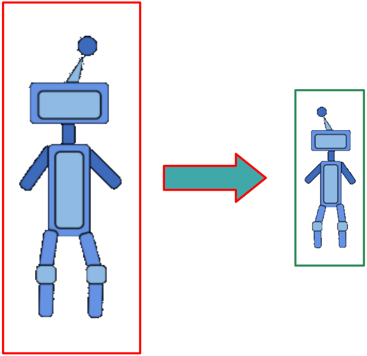
Finite Automata

- Deterministic Finite Automata
- Languages decided by a DFA – Regular Languages
- Closure Properties of regular languages
- Non-Deterministic Finite Automata, DFA= NFA
- Regular Expression: Computation as Description
- DFA=NFA=RegExp, Generalized NFA
- Non-Regular Languages, The Pumping Lemma
- Minimizing DFA

Imdad ullah Khan

Minimizing DFA

Minimizing DFA

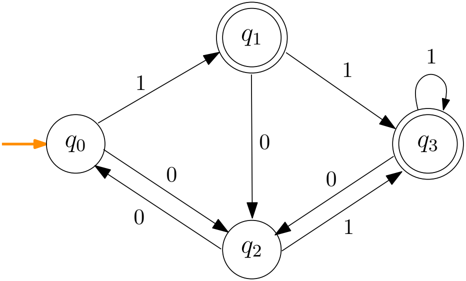


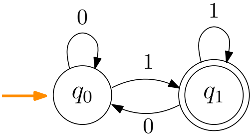
Reduce the "complexity" of DFA why?

Computational and storage efficiency

Minimizing DFA

What are the languages decided by these DFA's? Let $\Sigma = \{0, 1\}$





$\{w\ :\ w \text{ ends with a } 1\}$

Minimizing DFA

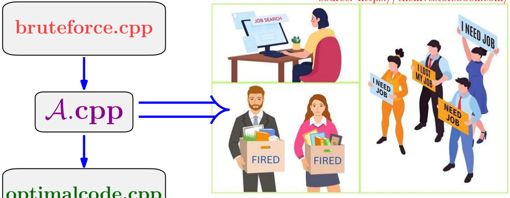
Theorem (DFA Minimization Theorem)

1. *For every regular language L, there is a unique (up to re-labeling of the states) minimal-state DFA M^* such that $L = L(M^*)$*
2. *There is an efficient algorithm to convert any M to the unique minimal state DFA M^*, such that $L(M) = L(M^*)$*

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

If there is a program $\mathcal{A}$.*cpp* that could convert any program to the most optimal, then . . .

source: https://theinvestorsbook.com/



Programmers

Minimizing DFA

Theorem (DFA Minimization Theorem)

1. *For every regular language L, there is a unique (up to re-labeling of the states) minimal-state DFA M^* such that $L = L(M^*)$*
2. *There is an efficient algorithm to convert any M to the unique minimal state DFA M^*, such that $L(M) = L(M^*)$*

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

Both these NFAs have minimal number of states

Extended transition function

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

Extend the transition function to strings

$$\delta : Q \times \Sigma \mapsto Q \qquad \text{to} \qquad \Delta : Q \times \Sigma^* \mapsto Q$$

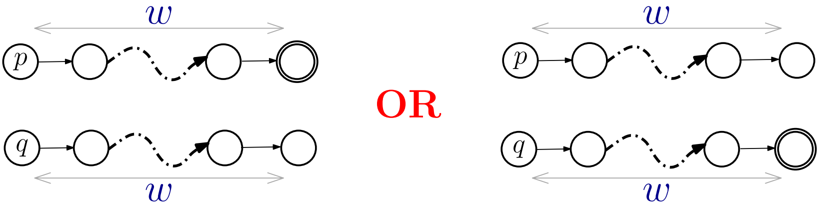
- $\Delta(q, \epsilon) \;=\; q$
- $\Delta(q, \sigma) \;=\; \delta(q, \sigma)$
- $\Delta(q, \sigma_1 \ldots \sigma_k) \;=\; \delta\big(\Delta(q, \sigma_1 \ldots \sigma_{k-1}), \sigma_k\big)$

Distinguishing States with Strings

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

A string $w \in \Sigma^*$ distinguishes two states p and q if exactly one of $\Delta(p, w)$ is in final state i.e.

$$[\,\Delta(p, w) \in F\,] \oplus [\,\Delta(q, w) \in F\,]$$



Distinguishing States with Strings

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

A string $w \in \Sigma^*$ distinguishes two states p and q if exactly one of $\Delta(p, w)$ is in final state i.e.

$$[\,\Delta(p, w) \in F\,] \oplus [\,\Delta(q, w) \in F\,]$$

States p and q are distinguishable iff there exists $w \in \Sigma^*$ that distinguishes them i.e. $\exists w \in \Sigma^*$ such that $\Delta(p, w) \in F \iff \Delta(q, w) \notin F$

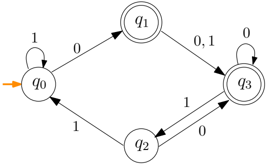
States p and q are indistinguishable iff no $w \in \Sigma^*$ distinguishes them i.e. $\forall w \in \Sigma^*$ we have $\Delta(p, w) \in F \iff \Delta(q, w) \in F$

Pairs of indistinguishable states are redundant, i.e. M has exactly the same behavior starting from p and q

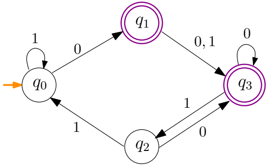
Distinguishing States with Strings

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

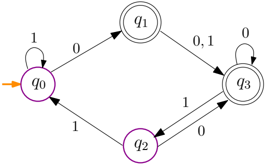
States p and q are distinguishable iff there exists $w \in \Sigma^*$ that distinguishes them i.e. $\exists w \in \Sigma^* \quad$ such that $\quad \Delta(p, w) \in F \iff \Delta(q, w) \notin F$



The string ϵ distinguishes all final states from all non-final states



1 distinguishes q_1 and q_3
0 does not distinguish them



01 distinguishes q_0 and q_2
0, 1, 10 do not distinguish them

Indistinguishable is an equivalence relation

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

States p and q are indistinguishable iff no $w \in \Sigma^*$ distinguishes them i.e.
$\forall w \in \Sigma^*$ we have $\Delta(p, w) \in F \iff \Delta(q, w) \in F$

Let $\sim$ be a binary relation on Q such that

$$p \sim q \iff p \text{ is indistinguishable from } q$$

$\triangleright$ $p \nsim q \iff p$ is distinguishable from q

1. $\forall q \in Q \; q \sim q$
2. $\forall p, q \in Q \; q \sim q \implies q \sim p$
3. $\forall p, q, r \in Q \; p \sim q \wedge q \sim r \implies p \sim r$

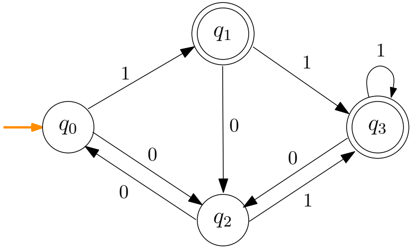
A relation R on a set X is an **equivalence relation** if it is

1. reflexive
2. symmetric, and
3. transitive

Distinguishing States with Strings

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

States p and q are indistinguishable iff no $w \in \Sigma^*$ distinguishes them i.e.
$\forall w \in \Sigma^*$ we have $\Delta(p, w) \in F \iff \Delta(q, w) \in F$



$\mathbf{q_0 \nsim q_1}$ $\mathbf{q_2 \nsim q_1}$ $\mathbf{q_0 \sim q_2}$

$\mathbf{q_0 \nsim q_3}$ $\mathbf{q_2 \nsim q_3}$ $\mathbf{q_1 \sim q_3}$

Indistinguishable is an equivalence relation

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

States p and q are indistinguishable iff no $w \in \Sigma^*$ distinguishes them i.e.
$\forall w \in \Sigma^*$ we have $\Delta(p, w) \in F \iff \Delta(q, w) \in F$

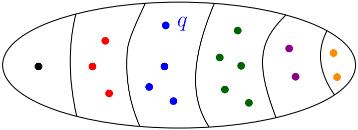
Let $\sim$ be a binary relation on Q such that

$$p \sim q \iff p \text{ is indistinguishable from } q$$

▷ $p \nsim q \iff p$ is distinguishable from q

$[q] := \{p \,|\, p \sim q\}$

$\sim$ partitions the states of M into disjoint equivalence classes



Indistinguishable is an equivalence relation

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

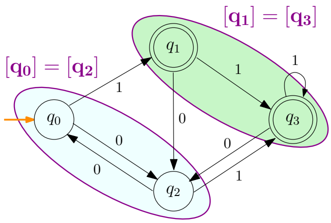
States p and q are indistinguishable iff no $w \in \Sigma^*$ distinguishes them i.e.
$\forall w \in \Sigma^*$ we have $\Delta(p, w) \in F \iff \Delta(q, w) \in F$

Let $\sim$ be a binary relation on Q such that

$$p \sim q \iff p \text{ is indistinguishable from } q$$

▷ $p \nsim q \iff p$ is distinguishable from q

$\sim$ partitions the states of M into disjoint equivalence classes



$\mathbf{q_0 \nsim q_1}$ $\mathbf{q_2 \nsim q_1}$ $\mathbf{q_0 \sim q_2}$

$\mathbf{q_0 \nsim q_3}$ $\mathbf{q_2 \nsim q_3}$ $\mathbf{q_1 \sim q_3}$

Minimizing DFA

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

Output: A DFA M_{min} such that

1. $L(M) = L(M_{min})$
2. M_{min} has no inaccessible states
3. M_{min} is irreducible $\triangleright$ All states p and q of M_{min} are indistinguishable

Theorem

M_{min} is the unique minimal equivalent to M DFA (up to states relabeling)

Intuitively, states of M_{min} are equivalence classes of M (under $\sim$)

How to find equivalence classes of Q?

What are transitions in M_{min}?

Table Filling Algorithm to find indistinguishable states

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

Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p, q) = D \iff p \nsim q$

Dynamic Programming Formulation

1. In iteration 0, mark pairs of states distinguishable by ϵ
2. In iteration i, find pairs of states distinguishable by strings of length i
3. In iteration $i + 1$, given pairs of states distinguishable by strings of length $\leq i$, mark the pairs distinguishable by strings of length $i + 1$

Table Filling Algorithm to find indistinguishable states

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

Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p,q) = D \iff p \nsim q$

Algorithm Table Filling Algorithm

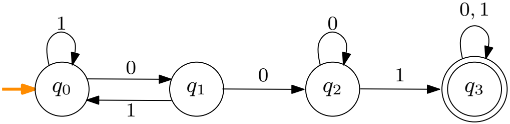
if $p \in F$ and $q \notin F$ **then**
  $\mathcal{D}(p,q) \leftarrow D$
while $\mathcal{D}$ changed in the previous iteration **do**
  for $p, q \in Q \times Q$ and $\sigma \in \Sigma$ **do**
    if $\delta(p,\sigma) = p'$ and $\delta(q,\sigma) = q'$ AND $\mathcal{D}(p', q') = D$ **then**
      $\mathcal{D}(p,q) \leftarrow D$

Table Filling Algorithm to find indistinguishable states

Input: A DFA $M = (Q, \Sigma, \delta, q_0, F)$
Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p,q) = D \iff p \not\approx q$

Algorithm Table Filling Algorithm

if $p \in F$ and $q \notin F$ **then**
  $\mathcal{D}(p,q) \leftarrow D$
while $\mathcal{D}$ changed in the previous iteration **do**
  for $p, q \in Q \times Q$ and $\sigma \in \Sigma$ **do**
    if $\delta(p,\sigma) = p'$ and $\delta(q,\sigma) = q'$ AND $\mathcal{D}(p',q') = D$ **then**
      $\mathcal{D}(p,q) \leftarrow D$



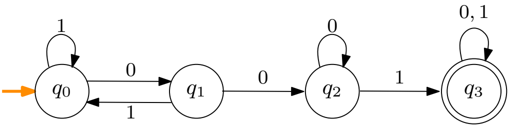
	q_0	q_1	q_2	q_3
q_0				
q_1				
q_2				
q_3				

Table Filling Algorithm to find indistinguishable states

Input: A DFA $M = (Q, \Sigma, \delta, q_0, F)$
Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p,q) = D \iff p \not\approx q$

Algorithm Table Filling Algorithm

if $p \in F$ and $q \notin F$ **then**
  $\mathcal{D}(p,q) \leftarrow D$
while $\mathcal{D}$ changed in the previous iteration **do**
  for $p, q \in Q \times Q$ and $\sigma \in \Sigma$ **do**
    if $\delta(p,\sigma) = p'$ and $\delta(q,\sigma) = q'$ AND $\mathcal{D}(p',q') = D$ **then**
      $\mathcal{D}(p,q) \leftarrow D$



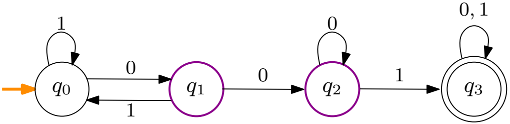
q_0				
q_1				
q_2				
q_3	D	D	D	
	q_0	q_1	q_2	q_3

Table Filling Algorithm to find indistinguishable states

Input: A DFA $M = (Q, \Sigma, \delta, q_0, F)$
Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p,q) = D \iff p \not\approx q$

Algorithm Table Filling Algorithm

if $p \in F$ and $q \notin F$ **then**
  $\mathcal{D}(p,q) \leftarrow D$
while $\mathcal{D}$ changed in the previous iteration **do**
  for $p, q \in Q \times Q$ and $\sigma \in \Sigma$ **do**
    if $\delta(p,\sigma) = p'$ and $\delta(q,\sigma) = q'$ AND $\mathcal{D}(p',q') = D$ **then**
      $\mathcal{D}(p,q) \leftarrow D$



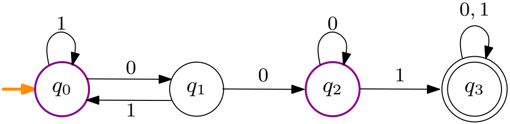
	q_0	q_1	q_2	q_3
q_0				
q_1				
q_2		D		
q_3	D	D	D	

Table Filling Algorithm to find indistinguishable states

Input: A DFA $M = (Q, \Sigma, \delta, q_0, F)$
Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p, q) = D \iff p \not\approx q$

Algorithm Table Filling Algorithm

if $p \in F$ and $q \notin F$ **then**
  $\mathcal{D}(p, q) \leftarrow D$
while $\mathcal{D}$ changed in the previous iteration **do**
  for $p, q \in Q \times Q$ and $\sigma \in \Sigma$ **do**
    if $\delta(p, \sigma) = p'$ and $\delta(q, \sigma) = q'$ AND $\mathcal{D}(p', q') = D$ **then**
      $\mathcal{D}(p, q) \leftarrow D$



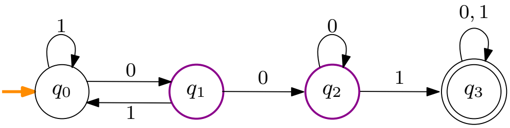
q_0				
q_1				
q_2	D	D		
q_3	D	D	D	
	q_0	q_1	q_2	q_3

Table Filling Algorithm to find indistinguishable states

Input: A DFA $M = (Q, \Sigma, \delta, q_0, F)$
Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p,q) = D \iff p \not\approx q$

Algorithm Table Filling Algorithm

if $p \in F$ and $q \notin F$ **then**
  $\mathcal{D}(p,q) \leftarrow D$
while $\mathcal{D}$ changed in the previous iteration **do**
  for $p, q \in Q \times Q$ and $\sigma \in \Sigma$ **do**
    if $\delta(p,\sigma) = p'$ and $\delta(q,\sigma) = q'$ AND $\mathcal{D}(p',q') = D$ **then**
      $\mathcal{D}(p,q) \leftarrow D$



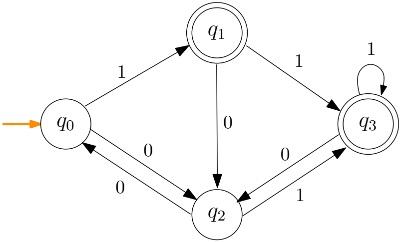
q_0				
q_1	D			
q_2	D	D		
q_3	D	D	D	
	q_0	q_1	q_2	q_3

Table Filling Algorithm to find indistinguishable states

Input: A DFA $M = (Q, \Sigma, \delta, q_0, F)$
Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p,q) = D \iff p \nsim q$

Algorithm Table Filling Algorithm

if $p \in F$ and $q \notin F$ **then**
  $\mathcal{D}(p,q) \leftarrow D$
while $\mathcal{D}$ changed in the previous iteration **do**
  for $p, q \in Q \times Q$ and $\sigma \in \Sigma$ **do**
    if $\delta(p,\sigma) = p'$ and $\delta(q,\sigma) = q'$ AND $\mathcal{D}(p',q') = D$ **then**
      $\mathcal{D}(p,q) \leftarrow D$



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

Table Filling Algorithm to find indistinguishable states

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

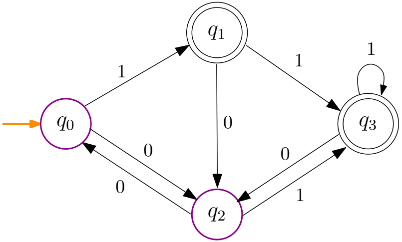
Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p, q) = D \iff p \not\approx q$

Algorithm Table Filling Algorithm

---

if $p \in F$ and $q \notin F$ **then**
  $\mathcal{D}(p, q) \leftarrow D$

while $\mathcal{D}$ changed in the previous iteration **do**
  for $p, q \in Q \times Q$ and $\sigma \in \Sigma$ **do**
    if $\delta(p, \sigma) = p'$ and $\delta(q, \sigma) = q'$ AND $\mathcal{D}(p', q') = D$ **then**
      $\mathcal{D}(p, q) \leftarrow D$



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

Table Filling Algorithm to find indistinguishable states

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

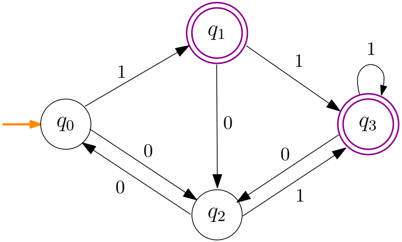
Output: A $Q \times Q$ matrix $\mathcal{D}$, such that $\mathcal{D}(p, q) = D \iff p \not\approx q$

Algorithm Table Filling Algorithm

---

if $p \in F$ and $q \notin F$ **then**
  $\mathcal{D}(p, q) \leftarrow D$

while $\mathcal{D}$ changed in the previous iteration **do**
  for $p, q \in Q \times Q$ and $\sigma \in \Sigma$ **do**
    if $\delta(p, \sigma) = p'$ and $\delta(q, \sigma) = q'$ AND $\mathcal{D}(p', q') = D$ **then**
      $\mathcal{D}(p, q) \leftarrow D$



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

Minimizing DFA

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

Output: A DFA M_{min} such that

1. $L(M) = L(M_{min})$
2. M_{min} has no inaccessible states
3. M_{min} is irreducible ▷ All states p and q of M_{min} are indistinguishable

Theorem

M_{min} is the unique minimal equivalent to M DFA (up to states relabeling)

Intuitively, states of M_{min} are equivalence classes of M (under $\sim$)

How to find equivalence classes of Q?

What are transitions in M_{min}?

Minimizing DFA

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

Output: A DFA M_{min} with fewest states and $L(M_{min}) = L(M)$

Algorithm DFA Minimizing Algorithm

1: Remove all inaccessible states from M

2: $\textsc{Table-Filling}(M)$ to get $\text{EQUIV}_M = \{[q] : q \text{ is an accessible state in } M\}$

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

$$Q_{min} = \text{EQUIV}_{min}$$
$$q_{0\,min} = [q_0]$$
$$F_{min} = \{[q] : q \in F\}$$
$$\delta_{min}([q], \sigma) = [\delta(q, \sigma)]$$

Minimizing DFA

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

Output: A DFA M_{min} with fewest states and $L(M_{min}) = L(M)$

Algorithm DFA Minimizing Algorithm

1: Remove all inaccessible states from M

2: $\textsc{Table-Filling}(M)$ to get

$$\text{EQUIV}_M = \{[q] : q \text{ accessible in } M\}$$

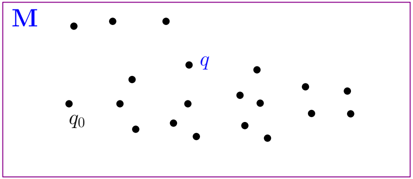
3: $M_{min} = (Q_{min}, \Sigma, \delta_{min}, q_{0\,min}, F_{min})$

$$Q_{min} = \text{EQUIV}_{min}$$

$$q_{0\,min} = [q_0]$$

$$F_{min} = \{[q] \ : \ q \in F\}$$

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



Minimizing DFA

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

Output: A DFA M_{min} with fewest states and $L(M_{min}) = L(M)$

Algorithm DFA Minimizing Algorithm

1: Remove all inaccessible states from M

2: TABLE-FILLING(M) to get

$\text{EQUIV}_M = \{[q] : q \text{ accessible in } M\}$

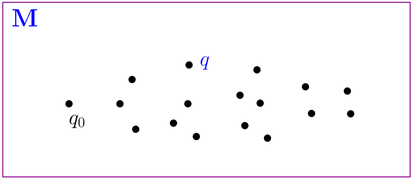
3: $M_{min} = (Q_{min}, \Sigma, \delta_{min}, q_{0\,min}, F_{min})$

$Q_{min} = \text{EQUIV}_{min}$

$q_{0\,min} = [q_0]$

$F_{min} = \{[q] \,:\, q \in F\}$

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



Minimizing DFA

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

Output: A DFA M_{min} with fewest states and $L(M_{min}) = L(M)$

Algorithm DFA Minimizing Algorithm

1: Remove all inaccessible states from M

2: TABLE-FILLING(M) to get

$$\text{EQUIV}_M = \{[q] : q \text{ accessible in } M\}$$

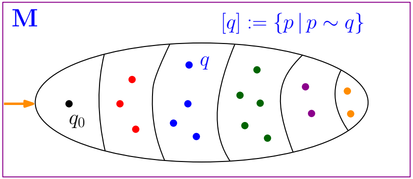
3: $M_{min} = (Q_{min}, \Sigma, \delta_{min}, q_{0\,min}, F_{min})$

$$Q_{min} = \text{EQUIV}_{min}$$

$$q_{0\,min} = [q_0]$$

$$F_{min} = \{[q] \,:\, q \in F\}$$

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



Minimizing DFA

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

Output: A DFA M_{min} with fewest states and $L(M_{min}) = L(M)$

Algorithm DFA Minimizing Algorithm

1: Remove all inaccessible states from M

2: TABLE-FILLING(M) to get

$\text{EQUIV}_M = \{[q] : q \text{ accessible in } M\}$

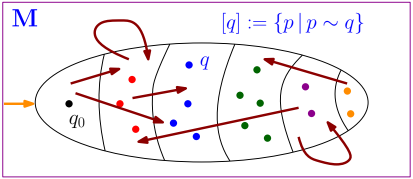
3: $M_{min} = (Q_{min}, \Sigma, \delta_{min}, q_{0\,min}, F_{min})$

$Q_{min} = \text{EQUIV}_{min}$

$q_{0\,min} = [q_0]$

$F_{min} = \{[q] \; : \; q \in F\}$

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



Minimizing DFA

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

Output: A DFA M_{min} with fewest states and $L(M_{min}) = L(M)$

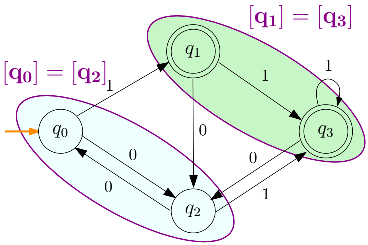
Algorithm DFA Minimizing Algorithm

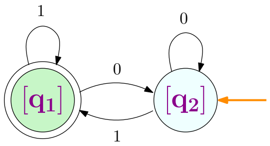
1: Remove all inaccessible states from M

2: $\textsc{Table-Filling}(M)$ to get $\textsc{equiv}_M = \{[q] : q \text{ accessible in } M\}$

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

$$Q_{min} = \textsc{equiv}_{min} \quad q_{0\,min} = [q_0] \quad F_{min} = \{[q] : q \in F\} \quad \delta_{min}([q], \sigma) = [\delta(q, \sigma)]$$





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

Output: A DFA M_{min} with fewest states and $L(M_{min}) = L(M)$

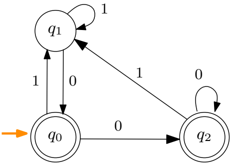
Algorithm DFA Minimizing Algorithm

1: Remove all inaccessible states from M

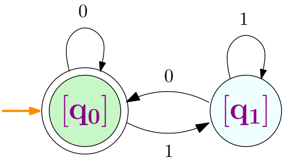
2: $\textsc{Table-Filling}(M)$ to get $\textsc{equiv}_M = \{[q] : q \text{ accessible in } M\}$

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

$Q_{min} = \textsc{equiv}_{min}$ $\quad q_{0\,min} = [q_0]$ $\quad F_{min} = \{[q] : q \in F\}$ $\quad \delta_{min}([q], \sigma) = [\delta(q, \sigma)]$



	q_0	q_1	q_2
q_0			
q_1	D		
q_2		D	



Minimizing DFA

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

Output: A DFA M_{min} with fewest states and $L(M_{min}) = L(M)$

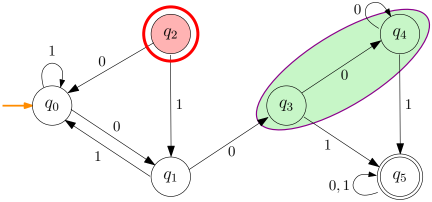
Algorithm DFA Minimizing Algorithm

1: Remove all inaccessible states from M

2: $\textsc{Table-Filling}(M)$ to get $\text{EQUIV}_M = \{[q] : q \text{ accessible in } M\}$

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

$Q_{min} = \text{EQUIV}_{min} \quad q_{0\,min} = [q_0] \quad F_{min} = \{[q] : q \in F\} \quad \delta_{min}([q], \sigma) = [\delta(q, \sigma)]$



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