

# Lecture 7 – Equivalence of Regular Expressions and Finite Automata

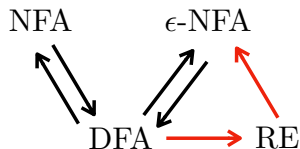
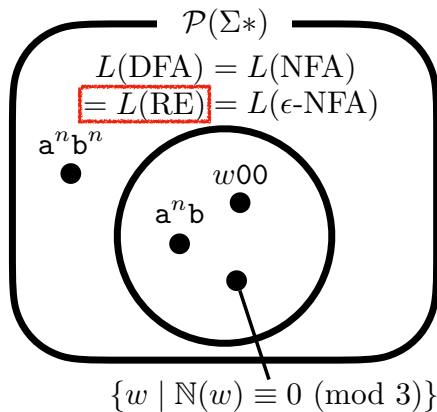
## COSE215: Theory of Computation

Jihyeok Park



2023 Spring

- ① Operations in Languages
  - Union
  - Concatenation
  - Kleene Star
- ② Nondeterministic Finite Automata (NFA)
  - Definition
  - Language of Regular Expressions
  - Extended Regular Expressions
  - Examples



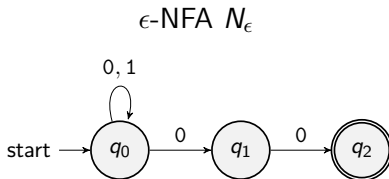
1. Regular Expressions to  $\epsilon$ -NFA

2. DFA to Regular Expressions

## Theorem (Regular Expressions to $\epsilon$ -NFA)

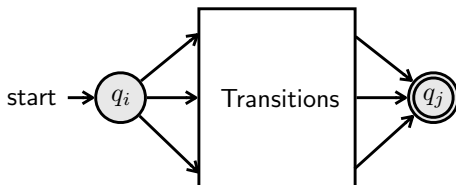
For a given regular expression  $R$ ,  $\exists$   $\epsilon$ -NFA  $N_\epsilon$ .  $L(R) = L(N_\epsilon)$ .

$(0|1)^*00$



For a given regular expression  $R$  and an integer  $i$ , we will construct an  $\epsilon$ -NFA  $N_\epsilon = (Q, \Sigma, \delta, q_i, F)$  that accepts the language of  $R$ . It satisfies the following properties:

- Exactly one final state  $q_j$  for some  $j$  greater than  $i$  ( $F = \{q_j\} \wedge j > i$ )
- States are  $q_i, q_{i+1}, \dots$ , and  $q_j$  ( $Q = \{q_k \mid i \leq k \leq j\}$ )
- No transition to the initial state ( $\forall q \in Q. \forall a \in \Sigma \cup \{\epsilon\}. q_i \notin \delta(q, a)$ )
- No transition out of the final state ( $\forall a \in \Sigma \cup \{\epsilon\}. \delta(q_j, a) = \emptyset$ )



```
// The type definitions of states and symbols
type State = Int
type Symbol = Char

// A transition allowing epsilon
type Transition = (State, Option[Symbol], State)

// A simplified epsilon-NFA
case class SimpleENFA(from: State, trans: Set[Transition], to: State)

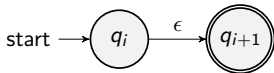
// Convert a regular expression to a simple epsilon-NFA with an initial state
def RE2SimpleENFA(rexp: RE, initSt: State): SimpleENFA = re match
  case REEmpty()           => ???
  case REEpsilon()         => ???
  case RESymbol(symbol)    => ???
  case REUnion(left, right) => ???
  case REConcat(left, right) => ???
  case REStar(re)          => ???
  case REParen(re)         => ???

// Convert a simple epsilon-NFA to an epsilon-NFA
def SimpleENFA2ENFA(senfa: SimpleENFA): ENFA = ...
```

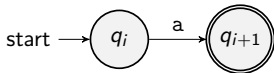
- $R = \emptyset$ :



- $R = \epsilon$ :



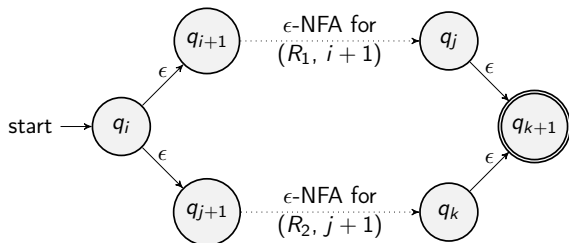
- $R = a$ :



```
case REEmpty()           => SimpleENFA(  
  from = i,  trans = Set(),           to = i + 1,  
)  
case REEpsilon()         => SimpleENFA(  
  from = i,  trans = Set((i, None, i + 1)),  to = i + 1,  
)  
case RESymbol(symbol) => SimpleENFA(  
  from = i,  trans = Set((i, Some(symbol), i + 1)),  to = i + 1,  
)
```

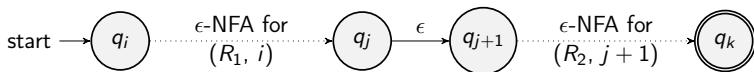


- $R = R_1 \mid R_2$ :

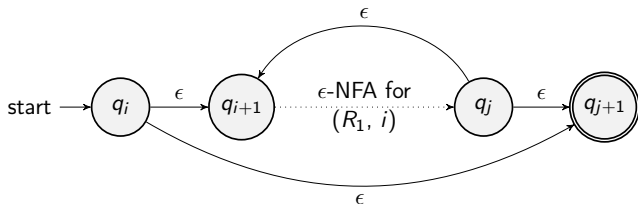


```
case REUnion(left, right) =>
  val SimpleENFA(_, trans1, j) = RE2SimpleENFA(left, i + 1)
  val SimpleENFA(_, trans2, k) = RE2SimpleENFA(right, j + 1)
  SimpleENFA(
    from = i,
    trans = trans1 ++ trans2 ++ Set(
      (i, None, i + 1), (i, None, j + 1),
      (j, None, k + 1), (k, None, k + 1),
    ),
    to = k + 1,
  )
```

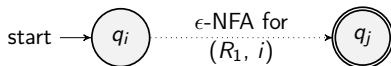
- $R = R_1 \cdot R_2$ :



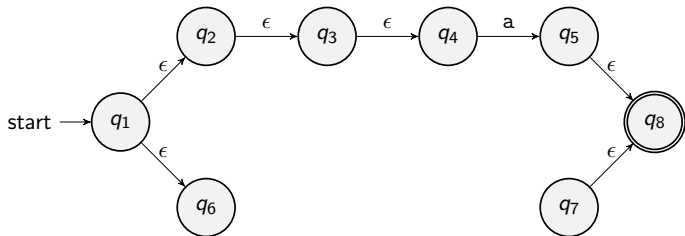
- $R = R_1^*$ :



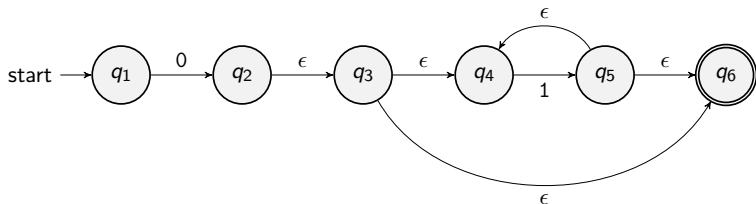
- $R = (R_1)$ :



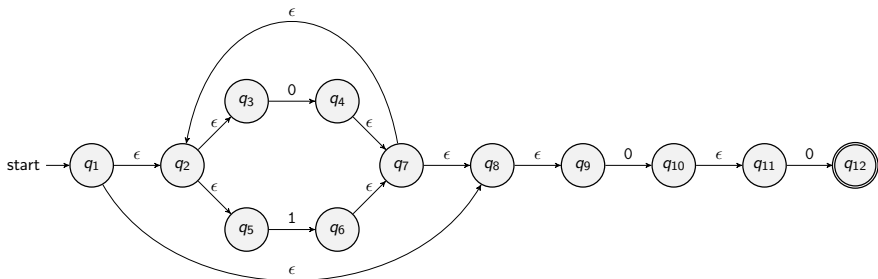
- $R = \epsilon \cdot a | \emptyset$



- $R = 0 \cdot 1^*$



- $R = (0|1)^* \cdot 0 \cdot 0$

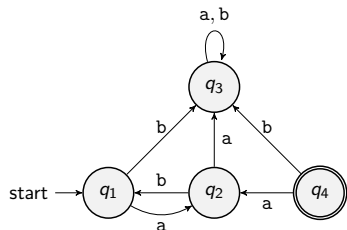
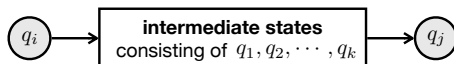


## Theorem (DFA to Regular Expressions)

For a given DFA  $D = (\{q_1, q_2, \dots, q_n\}, \Sigma, \delta, q_1, F)$ ,  $\exists RE R. L(D) = L(R)$ .

Let  $R_{i,j}^{(k)}$  be the regular expression that accepts the paths from  $q_i$  to  $q_j$  whose *intermediate* states are  $q_1, q_2, \dots, q_k$ . Then,

$$R = R_{1,f_1}^{(n)} \mid \cdot \mid R_{1,f_m}^{(n)} \text{ where } F = \{q_{f_1}, q_{f_2}, \dots, q_{f_m}\}$$



$$L(R_{1,3}^{(2)}) \ni \begin{array}{ccccc} b & a & a & a & b & a & a \\ \not\ni & a & b & a & a & a & b \end{array}$$

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

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

- If  $i \neq j$ ,

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

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  $i = j$ ,

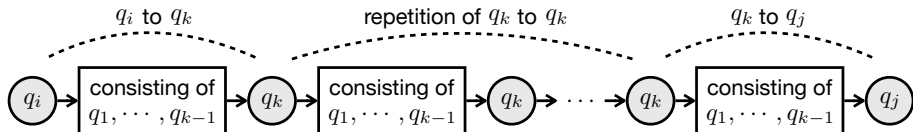
$$R_{i,j}^{(0)} = \epsilon | a_1 | a_2 | \cdots | a_m$$

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  $D$ .

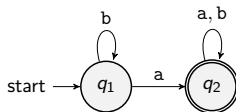
- (Induction Case)  $R_{i,j}^{(k-1)}$  are given for all  $i$  and  $j$ .

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

- $R_{i,j}^{(k-1)}$ : paths from  $q_i$  to  $q_j$  do not use state  $q_k$  at all.
- $R_{i,k}^{(k-1)} (R_{k,k}^{(k-1)})^* R_{k,j}^{(k-1)}$ : paths from  $q_i$  to  $q_j$  use state  $q_k$  at least once.



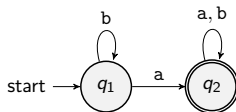




- $R_{1,1}^{(0)} = \epsilon | b$
- $R_{1,2}^{(0)} = a$
- $R_{2,1}^{(0)} = \emptyset$
- $R_{2,2}^{(0)} = \epsilon | a | b$

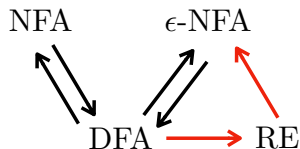
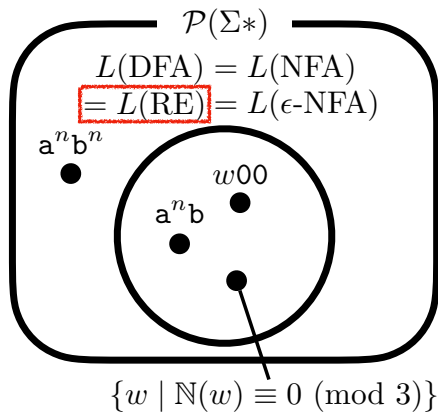
Note that  $(\epsilon | R)^+ = R^*$ ,  $(\epsilon | R)^* = R^*$ ,  $\emptyset \cdot R = \emptyset$ ,  $\emptyset | R = R$

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



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

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



- Properties of Regular Languages

Jihyeok Park

[jihyeok\\_park@korea.ac.kr](mailto:jihyeok_park@korea.ac.kr)

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