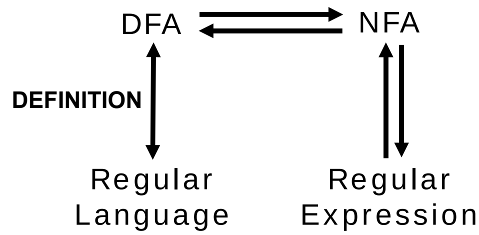
15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY



How can we prove that two regular expressions are equivalent?

How can we prove that two DFAs (or two NFAs) are equivalent?

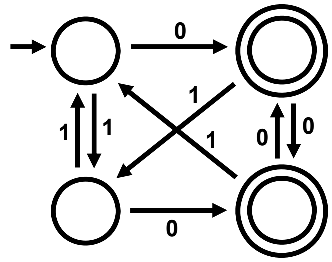
How can we prove that two regular languages are equivalent? (Does this question make sense?)

How can we prove that two DFAs (or two NFAs) are equivalent?

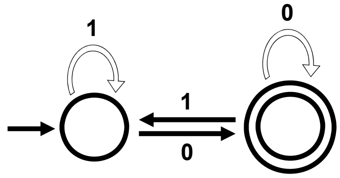
MINIMIZING DFAs

THURSDAY Jan 23

IS THIS MINIMAL?



IS THIS MINIMAL?



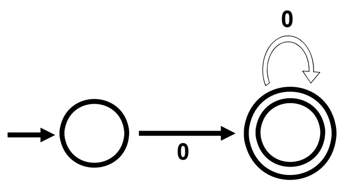
THEOREM

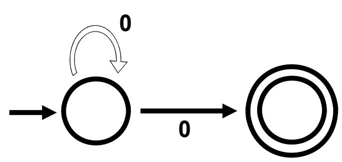
For every regular language L, there exists a UNIQUE (up to re-labeling of the states) minimal DFA M such that L = L(M)

Minimal means wrt number of states

Given a specification for L, via DFA, NFA or regex,this theorem is constructive.

NOT TRUE FOR NFAs





EXTENDING δ

Given DFA M = $(Q, \Sigma, \delta, q_0, F)$ extend δ to
$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ as follows:

$\hat{\delta}(q, \varepsilon) = q$

$\hat{\delta}(q, \sigma) = \delta(q, \sigma)$

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

Note: $\hat{\delta}(q_0, w) \in F \Leftrightarrow$ M accepts w

String $w \in \Sigma^*$ distinguishes states p and q iff
$\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \notin F$

EXTENDING δ

Given DFA M = $(Q, \Sigma, \delta, q_0, F)$ extend δ to
$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ as follows:

$\hat{\delta}(q, \varepsilon) = q$

$\hat{\delta}(q, \sigma) = \delta(q, \sigma)$

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

Note: $\hat{\delta}(q_0, w) \in F \Leftrightarrow$ M accepts w

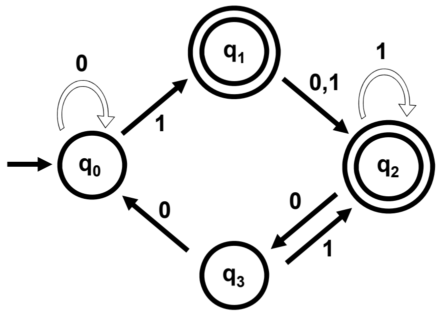
String $w \in \Sigma^*$ distinguishes states p and q iff
exactly ONE of $\hat{\delta}(p, w)$, $\hat{\delta}(q, w)$ is a final state

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

DEFINITION:

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

p is *indistinguishable* from q
iff
p is not distinguishable from q
iff
for all $w \in \Sigma^*$, $\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$



ε distinguishes accept from non-accept states

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

Define relation ~ :

$p \sim q$ iff p is indistinguishable from q
$p \nsim q$ iff p is distinguishable from q

Proposition: ~ is an equivalence relation

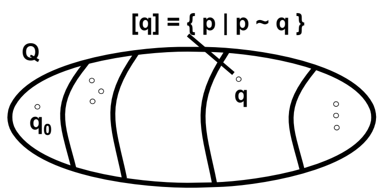
$p \sim p$ (reflexive)
$p \sim q \Rightarrow q \sim p$ (symmetric)
$p \sim q$ and $q \sim r \Rightarrow p \sim r$ (transitive)

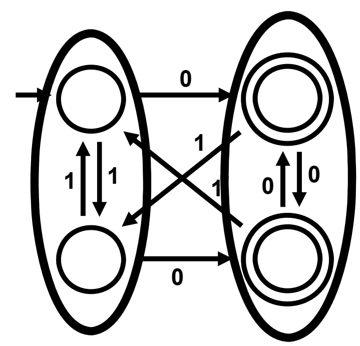
Proof (of transitivity): for all w, we have:
$\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F \Leftrightarrow \hat{\delta}(r, w) \in F$

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

Proposition: ~ is an equivalence relation

so ~ partitions the set of states of M into disjoint equivalence classes





Algorithm MINIMIZE

Input: DFA M

Output: DFA M_{MIN} such that:

$M \equiv M_{MIN}$ (that is, $L(M) = L(M_{MIN})$)

M_{MIN} has no inaccessible states

M_{MIN} is *irreducible*

||

all states of M_{MIN} are pairwise distinguishable

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

Intuition: States of M_{MIN} will be blocks of equivalent states of M

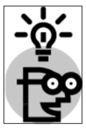
We'll find these equivalent states with a "Table-Filling" Algorithm

TABLE-FILLING ALGORITHM

Input: DFA M = (Q, Σ, δ, q_0, F)

Output: (1) D_M = { (p,q) | p,q ∈ Q and p/~ q }

(2) E_M = { [q] | q ∈ Q }



IDEA:

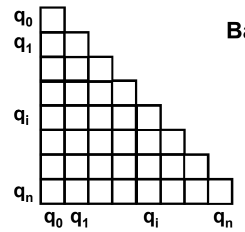
- **We know how to find those pairs of states that ε distinguishes…**
- **Use this and recursion to find those pairs distinguishable with *longer* strings**
- **Pairs left over will be indistinguishable**

TABLE-FILLING ALGORITHM

Input: DFA M = (Q, Σ, δ, q_0, F)

Output: (1) D_M = { (p,q) | p,q ∈ Q and p/~ q }

(2) E_M = { [q] | q ∈ Q }



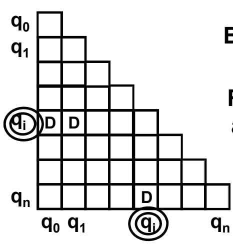
Base Case: p accepts and q "rejects" ⇒ p ≁ q

TABLE-FILLING ALGORITHM

Input: DFA M = (Q, Σ, δ, q_0, F)

Output: (1) D_M = { (p,q) | p,q ∈ Q and p/~ q }

(2) E_M = { [q] | q ∈ Q }

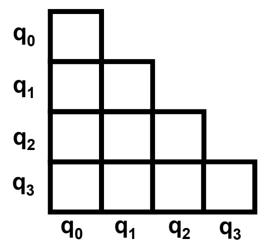


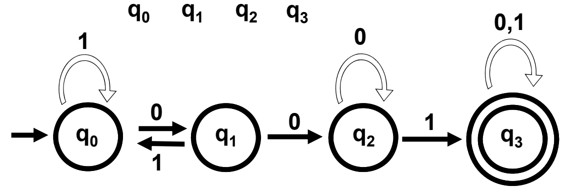
Base Case: p accepts and q "rejects" ⇒ p ≁ q

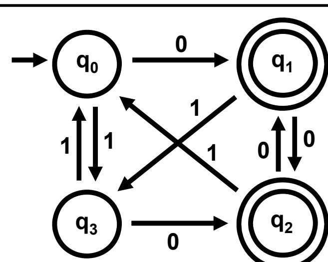
Recursion: if there is σ ∈ Σ and states p′, q′ satisfying

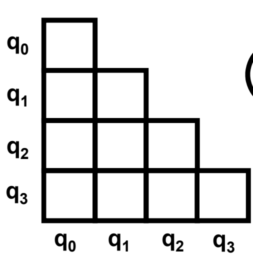
δ (p, σ) = p′
≁ ⇒ p ≁ q
δ (q, σ) = q′

Repeat until no more new D's









Claim: If p, q are distinguished by Table-Filling algorithm (ie pair labelled by D), then p ≁ q

Proof: By induction on the stage of the algorithm

If (p, q) is marked D at the start, then one's in F and one isn't, so ε distinguishes p and q

Suppose (p, q) is marked D at stage n+1
Then there are states p′, q′, string w ∈ Σ* and σ ∈ Σ such that:

1. **(p′, q′) are marked D ⇒ p′ ≁ q′ (by induction)**
   ⇒ $\hat{\delta}$(p′, w) ∈ F and $\hat{\delta}$(q′, w) ∉ F
2. **p′ = δ(p,σ) and q′ = δ(q,σ)**

The string σw distinguishes p and q!

Claim: If p, q are not distinguished by Table-Filling algorithm, then p ~ q

Proof (by contradiction):

Suppose the pair (p, q) is not marked D by the algorithm, yet p ≁ q (a "bad pair")

Suppose (p,q) is a bad pair with the shortest w.

$\hat{\delta}(p, w) \in F$ and $\hat{\delta}(q, w) \notin F$ **(Why is |w| >0 ?)**

So, w = $\sigma w'$, where $\sigma \in \Sigma$

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

Then (p′, q′) cannot be marked D (Why?)

But (p′, q′) is distinguished by w′ !

So (p′, q′) is also a bad pair, but with a SHORTER w′ !

Contradiction!

Algorithm MINIMIZE

Input: DFA M

Output: DFA M_{MIN}

(1) Remove all inaccessible states from M

(2) Apply Table-Filling algorithm to get:
E_M = { [q] | q is an accessible state of M }

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

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

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

Must show δ_{MIN} is well defined!

Algorithm MINIMIZE

Input: DFA M

Output: DFA M_{MIN}

(1) Remove all inaccessible states from M

(2) Apply Table-Filling algorithm to get:
E_M = { [q] | q is an accessible state of M }

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

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

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

Claim: $\hat{\delta}_{MIN}([q], w) = [\hat{\delta}(q, w)]$, $w \in \Sigma^*$

Algorithm MINIMIZE

Input: DFA M

Output: DFA M_{MIN}

(1) Remove all inaccessible states from M

(2) Apply Table-Filling algorithm to get:
E_M = { [q] | q is an accessible state of M }

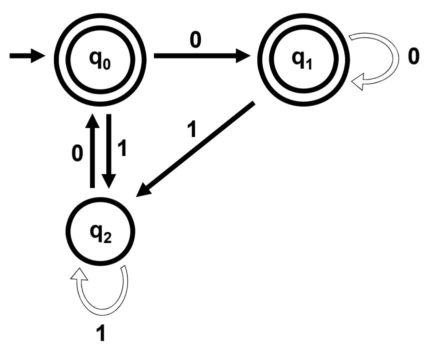
Define: $M_{MIN} = (Q_{MIN}, \Sigma, \delta_{MIN}, q_{0\,MIN}, F_{MIN})$

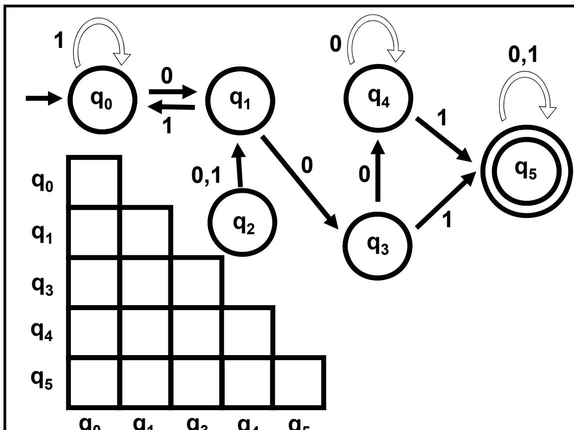
$Q_{MIN} = E_M$, $q_{0\,MIN} = [q_0]$, $F_{MIN} = \{ [q] \mid q \in F \}$

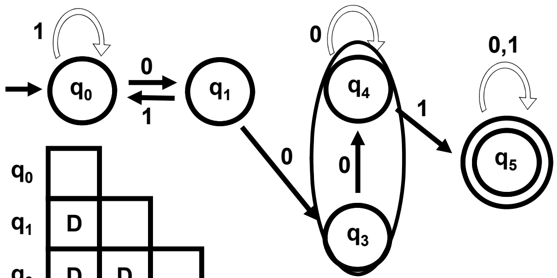
$\delta_{MIN}([q], \sigma) = [\delta(q, \sigma)]$

Follows: $M_{MIN} \equiv M$

MINIMIZE







	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	

PROPOSITION. Suppose $M' \equiv M$ and M' has no inaccessible states and is irreducible

Then, there exists a 1-1 onto correspondence between M_{MIN} and M' (preserving transitions**)**

i.e., M_{MIN} and M' are“Isomorphic”

COR: M_{MIN} is unique minimal DFA $\equiv$ M

PROPOSITION. Suppose $M' \equiv M$ and M' has no inaccessible states and is irreducible

Then, there exists a 1-1 onto correspondence between M_{MIN} and M' (preserving transitions**)**

i.e., M_{MIN} and M' are“Isomorphic”

COR: M_{MIN} is unique minimal DFA $\equiv$ M

Proof of Prop: We will construct a map recursively

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

Recursive Step: If $p \to p'$

$$\begin{array}{cc} p & \to & p' \\ \downarrow\sigma & & \downarrow\sigma \\ q & & q' \end{array} \quad \text{Then } q \to q'$$

We need to show:

The map is everywhere defined

The map is well defined

The map is a bijection (1-1 and onto)

The map preserves transitions

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

Recursive Step: If $p \to p'$

$$\begin{array}{cc} p & \to & p' \\ \downarrow\sigma & & \downarrow\sigma \\ q & & q' \end{array} \quad \text{Then } q \to q'$$

The map is everywhere defined:

That is, for all $q \in M_{MIN}$ there is a $q' \in M'$ such that $q \to q'$

If $q \in M_{MIN}$, there is a string w such that $\hat{\delta}_{MIN}(q_{0\ MIN}, w) = q$ (WHY?)

Let $q' = \hat{\delta}'(q_0', w)$. q will map to q' (by induction)

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

Recursive Step: If $p \to p'$

$$\begin{array}{cc} p & \to & p' \\ \downarrow\sigma & & \downarrow\sigma \\ q & & q' \end{array} \quad \text{Then } q \to q'$$

The map is well defined

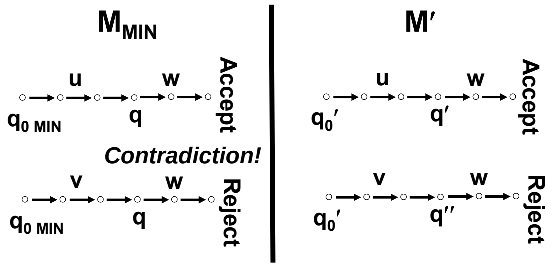
That is, for all $q \in M_{MIN}$ there is at most one $q' \in M'$ such that $q \to q'$

Suppose there exist q' and q'' such that $q \to q'$ and $q \to q''$

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

Suppose there exist q' and q'' such that
$q \to q'$ and $q \to q''$

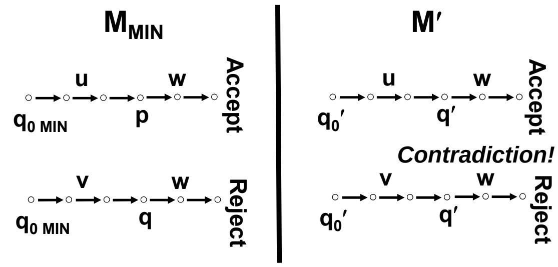
Suppose q' and q'' are distinguishable



The map is 1-1

Suppose there are distinct p and q such that
$p \to q'$ and $q \to q'$

p and q are distinguishable (why?)



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

Recursive Step: If $p \to p'$, $p \xrightarrow{\sigma} q$, $p' \xrightarrow{\sigma} q'$, Then $q \to q'$

The map is onto

That is, for all $q' \in M'$ there is a $q \in M_{MIN}$ such that $q \to q'$

If $q' \in M'$, there is w such that $\hat{\delta}'(q_0',w) = q'$

Let $q = \hat{\delta}_{MIN}(q_{0\,MIN},w)$. q will map to q' (why?)

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

Recursive Step: If $p \to p'$, $p \xrightarrow{\sigma} q$, $p' \xrightarrow{\sigma} q'$, Then $q \to q'$

The map preserves transitions

That is, if $\delta(p, \sigma) = q$ and $p \to p'$ and $q \to q'$

then, $\delta'(p', \sigma) = q'$

(Why?)

How can we prove that two regular expressions are equivalent?

WWW.FLAC.WS

Read Chapters 2.1 & 2.2 for next time