

Lecture 7 – Equivalence of Regular Expressions and Finite Automata

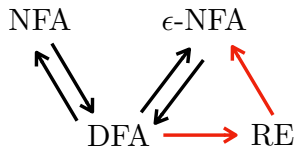
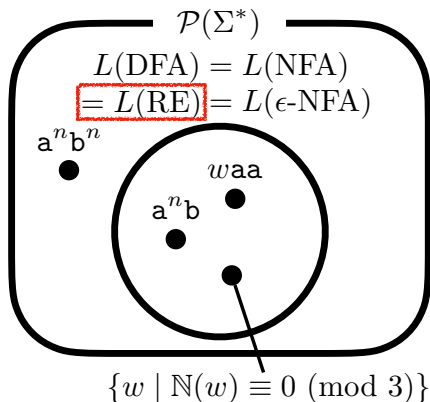
COSE215: Theory of Computation

Jihyeok Park



2026 Spring

- Regular Expressions
 - Operations in languages
 - Definition
 - Precedence order
 - Language of regular expressions
 - Extended regular expressions
 - Examples
- Regular Expressions in Practice



1. Regular Expressions to ϵ -NFA

2. DFA to Regular Expressions

Inductive Construction of Regular Expressions

State Elimination Method

1. Regular Expressions to ϵ -NFA

2. DFA to Regular Expressions

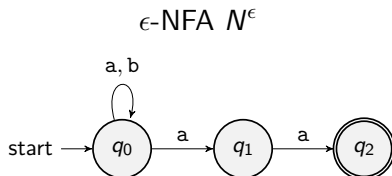
Inductive Construction of Regular Expressions

State Elimination Method

Theorem (Regular Expressions to ϵ -NFA)

For a given regular expression R , $\exists \epsilon$ -NFA N^ϵ . $L(R) = L(N^\epsilon)$.

$(a|b)^*aa$

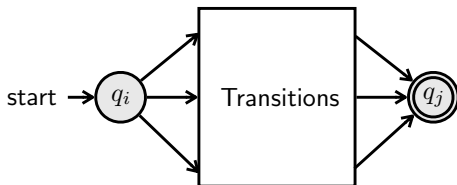


For a given regular expression R and an integer i , we will construct an ϵ -NFA $N^\epsilon = (Q, \Sigma, \delta, q_i, F)$ that accepts the language of R .

For a given regular expression R and an integer i , we will construct an ϵ -NFA $N^\epsilon = (Q, \Sigma, \delta, q_i, F)$ that accepts the language of R .

It satisfies the following properties:

- States are q_i, q_{i+1}, \dots , and q_j ($Q = \{q_k \mid i \leq k \leq j\}$)
- The last state is the unique final state ($F = \{q_j\}$)
- No transition to the initial state ($\forall q \in Q. \forall a \in \Sigma \cup \{\epsilon\}. q_i \notin \delta(q, a)$)
- No transition from the final state ($\forall a \in \Sigma \cup \{\epsilon\}. \delta(q_j, a) = \emptyset$)



ϵ -NFA for (R, i)

For a given regular expression R and an integer i , the ϵ -NFA for (R, i) is:

- $R = \emptyset$:

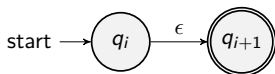


For a given regular expression R and an integer i , the ϵ -NFA for (R, i) is:

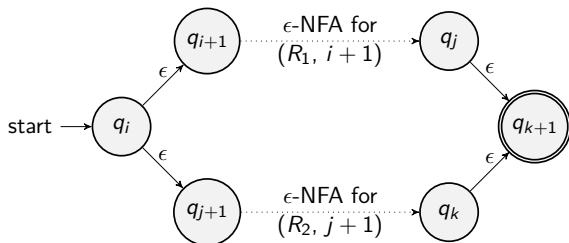
- $R = \emptyset$:



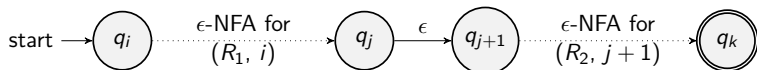
- $R = \epsilon$:



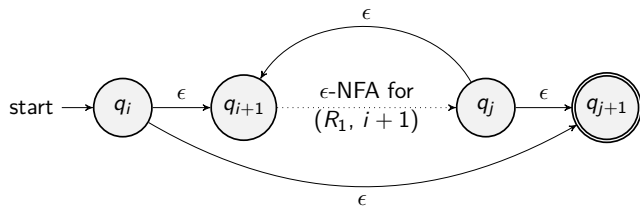
- $R = R_1 \mid R_2$:



- $R = R_1 R_2$:

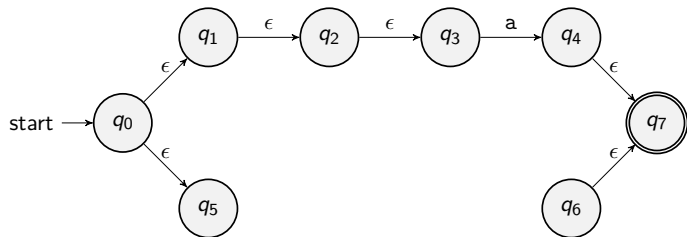


- $R = R_1^*$:



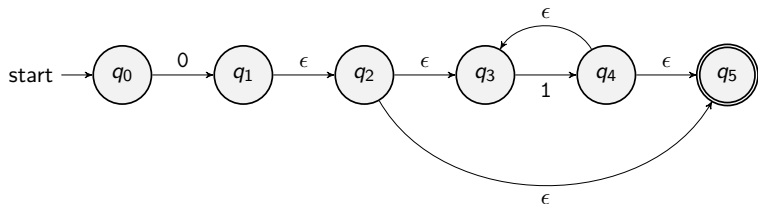
- $R = \epsilon a | \emptyset$

- $R = \epsilon a | \emptyset$



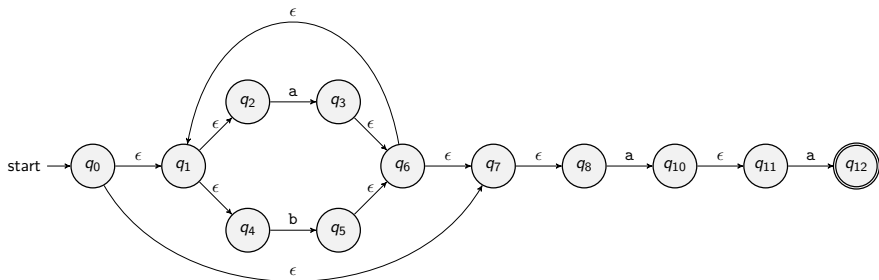
- $R = 01^*$

- $R = 01^*$



- $R = (a|b)^*aa$

- $R = (a|b)^*aa$



1. Regular Expressions to ϵ -NFA

2. DFA to Regular Expressions

Inductive Construction of Regular Expressions

State Elimination Method

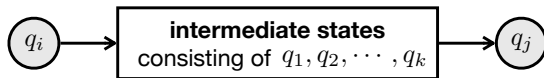
Theorem (DFA to Regular Expressions)

For a given DFA $D = (Q, \Sigma, \delta, q_1, F)$, $\exists RE R. L(D) = L(R)$.

We will learn two different way to convert a DFA to a regular expression.

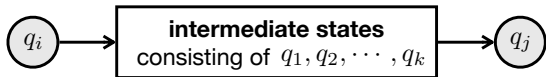
- 1 **Inductive Construction of Regular Expressions** for paths in a DFA with bounded intermediate states where $Q = \{q_1, q_2, \dots, q_n\}$.
- 2 **State Elimination Method** in an extended DFA using regular expressions as labels.

Let $R_{i,j}^{(k)}$ be the **regular expression** that accepts the **paths** from q_i to q_j whose indices of the **intermediate states** are **bounded** by k .

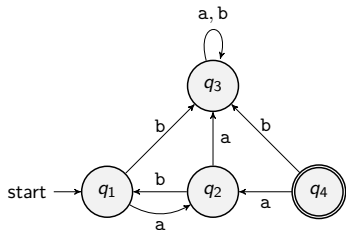


For example, $R_{1,3}^{(2)}$ is the regular expression that accepts the paths from q_1 to q_3 whose intermediate states are q_1 and q_2 .

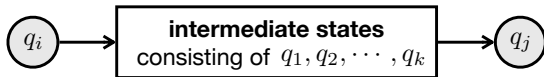
Let $R_{i,j}^{(k)}$ be the **regular expression** that accepts the **paths** from q_i to q_j whose indices of the **intermediate states** are **bounded** by k .



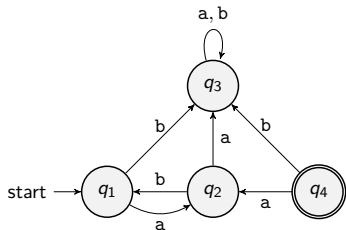
For example, $R_{1,3}^{(2)}$ is the regular expression that accepts the paths from q_1 to q_3 whose intermediate states are q_1 and q_2 .



Let $R_{i,j}^{(k)}$ be the **regular expression** that accepts the **paths** from q_i to q_j whose indices of the **intermediate states** are **bounded** by k .



For example, $R_{1,3}^{(2)}$ is the regular expression that accepts the paths from q_1 to q_3 whose intermediate states are q_1 and q_2 .



$L(R_{1,3}^{(2)}) \ni$ b a a a b a a
 $\not\ni$ a b a a a b

We can **inductively construct** regular expressions $R_{i,j}^{(k)}$ for all combination of i , j , and k (induction on k).

We can **inductively construct** regular expressions $R_{i,j}^{(k)}$ for all combination of i, j , and k (induction on k).

- **(Basis Case)** $k = 0$

It means that **no intermediate states** in the path.

- If $i \neq j$ (source and destination states are different),

$$R_{i,j}^{(0)} = a_1 | a_2 | \cdots | a_m$$

where $q_i \xrightarrow{a_1} q_j, q_i \xrightarrow{a_2} q_j, \cdots, q_i \xrightarrow{a_m} q_j$ are transitions in D .

- If $i = j$ (source and destination states are same),

$$R_{i,j}^{(0)} = R_{i,i}^{(0)} = \epsilon | a_1 | a_2 | \cdots | a_m$$

where $q_i \xrightarrow{a_1} q_i, q_i \xrightarrow{a_2} q_i, \cdots, q_i \xrightarrow{a_m} q_i$ are transitions in D .

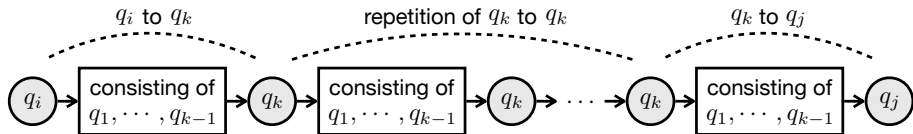
- **(Induction Case)** $R_{i,j}^{(k-1)}$ are given for all i and j .

$$R_{i,j}^{(k)} = R_{i,j}^{(k-1)} \mid R_{i,k}^{(k-1)} (R_{k,k}^{(k-1)})^* R_{k,j}^{(k-1)}$$

- **(Induction Case)** $R_{i,j}^{(k-1)}$ are given for all i and j .

$$R_{i,j}^{(k)} = R_{i,j}^{(k-1)} \mid R_{i,k}^{(k-1)} (R_{k,k}^{(k-1)})^* R_{k,j}^{(k-1)}$$

- $R_{i,j}^{(k-1)}$: paths from q_i to q_j **NOT** containing q_k as intermediate states.
- $R_{i,k}^{(k-1)} (R_{k,k}^{(k-1)})^* R_{k,j}^{(k-1)}$: paths from q_i to q_j containing q_k at least once as intermediate states.



Consider the following DFA:

$$D = (Q, \Sigma, \delta, q_1, F)$$

where $Q = \{q_1, q_2, \dots, q_n\}$ and $F = \{q_{f_1}, q_{f_2}, \dots, q_{f_m}\}$.

Consider the following DFA:

$$D = (Q, \Sigma, \delta, q_1, F)$$

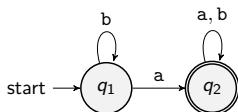
where $Q = \{q_1, q_2, \dots, q_n\}$ and $F = \{q_{f_1}, q_{f_2}, \dots, q_{f_m}\}$.

Then, using the regular expressions $R_{i,j}^{(k)}$ with bounded intermediate states, we can construct the regular expression R that accepts the language of the DFA D as follows:

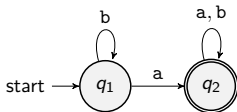
$$R = R_{1,f_1}^{(n)} \mid R_{1,f_2}^{(n)} \mid \dots \mid R_{1,f_m}^{(n)}$$

The regular expression R accepts all the paths from the **initial state** q_1 to one of the **final states** $q_{f_1}, q_{f_2}, \dots, q_{f_m}$ but **no bound** on the intermediate states (because $k = n$).

Let's construct the regular expressions $R_{i,j}^{(k)}$ for the following DFA:



Let's construct the regular expressions $R_{i,j}^{(k)}$ for the following DFA:



When $k = 0$, we have:

- $R_{1,1}^{(0)} = \epsilon | b$
- $R_{1,2}^{(0)} = a$
- $R_{2,1}^{(0)} = \emptyset$
- $R_{2,2}^{(0)} = \epsilon | a | b$

When $k = 0$, we have:

- $R_{1,1}^{(0)} = \epsilon | b$
- $R_{1,2}^{(0)} = a$
- $R_{2,1}^{(0)} = \emptyset$
- $R_{2,2}^{(0)} = \epsilon | a | b$

When $k = 0$, we have:

- $R_{1,1}^{(0)} = \epsilon | b$
- $R_{1,2}^{(0)} = a$
- $R_{2,1}^{(0)} = \emptyset$
- $R_{2,2}^{(0)} = \epsilon | a | b$

When $k = 1$, we have:

- $R_{1,1}^{(1)} = R_{1,1}^{(0)} | R_{1,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,1}^{(0)} = (R_{1,1}^{(0)})^+ = (\epsilon | b)^+ = b^*$

When $k = 0$, we have:

- $R_{1,1}^{(0)} = \epsilon | b$
- $R_{1,2}^{(0)} = a$
- $R_{2,1}^{(0)} = \emptyset$
- $R_{2,2}^{(0)} = \epsilon | a | b$

When $k = 1$, we have:

- $R_{1,1}^{(1)} = R_{1,1}^{(0)} | R_{1,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,1}^{(0)} = (R_{1,1}^{(0)})^+ = (\epsilon | b)^+ = b^*$
- $R_{1,2}^{(1)} = R_{1,2}^{(0)} | R_{1,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,2}^{(0)} = (R_{1,1}^{(0)})^* R_{1,2}^{(0)} = (\epsilon | b)^* a = b^* a$

When $k = 0$, we have:

- $R_{1,1}^{(0)} = \epsilon | b$
- $R_{1,2}^{(0)} = a$
- $R_{2,1}^{(0)} = \emptyset$
- $R_{2,2}^{(0)} = \epsilon | a | b$

When $k = 1$, we have:

- $R_{1,1}^{(1)} = R_{1,1}^{(0)} | R_{1,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,1}^{(0)} = (R_{1,1}^{(0)})^+ = (\epsilon | b)^+ = b^*$
- $R_{1,2}^{(1)} = R_{1,2}^{(0)} | R_{1,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,2}^{(0)} = (R_{1,1}^{(0)})^* R_{1,2}^{(0)} = (\epsilon | b)^* a = b^* a$
- $R_{2,1}^{(1)} = R_{2,1}^{(0)} | R_{2,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,1}^{(0)} = \emptyset | \emptyset = \emptyset$

When $k = 0$, we have:

- $R_{1,1}^{(0)} = \epsilon | b$
- $R_{1,2}^{(0)} = a$
- $R_{2,1}^{(0)} = \emptyset$
- $R_{2,2}^{(0)} = \epsilon | a | b$

When $k = 1$, we have:

- $R_{1,1}^{(1)} = R_{1,1}^{(0)} | R_{1,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,1}^{(0)} = (R_{1,1}^{(0)})^+ = (\epsilon | b)^+ = b^*$
- $R_{1,2}^{(1)} = R_{1,2}^{(0)} | R_{1,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,2}^{(0)} = (R_{1,1}^{(0)})^* R_{1,2}^{(0)} = (\epsilon | b)^* a = b^* a$
- $R_{2,1}^{(1)} = R_{2,1}^{(0)} | R_{2,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,1}^{(0)} = \emptyset | \emptyset = \emptyset$
- $R_{2,2}^{(1)} = R_{2,2}^{(0)} | R_{2,1}^{(0)} (R_{1,1}^{(0)})^* R_{1,2}^{(0)} = R_{2,2}^{(0)} | \emptyset = R_{2,2}^{(0)} = \epsilon | a | b$

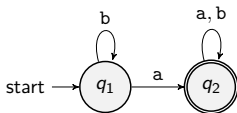
When $k = 1$, we have:

- $R_{1,1}^{(1)} = b^*$
- $R_{1,2}^{(1)} = b^*a$
- $R_{2,1}^{(1)} = \emptyset$
- $R_{2,2}^{(1)} = \epsilon | a | b$

When $k = 1$, we have:

- $R_{1,1}^{(1)} = b^*$
- $R_{1,2}^{(1)} = b^*a$
- $R_{2,1}^{(1)} = \emptyset$
- $R_{2,2}^{(1)} = \epsilon | a | b$

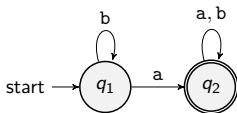
Let's focus on the regular expression for the language of the DFA.



When $k = 1$, we have:

- $R_{1,1}^{(1)} = b^*$
- $R_{1,2}^{(1)} = b^*a$
- $R_{2,1}^{(1)} = \emptyset$
- $R_{2,2}^{(1)} = \epsilon | a | b$

Let's focus on the regular expression for the language of the DFA.



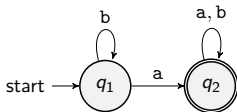
We need to compute $R_{1,2}^{(2)}$ using the results when $k = 1$:

- $R_{1,2}^{(2)} =$

When $k = 1$, we have:

- $R_{1,1}^{(1)} = b^*$
- $R_{1,2}^{(1)} = b^*a$
- $R_{2,1}^{(1)} = \emptyset$
- $R_{2,2}^{(1)} = \epsilon | a | b$

Let's focus on the regular expression for the language of the DFA.



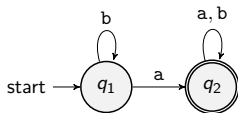
We need to compute $R_{1,2}^{(2)}$ using the results when $k = 1$:

$$\bullet R_{1,2}^{(2)} = R_{1,2}^{(1)} \mid R_{1,2}^{(1)} (R_{2,2}^{(1)})^* R_{2,2}^{(1)} =$$

When $k = 1$, we have:

- $R_{1,1}^{(1)} = b^*$
- $R_{1,2}^{(1)} = b^*a$
- $R_{2,1}^{(1)} = \emptyset$
- $R_{2,2}^{(1)} = \epsilon | a | b$

Let's focus on the regular expression for the language of the DFA.



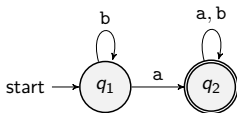
We need to compute $R_{1,2}^{(2)}$ using the results when $k = 1$:

$$\begin{aligned}
 \bullet R_{1,2}^{(2)} &= R_{1,2}^{(1)} \mid R_{1,2}^{(1)} (R_{2,2}^{(1)})^* R_{2,2}^{(1)} = R_{1,2}^{(1)} (R_{2,2}^{(1)})^* \\
 &=
 \end{aligned}$$

When $k = 1$, we have:

- $R_{1,1}^{(1)} = b^*$
- $R_{1,2}^{(1)} = b^*a$
- $R_{2,1}^{(1)} = \emptyset$
- $R_{2,2}^{(1)} = \epsilon | a | b$

Let's focus on the regular expression for the language of the DFA.



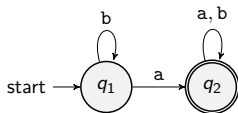
We need to compute $R_{1,2}^{(2)}$ using the results when $k = 1$:

$$\begin{aligned}
 \bullet R_{1,2}^{(2)} &= R_{1,2}^{(1)} \mid R_{1,2}^{(1)} (R_{2,2}^{(1)})^* R_{2,2}^{(1)} = R_{1,2}^{(1)} (R_{2,2}^{(1)})^* \\
 &= b^* a (\epsilon | a | b)^* \\
 &=
 \end{aligned}$$

When $k = 1$, we have:

- $R_{1,1}^{(1)} = b^*$
- $R_{1,2}^{(1)} = b^*a$
- $R_{2,1}^{(1)} = \emptyset$
- $R_{2,2}^{(1)} = \epsilon | a | b$

Let's focus on the regular expression for the language of the DFA.



We need to compute $R_{1,2}^{(2)}$ using the results when $k = 1$:

$$\begin{aligned}
 \bullet R_{1,2}^{(2)} &= R_{1,2}^{(1)} \mid R_{1,2}^{(1)} (R_{2,2}^{(1)})^* R_{2,2}^{(1)} = R_{1,2}^{(1)} (R_{2,2}^{(1)})^* \\
 &= b^* a (\epsilon | a | b)^* \\
 &= b^* a (a | b)^* \quad (\text{the regular expression for the above DFA})
 \end{aligned}$$

- There is another way to convert a DFA to a regular expression.

- There is another way to convert a DFA to a regular expression.
- It is **more intuitive** and easier to understand but **not easy to implement**.

- There is another way to convert a DFA to a regular expression.
- It is **more intuitive** and easier to understand but **not easy to implement**.
- The idea is to **eliminate** the **states** of the DFA one by one and **construct the regular expressions**.

- There is another way to convert a DFA to a regular expression.
- It is **more intuitive** and easier to understand but **not easy to implement**.
- The idea is to **eliminate** the **states** of the DFA one by one and **construct the regular expressions**.
- We will assign constructed **regular expressions** instead of symbols as **labels** on the transitions between the states in the DFA.

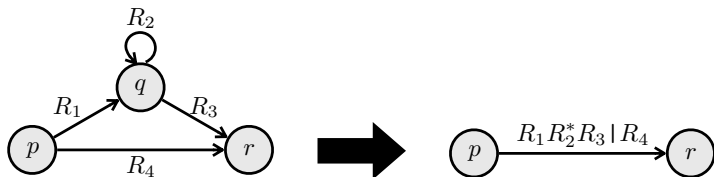
We can convert a DFA to a regular expression using the following steps for **each final state** $q_f \in F$:

We can convert a DFA to a regular expression using the following steps for **each final state** $q_f \in F$:

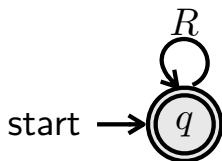
- 1 **Merge** symbols a_1, a_2, \dots, a_m on the transition from q_i to q_j into a single regular expression $a_1 | a_2 | \dots | a_m$.

We can convert a DFA to a regular expression using the following steps for **each final state** $q_f \in F$:

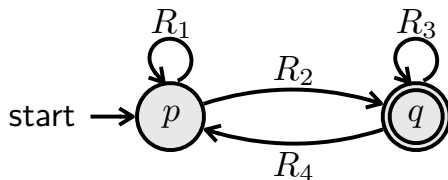
- 1 **Merge** symbols a_1, a_2, \dots, a_m on the transition from q_i to q_j into a single regular expression $a_1 | a_2 | \dots | a_m$.
- 2 **Eliminate** a state q that is **not** the **initial** state or the **target final** state using the following mechanism:



- 3 **Construct** regular expressions for the remaining one or two states using the regular expressions on the transitions between the states.



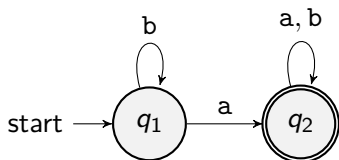
R^*



$(R_1 \mid R_2 R_3^* R_4)^* R_2 R_3^*$

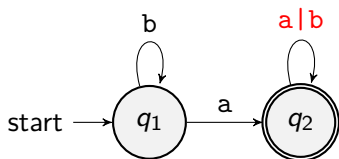
Example 1

Let's convert the following DFA to a regular expression using the **state elimination** method:



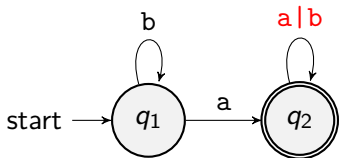
Example 1

Let's convert the following DFA to a regular expression using the **state elimination** method:



Example 1

Let's convert the following DFA to a regular expression using the **state elimination** method:

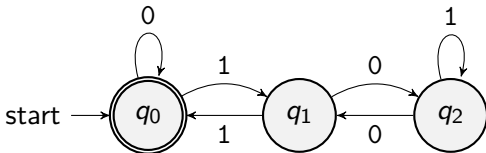


The regular expression for the above DFA is:

$$b^*a(a|b)^*$$

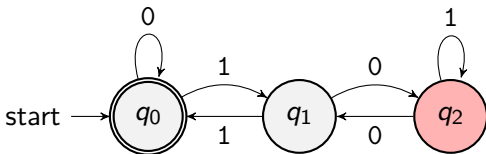
Example 2

The following DFA accepts $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \pmod{3}\}$ where $\mathbb{N}(w)$ is the natural number represented by w in binary:



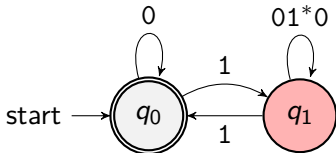
Example 2

The following DFA accepts $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \pmod{3}\}$ where $\mathbb{N}(w)$ is the natural number represented by w in binary:



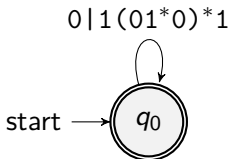
Example 2

The following DFA accepts $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \pmod{3}\}$ where $\mathbb{N}(w)$ is the natural number represented by w in binary:



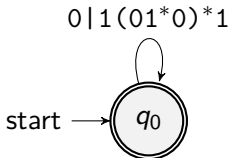
Example 2

The following DFA accepts $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \pmod{3}\}$ where $\mathbb{N}(w)$ is the natural number represented by w in binary:



Example 2

The following DFA accepts $L = \{w \in \{0, 1\}^* \mid \mathbb{N}(w) \equiv 0 \pmod{3}\}$ where $\mathbb{N}(w)$ is the natural number represented by w in binary:

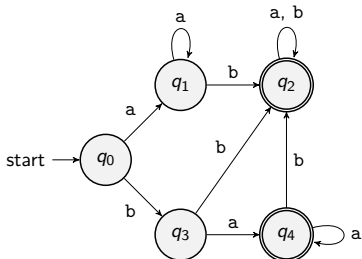


Then, the regular expression for the above DFA is:

$$(0|1(01^*0)^*1)^*$$

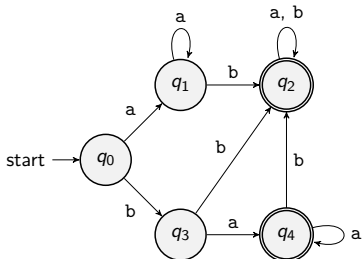
Example 3

Let's convert the following DFA to a regular expression using the **state elimination** method:



Example 3

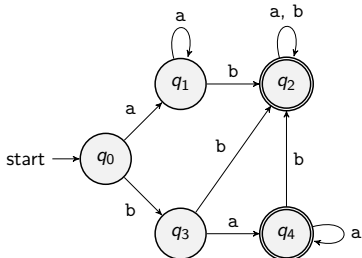
Let's convert the following DFA to a regular expression using the **state elimination** method:



We need to consider two final states q_2 and q_4 .

Example 3

Let's convert the following DFA to a regular expression using the **state elimination** method:

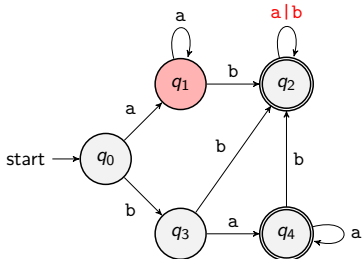


We need to consider two final states q_2 and q_4 .

Let's start by eliminating non-initial and non-final states.

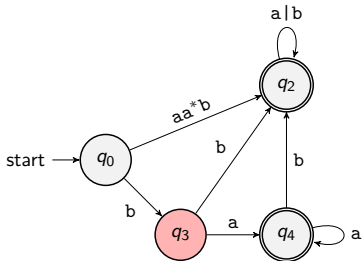
Example 3

Let's convert the following DFA to a regular expression using the **state elimination** method:



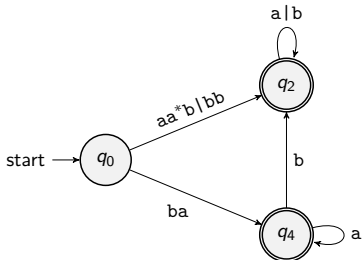
Example 3

Let's convert the following DFA to a regular expression using the **state elimination** method:



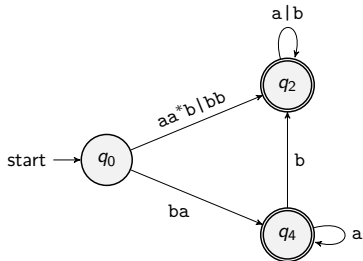
Example 3

Let's convert the following DFA to a regular expression using the **state elimination** method:



Example 3

Let's convert the following DFA to a regular expression using the **state elimination** method:

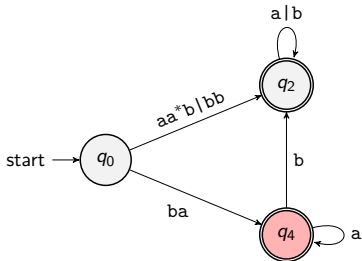


Now, we know that the regular expression for the **final state** q_4 is:

$$baa^*$$

Example 3

Let's convert the following DFA to a regular expression using the **state elimination** method:



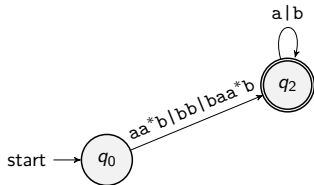
Now, we know that the regular expression for the **final state** q_4 is:

$$baa^*$$

Let's keep eliminating q_4 to know the regular expression for q_2 .

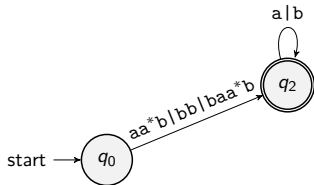
Example 3

Let's convert the following DFA to a regular expression using the **state elimination** method:



Example 3

Let's convert the following DFA to a regular expression using the **state elimination** method:

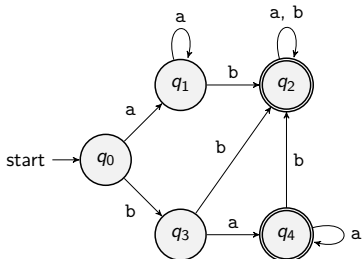


Now, we know that the regular expression for the **final state** q_2 is:

$$(aa^*b|bb|baa^*b)(a|b)^*$$

Example 3

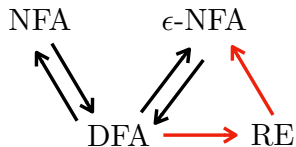
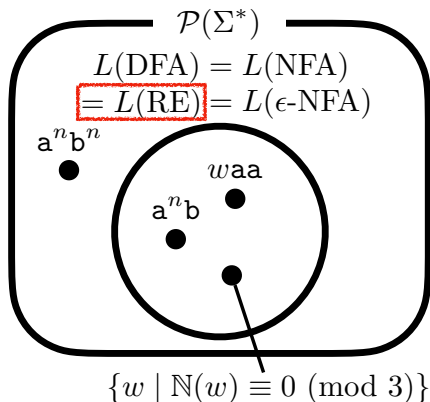
Let's convert the following DFA to a regular expression using the **state elimination** method:



Finally, we have the regular expression for the above DFA:

$$(aa^*b | bb | baa^*b) (a | b)^* | baa^*$$

Note that $(aa^*b | bb | baa^*b) (a | b)^*$ is for q_2 and baa^* is for q_4 .



- Closure Properties of Regular Languages

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