Lecture 10 – Equivalence and Minimization of Finite Automata

COSE215: Theory of Computation

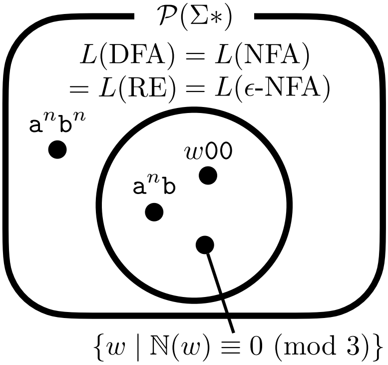
Jihyeok Park

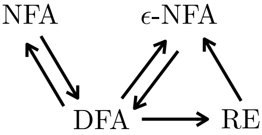
PLRG

2023 Spring

Recall

- Closure Properties of Regular Languages
- Pumping Lemma for Regular Languages





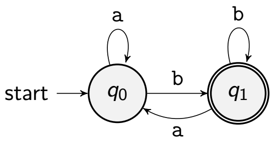
- How to test whether two finite automata are equivalent?
- How to minimize a finite automaton?

Contents

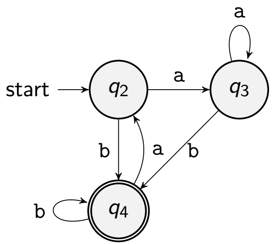
Equivalence of Finite Automata

- Are the following two DFA **equivalent** (i.e., $L(D_0) = L(D_1)$)?

$D_0 =$



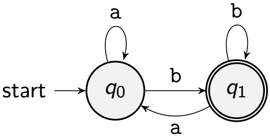
$D_1 =$



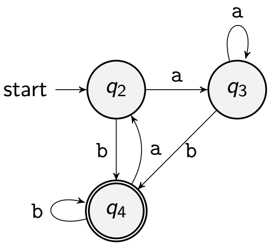
Equivalence of Finite Automata

- Are the following two DFA **equivalent** (i.e., $L(D_0) = L(D_1)$)?

$D_0 =$



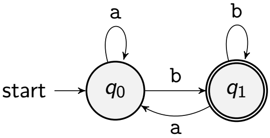
$D_1 =$

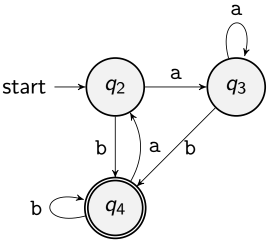


- Yes, because $L(D_0) = L(D_1) = \{w\mathtt{b} \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$.

Equivalence of Finite Automata

- Are the following two DFA **equivalent** (i.e., $L(D_0) = L(D_1)$)?

$D_0 =$ 

$D_1 =$ 

- Yes, because $L(D_0) = L(D_1) = \{w\texttt{b} \mid w \in \{\texttt{a}, \texttt{b}\}^*\}$.
- We first define the **equivalence of states** and utilize it to test the **equivalence of DFA**.

Equivalence of States ($\equiv$)

Definition (Equivalence of States ($\equiv$))

For a given DFA D, q_i is **equivalent** to q_j (i.e., $q_i \equiv q_j$) if and only if

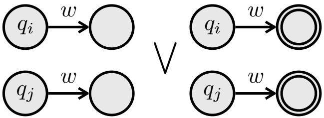
$$\forall w \in \Sigma^*.\ \delta^*(q_i, w) \in F \iff \delta^*(q_j, w) \in F$$

Definition (Equivalence of States ($\equiv$))

For a given DFA D, q_i is **equivalent** to q_j (i.e., $q_i \equiv q_j$) if and only if

$$\forall w \in \Sigma^*.\ \delta^*(q_i, w) \in F \iff \delta^*(q_j, w) \in F$$

$q_i \equiv q_j \iff \forall w \in \Sigma^*$



Equivalence of States ($\equiv$)

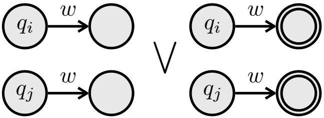
PLRG

Definition (Equivalence of States ($\equiv$))

For a given DFA D, q_i is **equivalent** to q_j (i.e., $q_i \equiv q_j$) if and only if

$$\forall w \in \Sigma^*.\ \delta^*(q_i, w) \in F \iff \delta^*(q_j, w) \in F$$

$q_i \equiv q_j \iff \forall w \in \Sigma^*$



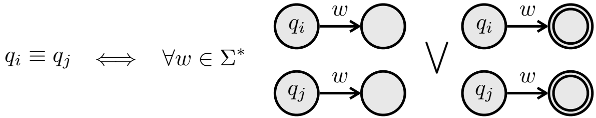
However, it is difficult to make it as an algorithm.

Equivalence of States ($\equiv$)

Definition (Equivalence of States ($\equiv$))

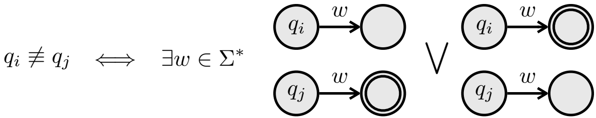
For a given DFA D, q_i is **equivalent** to q_j (i.e., $q_i \equiv q_j$) if and only if

$$\forall w \in \Sigma^*.\ \delta^*(q_i, w) \in F \iff \delta^*(q_j, w) \in F$$



However, it is difficult to make it as an algorithm. Let's consider $q_i \not\equiv q_j$:

$$q_i \not\equiv q_j \iff \exists w \in \Sigma^*.\ (\delta^*(q_i, w) \in F \iff \delta^*(q_j, w) \notin F)$$



Distinguishable States ($\not\equiv$)

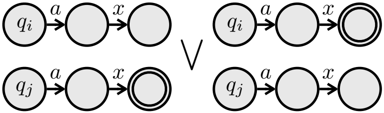
We can *inductively* test q_i is **distinguishable** with q_j (i.e., $q_i \not\equiv q_j$):

- **(Basis Case)** $w = \epsilon$

$$(\delta^*(q_i, \epsilon) \in F \iff \delta^*(q_j, \epsilon) \notin F)$$
$$\iff q_i \in F \iff q_j \notin F$$

- **(Induction Case)** $w = ax$

$\exists a \in \Sigma.\ \exists x \in \Sigma^*$



$$\exists a \in \Sigma.\ \exists x \in \Sigma^*.\ (\delta^*(q_i, ax) \in F \iff \delta^*(q_j, ax) \notin F)$$
$$\iff \exists a \in \Sigma.\ \exists x \in \Sigma^*.\ (\delta^*(\delta(q_i, a), x) \in F \iff \delta^*(\delta(q_j, a), x) \notin F)$$
$$\iff \exists a \in \Sigma.\ \delta(q_i, a) \not\equiv \delta(q_j, a)$$

Distinguishable States ($\not\equiv$)

Definition (Distinguishable States ($\not\equiv$))

For a given DFA D, q_i is **distinguishable** with q_j (i.e., $q_i \not\equiv q_j$) if and only if

- **(Basis Case)** $q_i \in F \iff q_j \notin F$.
- **(Induction Case)** $\exists \texttt{a} \in \Sigma.\ \delta(q_i, \texttt{a}) \not\equiv \delta(q_j, \texttt{a})$.

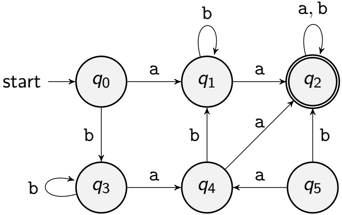
Distinguishable States ($\not\equiv$)

PLRG

Definition (Distinguishable States ($\not\equiv$))

For a given DFA D, q_i is **distinguishable** with q_j (i.e., $q_i \not\equiv q_j$) if and only if

- **(Basis Case)** $q_i \in F \iff q_j \notin F$.
- **(Induction Case)** $\exists \texttt{a} \in \Sigma.\ \delta(q_i, \texttt{a}) \not\equiv \delta(q_j, \texttt{a})$.

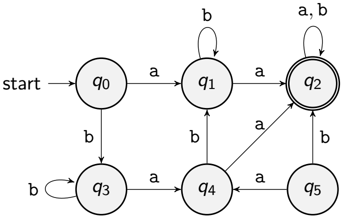


$q_2 \not\equiv q_4$
$(\because q_2 \in F \land q_4 \notin F)$

$q_1 \not\equiv q_3$
$(\because \delta(q_1, \texttt{a}) = q_2 \not\equiv q_4 = \delta(q_3, \texttt{a}))$

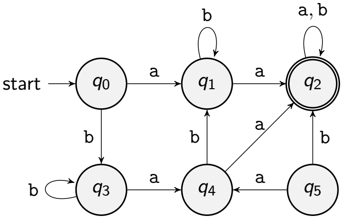
$q_0 \not\equiv q_4$
$(\because \delta(q_0, \texttt{b}) = q_3 \not\equiv q_1 = \delta(q_4, \texttt{b}))$

Table-Filling Algorithm



q	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_2	q_1
$*q_2$	q_2	q_2
q_3	q_4	q_3
q_4	q_2	q_1
q_5	q_4	q_2

Table-Filling Algorithm



q	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_2	q_1
$*q_2$	q_2	q_2
q_3	q_4	q_3
q_4	q_2	q_1
q_5	q_4	q_2

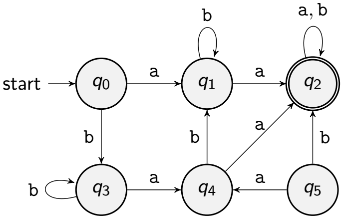
(Basis case) $w = \epsilon$.

$$q_i \in F \iff q_j \notin F$$

(Induction case) $w = ax$.

$$\exists a \in \Sigma.\ \delta(q_i, a) \not\equiv \delta(q_j, a)$$

Table-Filling Algorithm



q	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_2	q_1
$*q_2$	q_2	q_2
q_3	q_4	q_3
q_4	q_2	q_1
q_5	q_4	q_2

(Basis case) $w = \epsilon$.

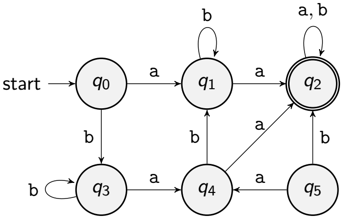
$$q_i \in F \iff q_j \notin F$$

(Induction case) $w = ax$.

$$\exists a \in \Sigma.\ \delta(q_i, a) \not\equiv \delta(q_j, a)$$

q_1					
q_2					
q_3					
q_4					
q_5					
	q_0	q_1	q_2	q_3	q_4

Table-Filling Algorithm



q	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_2	q_1
$*q_2$	q_2	q_2
q_3	q_4	q_3
q_4	q_2	q_1
q_5	q_4	q_2

(Basis case) $w = \epsilon$.

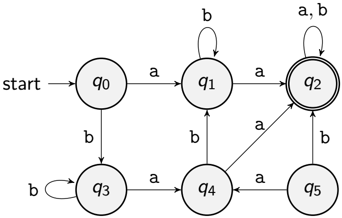
$$q_i \in F \iff q_j \notin F$$

(Induction case) $w = ax$.

$$\exists a \in \Sigma.\ \delta(q_i, a) \not\equiv \delta(q_j, a)$$

q_1	x				
q_2	x	x			
q_3		x	x		
q_4	x		x	x	
q_5	x	x	x	x	x
	q_0	q_1	q_2	q_3	q_4

Table-Filling Algorithm



q	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_2	q_1
$*q_2$	q_2	q_2
q_3	q_4	q_3
q_4	q_2	q_1
q_5	q_4	q_2

(Basis case) $w = \epsilon$.

$$q_i \in F \iff q_j \notin F$$

(Induction case) $w = ax$.

$$\exists a \in \Sigma.\ \delta(q_i, a) \not\equiv \delta(q_j, a)$$

	q_0	q_1	q_2	q_3	q_4
q_1	X				
q_2	X	X			
q_3		X	X		
q_4	X		X	X	
q_5	X	X	X	X	X

$$q_0 \equiv q_3 \wedge q_1 \equiv q_4$$

PLRG

Theorem (Equivalence of Finite Automata)

Consider two DFA $D = (Q, \Sigma, \delta, q_0, F)$ *and* $D' = (Q', \Sigma, \delta', q_0', F')$

$$L(D) = L(D') \iff q_0 \equiv q_0'$$

in a DFA $D'' = (Q \uplus Q', \Sigma, \delta'', q_0, F \uplus F')$ *where*

$$\forall q'' \in Q \uplus Q'.\ \delta''(q, a) = \begin{cases} \delta(q'', a) & q'' \in Q \\ \delta'(q'', a) & q'' \in Q' \end{cases}$$

Proof) By the definition of equivalence of states, we have

PLRG

Theorem (Equivalence of Finite Automata)

Consider two DFA $D = (Q, \Sigma, \delta, q_0, F)$ *and* $D' = (Q', \Sigma, \delta', q'_0, F')$

$$L(D) = L(D') \iff q_0 \equiv q'_0$$

in a DFA $D'' = (Q \uplus Q', \Sigma, \delta'', q_0, F \uplus F')$ *where*

$$\forall q'' \in Q \uplus Q'.\ \delta''(q, a) = \begin{cases} \delta(q'', a) & q'' \in Q \\ \delta'(q'', a) & q'' \in Q' \end{cases}$$

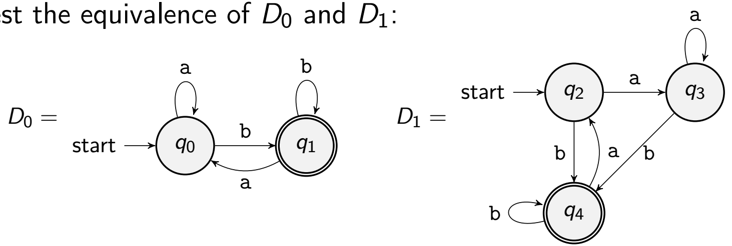
Proof) By the definition of equivalence of states, we have

$$\begin{aligned} L(D) = L(D') &\iff \forall w \in \Sigma^*.\ (D \text{ accepts } w \iff D' \text{ accepts } w) \\ &\iff \forall w \in \Sigma^*.\ (\delta^*(q_0, w) \in F \iff \delta'^*(q'_0, w) \in F') \\ &\iff \forall w \in \Sigma^*.\ (\delta''^*(q_0, w) \in F \cup F' \iff \delta''^*(q'_0, w) \in F \cup F') \\ &\iff q_0 \equiv q'_0 \text{ in } D'' \end{aligned}$$

□

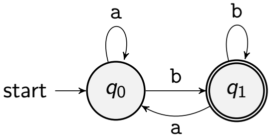
Equivalence of Finite Automata – Example 1

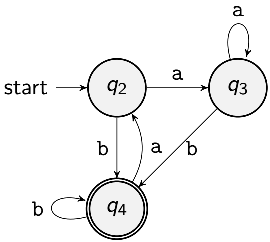
Let's test the equivalence of D_0 and D_1:



Equivalence of Finite Automata – Example 1

Let's test the equivalence of D_0 and D_1:

$D_0 =$ 

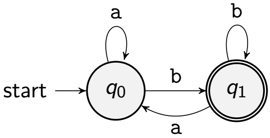
$D_1 =$ 

Let's perform the **table-filling algorithm**:

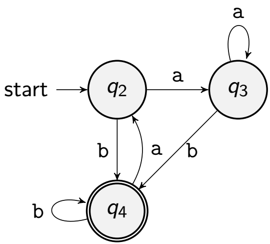
Equivalence of Finite Automata – Example 1

Let's test the equivalence of D_0 and D_1:

$D_0 =$



$D_1 =$



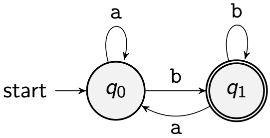
Let's perform the **table-filling algorithm**:

q_1	X			
q_2		X		
q_3		X		
q_4	X		X	X
	q_0	q_1	q_2	q_3

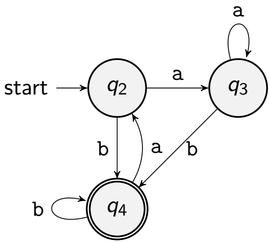
Equivalence of Finite Automata – Example 1

Let's test the equivalence of D_0 and D_1:

$D_0 =$



$D_1 =$



Let's perform the **table-filling algorithm**:

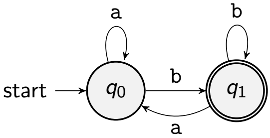
q_1	x			
q_2		x		
q_3		x		
q_4	x		x	x
	q_0	q_1	q_2	q_3

- $q_0 \equiv q_2 \equiv q_3$
- $q_1 \equiv q_4$

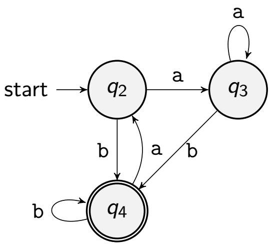
Equivalence of Finite Automata – Example 1

Let's test the equivalence of D_0 and D_1:

$D_0 =$



$D_1 =$



Let's perform the **table-filling algorithm**:

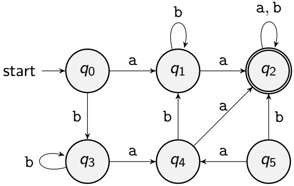
	q_0	q_1	q_2	q_3
q_1	x			
q_2		x		
q_3		x		
q_4	x		x	x

- $q_0 \equiv q_2 \equiv q_3$
- $q_1 \equiv q_4$

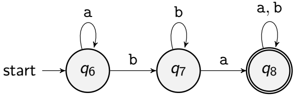
$$q_0 \equiv q_2 \implies L(D_0) = L(D_1) = \{w\texttt{b} \mid w \in \{\texttt{a}, \texttt{b}\}^*\}$$

Let's test the equivalence of D_2 and D_3:

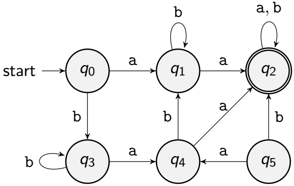
$D_2 =$

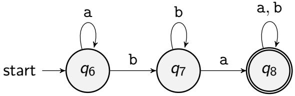


$D_3 =$



Let's test the equivalence of D_2 and D_3:

$D_2 =$ 

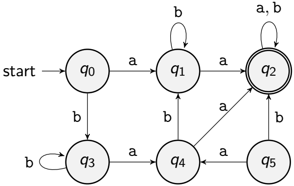
$D_3 =$ 

Let's perform the **table-filling algorithm**:

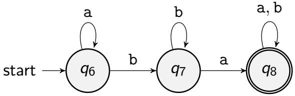
Equivalence of Finite Automata – Example 2

Let's test the equivalence of D_2 and D_3:

$D_2 =$



$D_3 =$



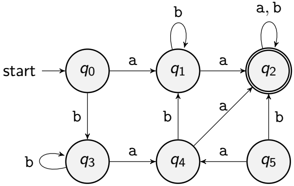
Let's perform the **table-filling algorithm**:

q_1	$\times$							
q_2	$\times$	$\times$						
q_3		$\times$	$\times$					
q_4	$\times$		$\times$	$\times$				
q_5	$\times$	$\times$	$\times$	$\times$	$\times$			
q_6	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$		
q_7	$\times$		$\times$	$\times$		$\times$	$\times$	
q_8	$\times$	$\times$		$\times$	$\times$	$\times$	$\times$	$\times$
	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7

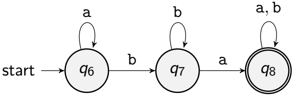
- $q_0 \equiv q_3$
- $q_1 \equiv q_4 \equiv q_7$
- $q_2 \equiv q_8$
- q_5
- q_6

Let's test the equivalence of D_2 and D_3:

$D_2 =$



$D_3 =$



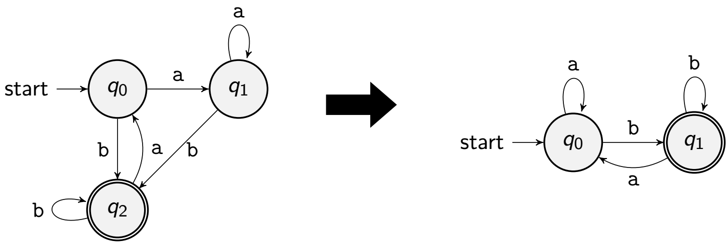
Let's perform the **table-filling algorithm**:

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_1	x							
q_2	x	x						
q_3		x	x					
q_4	x		x	x				
q_5	x	x	x	x	x			
q_6	x	x	x	x	x	x		
q_7	x		x	x		x	x	
q_8	x	x		x	x	x	x	x

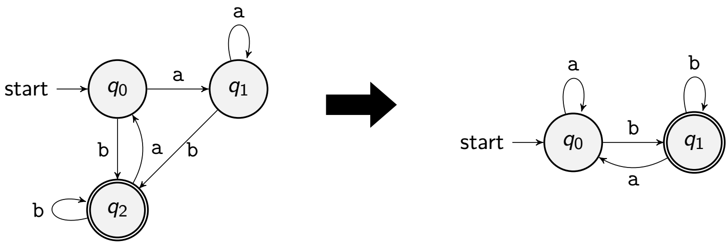
- $q_0 \equiv q_3$
- $q_1 \equiv q_4 \equiv q_7$
- $q_2 \equiv q_8$
- q_5
- q_6

$$q_0 \not\equiv q_6 \implies L(D_2) \neq L(D_3) \ (\because \texttt{ba} \notin L(D_2) \text{ but } \texttt{ba} \in L(D_3))$$

- Is it possible to **minimize** a DFA?



- Is it possible to **minimize** a DFA?



- Yes, let's utilize **equivalence classes** $Q/_{\equiv}$ of states defined with $\equiv$.
- Note that $\equiv$ is an **equivalence relation**:
  - reflexive: $\forall q \in Q.\ q \equiv q$
  - symmetric: $\forall q, q' \in Q.\ q \equiv q' \Leftrightarrow q' \equiv q$
  - transitive: $\forall q, q', q'' \in Q.\ q \equiv q' \land q' \equiv q'' \Leftrightarrow q \equiv q''$

Minimization Algorithm

For a given DFA $D = (Q, \sigma, \delta, q_0, F)$, the **minimization** algorithm is:

1. Remove all **unreachable states** from the initial state q_0.
2. Partition the remaining states into **equivalence classes**:

$$Q/_{\equiv} = \{[q]_{\equiv} \mid q \in Q\}$$

   where the **equivalence class** of a state q is defined as:

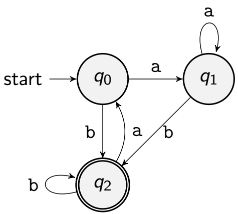
$$[q]_{\equiv} = \{q' \in Q \mid q \equiv q'\}$$

3. Construct a new DFA $D/_{\equiv} = (Q/_{\equiv}, \Sigma, \delta/_{\equiv}, [q_0]_{\equiv}, F/_{\equiv})$ where
   - $\delta/_{\equiv} : Q/_{\equiv} \times \Sigma \to Q/_{\equiv}$ is defined by:

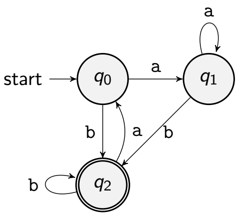
$$\forall q \in Q.\ \forall a \in \Sigma.\ \delta/_{\equiv}([q]_{\equiv}, a) = [\delta(q, a)]_{\equiv}$$

     (We can prove $\forall q', q'' \in [q]_{\equiv}.\ \forall a \in \Sigma.\ [\delta_{\equiv}(q', a)]_{\equiv} = [\delta_{\equiv}(q'', a)]_{\equiv}$.)
   - $F/_{\equiv} = \{[q]_{\equiv} \mid q \in F\}$

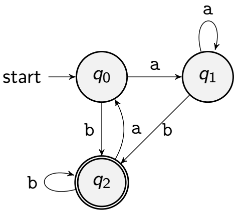
Minimization Algorithm - Example 1



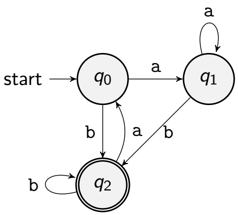
Minimization Algorithm - Example 1



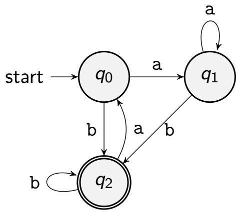
① Remove unreachable states



Minimization Algorithm - Example 1



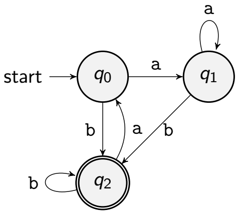
① Remove unreachable states



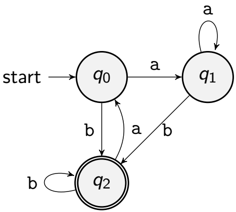
② Partition the states into $Q/_{\equiv}$

$$
\begin{aligned}
Q/_{\equiv} = \{ & \\
& \{q_0, q_1\}, \quad (\because q_0 \equiv q_1) \\
& \{q_2\}, \\
\} &
\end{aligned}
$$

Minimization Algorithm - Example 1



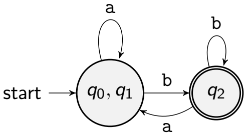
① Remove unreachable states



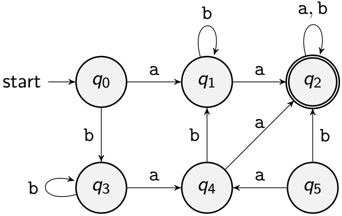
② Partition the states into $Q/_\equiv$

$$
\begin{aligned}
Q/_\equiv = \{ & \\
& \{q_0, q_1\}, \quad (\because q_0 \equiv q_1) \\
& \{q_2\}, \\
\} &
\end{aligned}
$$

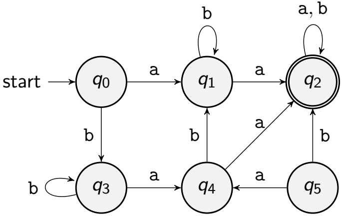
③ Construct a new DFA $D/_\equiv$



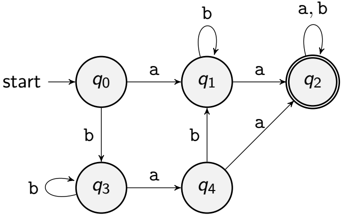
Minimization Algorithm - Example 2



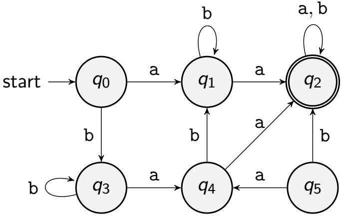
Minimization Algorithm - Example 2



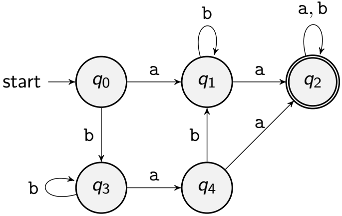
① Remove unreachable states



Minimization Algorithm - Example 2



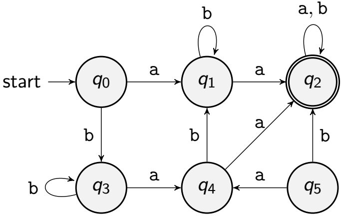
① Remove unreachable states



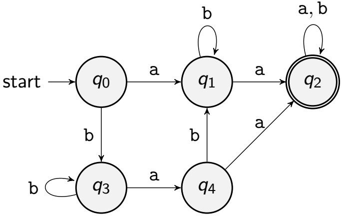
② Partition the states into $Q/_\equiv$

$$
\begin{aligned}
Q/_\equiv = \{ & \\
& \{q_0, q_3\}, \quad (\because q_0 \equiv q_3) \\
& \{q_1, q_4\}, \quad (\because q_1 \equiv q_4) \\
& \{q_2\}, \\
\} &
\end{aligned}
$$

Minimization Algorithm - Example 2



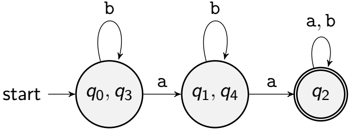
① Remove unreachable states



② Partition the states into $Q/_{\equiv}$

$$
\begin{aligned}
Q/_{\equiv} = \{ & \\
& \{q_0, q_3\}, \quad (\because q_0 \equiv q_3) \\
& \{q_1, q_4\}, \quad (\because q_1 \equiv q_4) \\
& \{q_2\}, \\
\} &
\end{aligned}
$$

③ Construct a new DFA $D/_{\equiv}$



Proof of Minimum-State DFA

PLRG

Theorem (Minimum-State DFA)

For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$*, its minimized DFA* $D/_{\equiv}$ *is a* **minimum-state DFA** *of* D*.*
(i.e., $\nexists$ *DFA* $D' = (Q', \Sigma, \delta', q'_0, F')$*. s.t.* $L(D') = L(D) \land |Q'| < |Q/_{\equiv}|$*).*

Proof of Minimum-State DFA

Theorem (Minimum-State DFA)

For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$*, its minimized DFA* $D/_{\equiv}$ *is a* **minimum-state DFA** *of* D*.*
(i.e., $\nexists$ *DFA* $D' = (Q', \Sigma, \delta', q'_0, F')$*. s.t.* $L(D') = L(D) \land |Q'| < |Q/_{\equiv}|$*).*

- Assume that $\exists$ DFA D'. Then, $m < n$ when $m = |Q'|$ and $n = |Q/_{\equiv}|$.

Proof of Minimum-State DFA

PLRG

Theorem (Minimum-State DFA)

For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$, its minimized DFA $D/_\equiv$ is a **minimum-state DFA** *of D.*
(i.e., $\nexists$ DFA $D' = (Q', \Sigma, \delta', q'_0, F')$. s.t. $L(D') = L(D) \wedge |Q'| < |Q/_\equiv|$).

- Assume that $\exists$ DFA D'. Then, $m < n$ when $m = |Q'|$ and $n = |Q/_\equiv|$.
- For any state $q \in Q/_\equiv$, we can find a state $q' \in Q'$ such that $q \equiv q'$.

Proof of Minimum-State DFA

Theorem (Minimum-State DFA)

For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$, its minimized DFA $D/_{\equiv}$ is a **minimum-state DFA** *of D.*
(i.e., $\nexists$ DFA $D' = (Q', \Sigma, \delta', q'_0, F')$. s.t. $L(D') = L(D) \wedge |Q'| < |Q/_{\equiv}|$).

- Assume that $\exists$ DFA D'. Then, $m < n$ when $m = |Q'|$ and $n = |Q/_{\equiv}|$.
- For any state $q \in Q/_{\equiv}$, we can find a state $q' \in Q'$ such that $q \equiv q'$.
  - $\forall q \in Q/_{\equiv}.\ \exists w = a_1 \cdots a_k.$ s.t. $\delta/_{\equiv}(q_0, w) = q$. ($\because$ q is reachable.)
  - Let $q' = \delta'(q'_0, w)$. Then, $\delta'^*(q'_0, a_0 \cdots a_i) \equiv \delta/_{\equiv}{}^*(q_0, a_0 \cdots a_i)$ for all $0 \leq i \leq k$.
    - **(Basis Case)** $\delta'^*(q'_0, \epsilon) = q'_0 \equiv q_0 = \delta/_{\equiv}{}^*(q_0, \epsilon)$
    - **(Induction Case)** Assume $\delta'^*(q'_0, a_0 \cdots a_i) \not\equiv \delta/_{\equiv}{}^*(q_0, a_0 \cdots a_i)$. Then, by the definition of distinguishable states, $\delta'^*(q'_0, a_0 \cdots a_{i-1}) \not\equiv \delta/_{\equiv}{}^*(q_0, a_0 \cdots a_{i-1})$. But, it contradicts the induction hypothesis.
- By Pigeonhole Principle, $\exists q_i \neq q_j \in Q/_{\equiv}.\ \exists q' \in Q'.\ q_i \equiv q' \wedge q_j \equiv q'$.

Proof of Minimum-State DFA

Theorem (Minimum-State DFA)

For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$, its minimized DFA $D/_{\equiv}$ is a **minimum-state DFA** *of D.*
(i.e., $\nexists$ DFA $D' = (Q', \Sigma, \delta', q'_0, F')$. s.t. $L(D') = L(D) \land |Q'| < |Q/_{\equiv}|$).

- Assume that $\exists$ DFA D'. Then, $m < n$ when $m = |Q'|$ and $n = |Q/_{\equiv}|$.
- For any state $q \in Q/_{\equiv}$, we can find a state $q' \in Q'$ such that $q \equiv q'$.
    - $\forall q \in Q/_{\equiv}.\ \exists w = a_1 \cdots a_k.$ s.t. $\delta/_{\equiv}(q_0, w) = q$. ($\because$ q is reachable.)
    - Let $q' = \delta'(q'_0, w)$. Then, ${\delta'}^*(q'_0, a_0 \cdots a_i) \equiv {\delta/_{\equiv}}^*(q_0, a_0 \cdots a_i)$ for all $0 \leq i \leq k$.
        - **(Basis Case)** ${\delta'}^*(q'_0, \epsilon) = q'_0 \equiv q_0 = {\delta/_{\equiv}}^*(q_0, \epsilon)$
        - **(Induction Case)** Assume ${\delta'}^*(q'_0, a_0 \cdots a_i) \not\equiv {\delta/_{\equiv}}^*(q_0, a_0 \cdots a_i)$. Then, by the definition of distinguishable states, ${\delta'}^*(q'_0, a_0 \cdots a_{i-1}) \not\equiv {\delta/_{\equiv}}^*(q_0, a_0 \cdots a_{i-1})$. But, it contradicts the induction hypothesis.
- By Pigeonhole Principle, $\exists q_i \neq q_j \in Q/_{\equiv}.\ \exists q' \in Q'.\ q_i \equiv q' \land q_j \equiv q'$.
- It means that $q_i \equiv q_j$. However, it contradicts that $Q/_{\equiv}$ is partitioned into equivalence classes of states. □

Summary

1. Equivalence of Finite Automata
   - Equivalence of States ($\equiv$)
   - Distinguishable States ($\not\equiv$)
   - Table-Filling Algorithm
   - Equivalence of Finite Automata
     - Examples

2. Minimization of Finite Automata
   - Minimization Algorithm
     - Examples
   - Proof of Minimum-State DFA

Next Lecture

- Context-Free Grammars (CFGs) and Languages (CFLs)

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr