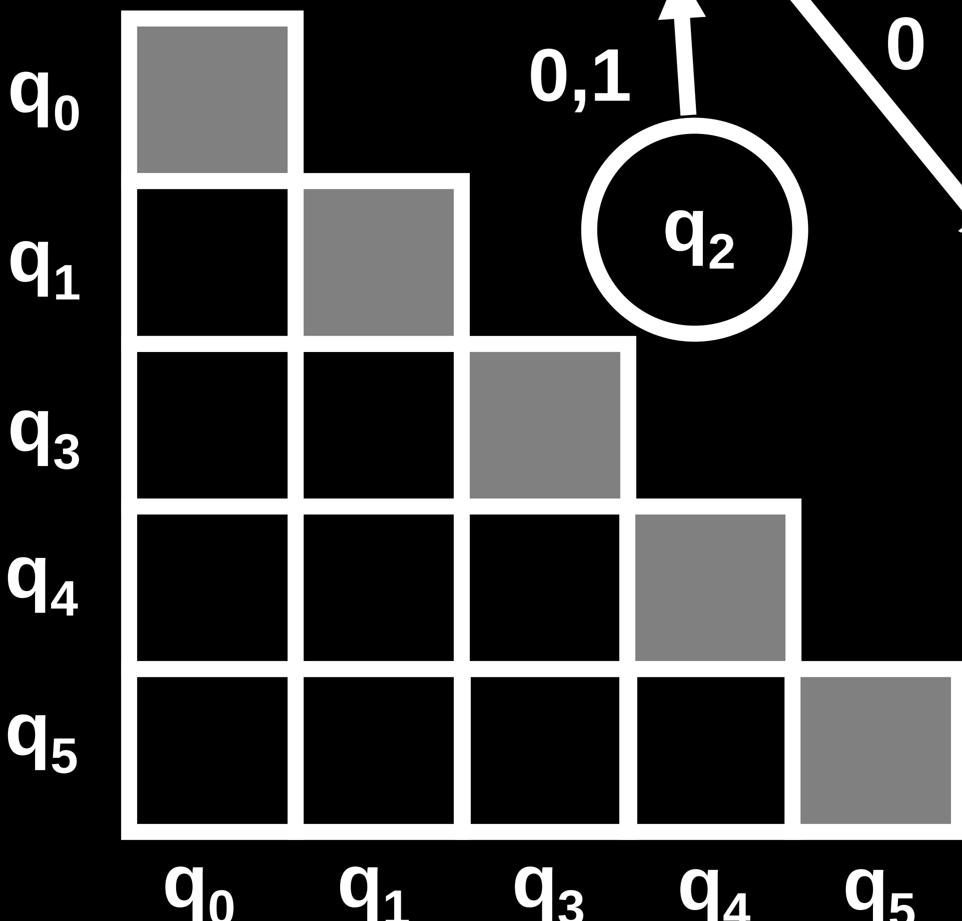


q_0			
q_1			
q_2	D	D	
	q_0	q_1	q_2	q_3

MINIMIZE







Distinguish q_1 from q_3 and q_4

Distinguish q_0 from q_3 and q_4

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

Claim: Let M′ be any DFA where 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}

Suppose we have proved the Claim is true.
Assuming the Claim we can prove the Thm:

Proof of Thm: Let M′ be any minimal DFA for M.
Since M′ is minimal, M′ has no inaccessible states and is irreducible *(otherwise, we could make M' smaller!)*
By the Claim, there is an isomorphism between M' and the DFA M_{MIN} that is output by MINIMIZE(M).
That is, M_{MIN} is isomorphic to every minimal M'.

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

Claim: Let M′ be any DFA where 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}

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 $p \mapsto p'$

$$\begin{array}{ccc} p & \mapsto & p' \\ \downarrow{\scriptstyle\sigma} & & \downarrow{\scriptstyle\sigma} \\ q & & q' \end{array}$$

Then $q \mapsto q'$

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

Recursive Step: If $p \mapsto p'$ Then $q \mapsto q'$

$$
\begin{tikzcd}
p \arrow[d, "\sigma"] & p' \arrow[d, "\sigma"] \\
q & q'
\end{tikzcd}
$$

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

Recursive Step: If

$$\begin{tikzcd} p \arrow[r, mapsto] \arrow[d, "\sigma"] & p' \arrow[d, "\sigma"] \\ q & q' \end{tikzcd}$$

Then $q \mapsto q'$

Claim: Map is an isomorphism. Need to prove:

The map is **defined** everywhere

The map is **well defined**

The map is a **bijection (one-to-one and onto)**

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 $p \mapsto p'$

$$\begin{array}{ccc} p & & p' \\ \downarrow \sigma & & \downarrow \sigma \\ q & & q' \end{array}$$

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
M_{MIN} is in state q after reading in w

Let q′ be the state of M’ after reading in w.
Claim: $q \mapsto q'$ *(proof by induction on |w|)*

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

Recursive Step: If

$$\begin{tikzcd} p \arrow[r, mapsto] \arrow[d, "\sigma"] & p' \arrow[d, "\sigma"] \\ q & q' \end{tikzcd}$$

Then $q \mapsto q'$

The map is **onto:** $\forall q' \; \exists \; q$ such that $q \mapsto q'$

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

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

Let q be the state of M_{MIN} after reading in w.

Claim: $q \mapsto q'$ *(proof by induction on $|w|$)*

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

Recursive Step: If

$$\begin{tikzcd} p \arrow[r, mapsto] \arrow[d, "\sigma"] & p' \arrow[d, "\sigma"] \\ q & q' \end{tikzcd}$$

Then $q \mapsto q'$

The map is well defined: $\forall q\ \exists!\ q'$ such that $q \mapsto q'$

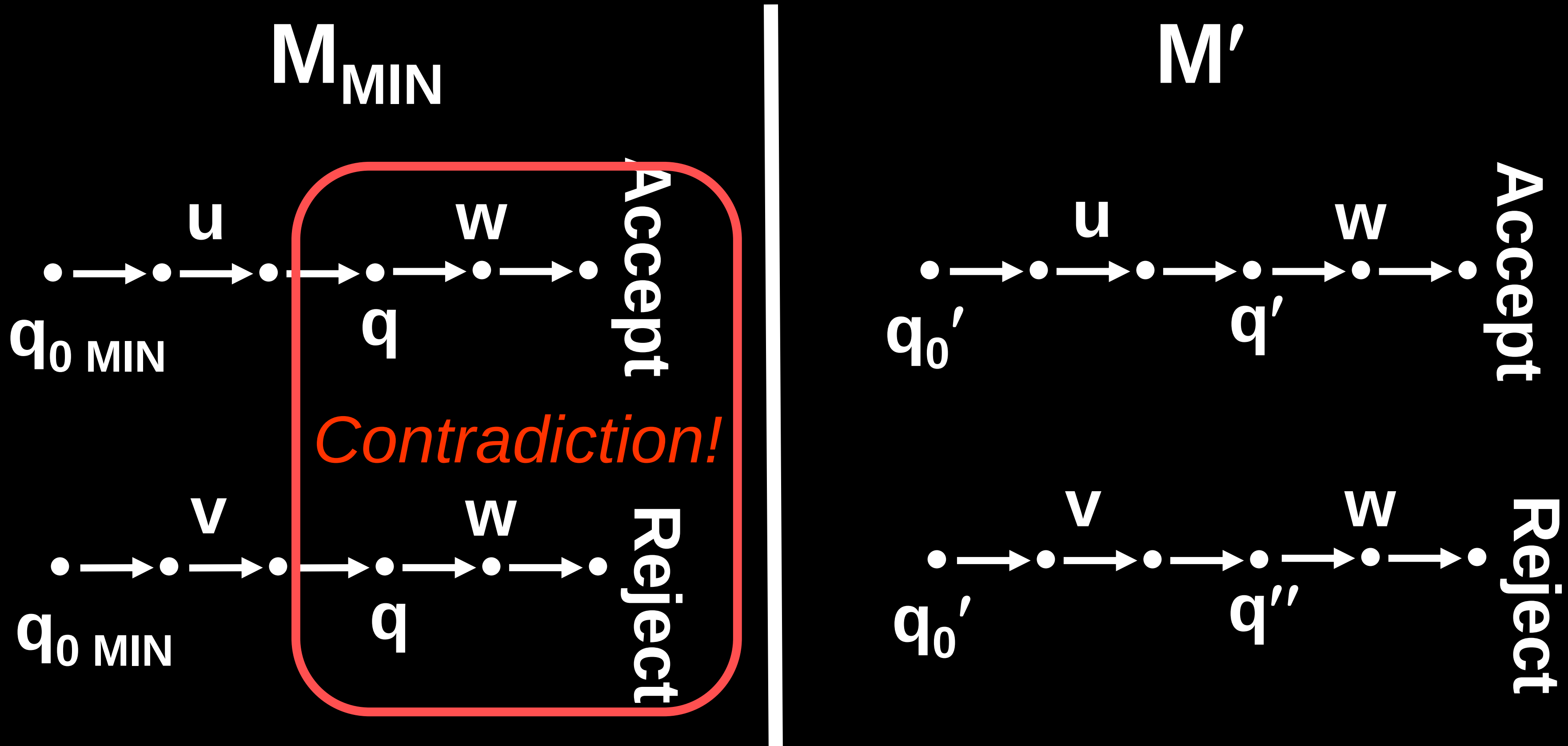
Suppose there are states q' and q'' such that $q \mapsto q'$ and $q \mapsto q''$

We show that q' and q'' are *indistinguishable*, so it must be that $q' = q''$ *(why?)*

Suppose there are states q′ and q′′ such that

$q \mapsto q'$ and $q \mapsto q''$

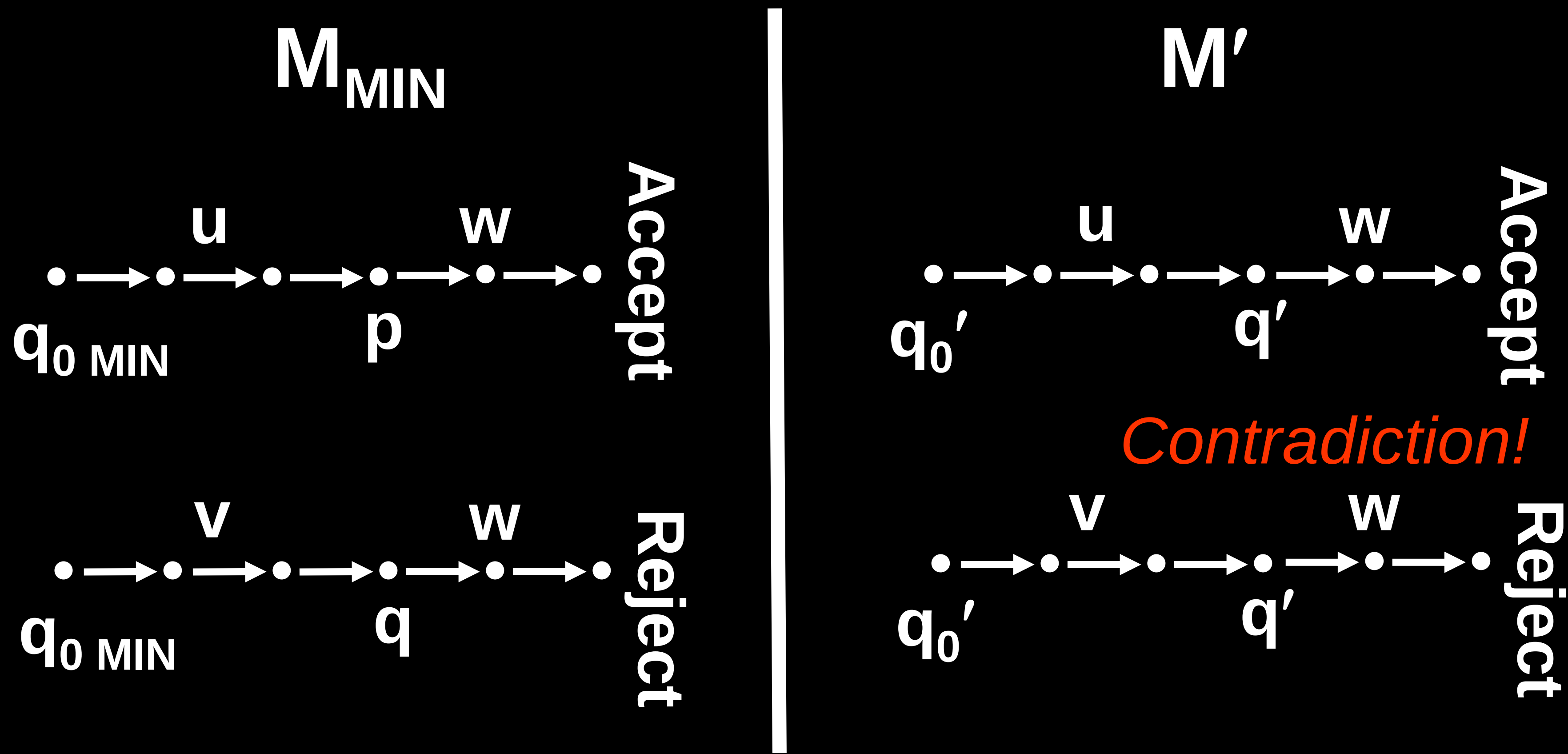
Assume for contradiction q′ and q′′ are distinguishable



Map is 1-to-1: $\forall p \neq q,\ p \mapsto q'$ and $q \mapsto q'' \Rightarrow q' \neq q''$

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?