

Lecture 10 – Equivalence and Minimization of Finite Automata

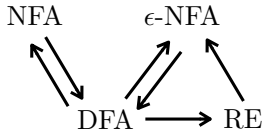
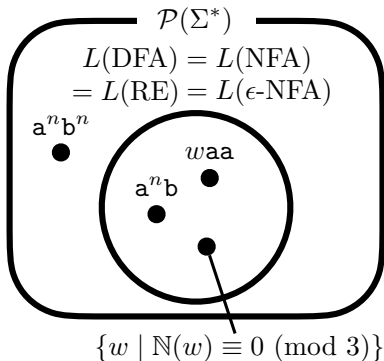
COSE215: Theory of Computation

Jihyeok Park



2026 Spring

- 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?

1. Equivalence of Finite Automata

Equivalence of States (\equiv)

Distinguishable States (\neq)

Table-Filling Algorithm

Equivalence of Finite Automata

Examples

2. Minimization of Finite Automata

Minimization Algorithm

Examples

Proof of Minimum-State DFA

1. Equivalence of Finite Automata

Equivalence of States (\equiv)

Distinguishable States (\neq)

Table-Filling Algorithm

Equivalence of Finite Automata

Examples

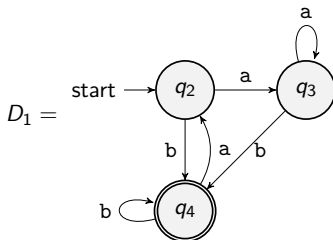
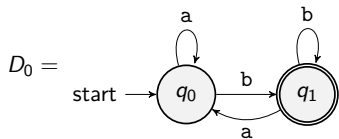
2. Minimization of Finite Automata

Minimization Algorithm

Examples

Proof of Minimum-State DFA

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

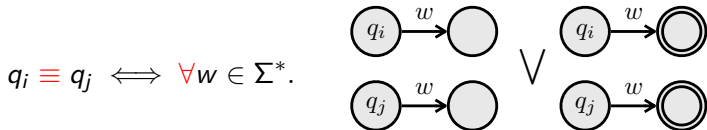


- Yes, because $L(D_0) = L(D_1) = \{wb \mid w \in \{a, b\}^*\}$.
- We first define the **equivalence of states** and utilize it to test the **equivalence of DFA**.

Definition (Equivalence of States (\equiv))

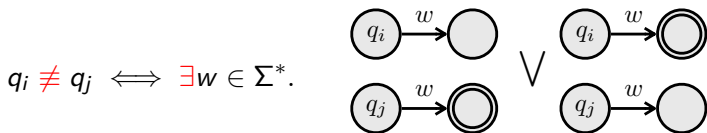
For a given DFA D , q_i is **equivalent** to q_j (i.e., $q_i \equiv q_j$) if and only if

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



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

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

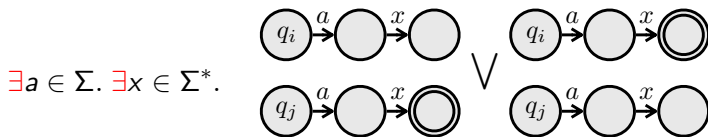


We can *inductively* test q_i is **distinguishable** with q_j (i.e., $q_i \neq q_j$):

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

$$\begin{aligned}
 & (q_i \text{ is not final} \wedge q_j \text{ is final}) \vee (q_i \text{ is final} \wedge q_j \text{ is not final}) \\
 & \quad (\delta^*(q_i, \epsilon) \in F \iff \delta^*(q_j, \epsilon) \notin F) \\
 \iff & \quad (q_i \in F \iff q_j \notin F)
 \end{aligned}$$

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



$$\exists a \in \Sigma. \exists x \in \Sigma^*. (\delta^*(q_i, ax) \in F \iff \delta^*(q_j, ax) \notin F)$$

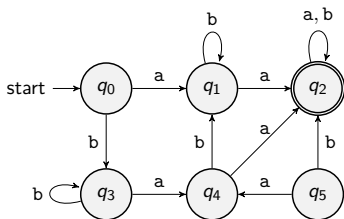
$$\iff \exists a \in \Sigma. \exists x \in \Sigma^*. (\delta^*(\delta(q_i, a), x) \in F \iff \delta^*(\delta(q_j, a), x) \notin F)$$

$$\iff \exists a \in \Sigma. \delta(q_i, a) \neq \delta(q_j, a)$$

Definition (Distinguishable States (\neq))

For a given DFA D , q_i is **distinguishable** with q_j (i.e., $q_i \neq q_j$) iff

- **(Basis Case)** $q_i \in F \iff q_j \notin F$.
- **(Induction Case)** $\exists a \in \Sigma. \delta(q_i, a) \neq \delta(q_j, a)$.



$$q_2 \neq q_4$$

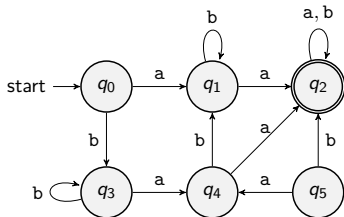
$$(\because q_2 \in F \wedge q_4 \notin F)$$

$$q_1 \neq q_3$$

$$(\because \delta(q_1, a) = q_2 \neq q_4 = \delta(q_3, a))$$

$$q_0 \neq q_4$$

$$(\because \delta(q_0, b) = q_3 \neq q_1 = \delta(q_4, b))$$



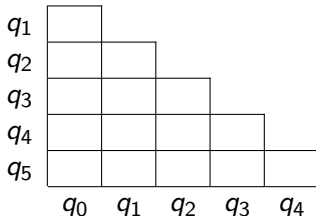
q	a	b
→ q ₀	q ₁	q ₃
q ₁	q ₂	q ₁
*q ₂	q ₂	q ₂
q ₃	q ₄	q ₃
q ₄	q ₂	q ₁
q ₅	q ₄	q ₂

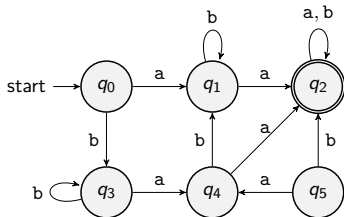
(Basis case) $w = \epsilon$.

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

(Induction case) $w = ax$.

$$\exists a \in \Sigma. \delta(q_i, a) \neq \delta(q_j, a)$$





q	a	b
→ 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

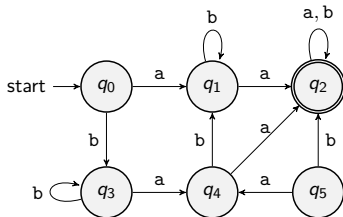
(Basis case) $w = \epsilon$.

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

(Induction case) $w = ax$.

$$\exists a \in \Sigma. \delta(q_i, a) \neq \delta(q_j, a)$$

q ₁					
q ₂	x	x			
q ₃			x		
q ₄			x		
q ₅			x		
	q ₀	q ₁	q ₂	q ₃	q ₄



q	a	b
→ 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) $w = ax$.

$$\exists a \in \Sigma. \delta(q_i, a) \neq \delta(q_j, a)$$

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

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

Theorem (Equivalence of Finite Automata)

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

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

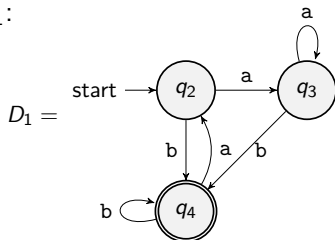
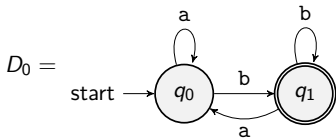
in a DFA $D'' = (Q \uplus Q', \Sigma, \delta'', q_0, F \uplus F')$ where

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

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

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

Let's test the equivalence of D_0 and D_1 :



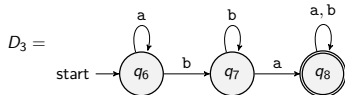
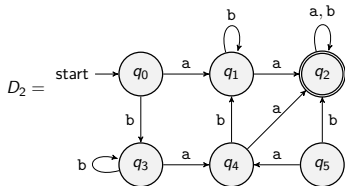
Let's perform the **table-filling algorithm**:

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

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

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

Let's test the equivalence of D_2 and D_3 :



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

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	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

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

1. Equivalence of Finite Automata

Equivalence of States (\equiv)

Distinguishable States (\neq)

Table-Filling Algorithm

Equivalence of Finite Automata

Examples

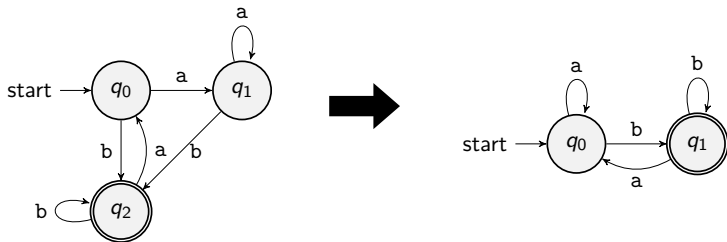
2. Minimization of Finite Automata

Minimization Algorithm

Examples

Proof of Minimum-State 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: $\forall q, q' \in Q. q \equiv q' \implies q' \equiv q$
- transitive: $\forall q, q', q'' \in Q. q \equiv q' \wedge q' \equiv q'' \implies q \equiv q''$

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

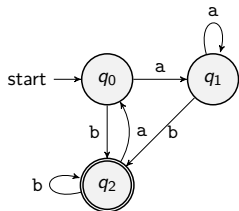
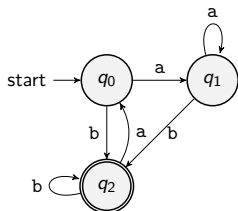
- $\delta/\equiv : Q/\equiv \times \Sigma \rightarrow Q/\equiv$ is defined by:

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

(We can prove $\forall q', q'' \in [q]_{\equiv}. \forall a \in \Sigma. [\delta(q', a)]_{\equiv} = [\delta(q'', a)]_{\equiv}$.)

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

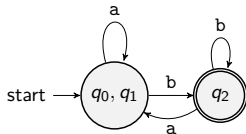
① Remove unreachable states

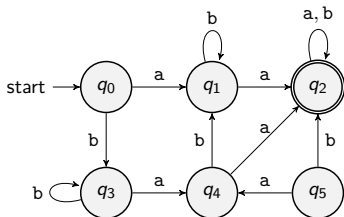


② Partition the states into Q/\equiv

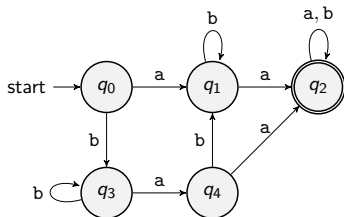
$$Q/\equiv = \left\{ \begin{array}{l} \{q_0, q_1\}, \quad (\because q_0 \equiv q_1) \\ \{q_2\}, \end{array} \right\}$$

③ Construct a new DFA D/\equiv





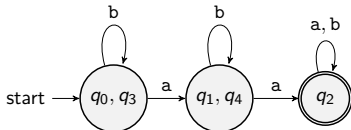
① Remove unreachable states



② Partition the states into Q/\equiv

$$Q/\equiv = \left\{ \begin{array}{l} \{q_0, q_3\}, \quad (\because q_0 \equiv q_3) \\ \{q_1, q_4\}, \quad (\because q_1 \equiv q_4) \\ \{q_2\}, \\ \end{array} \right\}$$

③ Construct a new DFA D/\equiv



Theorem (Minimum-State DFA)

For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$, its minimized DFA $D/\equiv = (Q/\equiv, \Sigma, \delta/\equiv, [q_0]_{\equiv}, F/\equiv)$ 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 $q \equiv q'$.
(We will prove it as a lemma in the next slide.)
- By Pigeonhole Principle, $\exists q_i \neq q_j \in Q/\equiv. \exists q' \in Q'. q_i \equiv q' \wedge q_j \equiv q'$.
- It means that $q_i \equiv q_j$. However, it contradicts that Q/\equiv is partitioned into equivalence classes of states. □

Lemma

Consider a given DFA $D = (Q, \Sigma, \delta, q_0, F)$. Then, let

- $D/\equiv = (Q/\equiv, \Sigma, \delta/\equiv, [q_0]_{\equiv}, F/\equiv)$ be its minimized DFA
- $D' = (Q', \Sigma, \delta', q'_0, F')$ be another DFA such that $L(D) = L(D')$

Then, for any state $q \in Q/\equiv$, we can find a state $q' \in Q'$ such that $q \equiv q'$.

First, $\delta'^*(q'_0, a_1 \cdots a_i) \equiv \delta/\equiv^*([q_0]_{\equiv}, a_1 \cdots a_i)$ for all $0 \leq i \leq k$.

- **(Basis Case)** $\delta'^*(q'_0, \epsilon) = q'_0 \equiv [q_0]_{\equiv} = \delta/\equiv^*([q_0]_{\equiv}, \epsilon)$
($\because L(D') = L(D/\equiv)$)
- **(Induction Case)** Assume $\delta'^*(q'_0, a_1 \cdots a_i) \not\equiv \delta/\equiv^*([q_0]_{\equiv}, a_1 \cdots a_i)$.

Then, by the definition of distinguishable states,

$$\delta'^*(q'_0, a_1 \cdots a_{i-1}) \not\equiv \delta/\equiv^*([q_0]_{\equiv}, a_1 \cdots a_{i-1}).$$

But, it contradicts the induction hypothesis. □

Let $w \in L(D/\equiv)$ s.t. $\delta/\equiv([q_0]_{\equiv}, w) = q$ and $q' = \delta'(q'_0, w)$.

Then $q' \equiv q$ by the above induction □

1. Equivalence of Finite Automata

Equivalence of States (\equiv)

Distinguishable States (\neq)

Table-Filling Algorithm

Equivalence of Finite Automata

Examples

2. Minimization of Finite Automata

Minimization Algorithm

Examples

Proof of Minimum-State DFA

- Please see this document for the exercise.

<https://github.com/ku-plrg-classroom/docs/tree/main/cose215/dfa-eq-min>

- Please implement the following functions in `Implementation.scala`.
 - `nonEqPairs` for the **table-filling algorithm**.
 - `isEqual` for the **equivalence** of DFAs.
 - `minimize` for the **minimization** of DFAs.
- It is just an exercise, and you **don't need to submit** anything.

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

Jihyeok Park

jihyeok_park@korea.ac.kr

<https://plrg.korea.ac.kr>