

Lecture 5 – ϵ -Nondeterministic Finite Automata (ϵ -NFA)

COSE215: Theory of Computation

Jihyeok Park



2024 Spring

① Deterministic Finite Automata (DFA)

- Definition
- Transition Diagram and Transition Table
- Extended Transition Function
- Acceptance of a Word
- Language of DFA (Regular Language)
- Examples

② Nondeterministic Finite Automata (NFA)

- Definition
- Transition Diagram and Transition Table
- Extended Transition Function
- Language of NFA
- Examples

③ Equivalence of DFA and NFA

- $\text{DFA} \rightarrow \text{NFA}$
- $\text{DFA} \leftarrow \text{NFA}$ (Subset Construction)

1. ϵ -Nondeterministic Finite Automata (ϵ -NFA)

- ϵ -Transition

- Definition

- Transition Diagram and Transition Table

- ϵ -Closures

- Extended Transition Function

- Language of ϵ -NFA

2. Equivalence of DFA and ϵ -NFA

- DFA \leftarrow ϵ -NFA (Subset Construction)

1. ϵ -Nondeterministic Finite Automata (ϵ -NFA)

- ϵ -Transition

- Definition

- Transition Diagram and Transition Table

- ϵ -Closures

- Extended Transition Function

- Language of ϵ -NFA

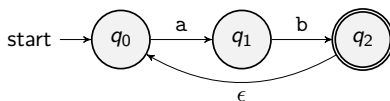
2. Equivalence of DFA and ϵ -NFA

- DFA \leftarrow ϵ -NFA (Subset Construction)

Let's consider ϵ -**transitions** which can be taken **without consuming any input symbol** in nondeterministic finite automata (NFA).

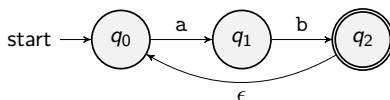
Let's consider ϵ -**transitions** which can be taken **without consuming any input symbol** in nondeterministic finite automata (NFA).

For example, consider an ϵ -**transition** from q_2 to q_0 in the following NFA:



Let's consider ϵ -**transitions** which can be taken **without consuming any input symbol** in nondeterministic finite automata (NFA).

For example, consider an ϵ -**transition** from q_2 to q_0 in the following NFA:

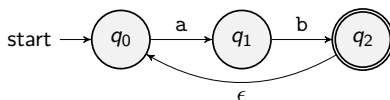


Then, the above automaton **accepts** the following words:

ab abab ababab ...

Let's consider ϵ -**transitions** which can be taken **without consuming any input symbol** in nondeterministic finite automata (NFA).

For example, consider an ϵ -**transition** from q_2 to q_0 in the following NFA:



Then, the above automaton **accepts** the following words:

ab abab ababab ...

Let's formally define ϵ -**NFA**, an extension of NFA with ϵ - transitions.

Definition (ϵ -Nondeterministic Finite Automaton (ϵ -NFA))

An ϵ -**nondeterministic finite automaton** is a 5-tuple:

$$N^\epsilon = (Q, \Sigma, \delta, q_0, F)$$

- Q is a finite set of **states**
- Σ is a finite set of **symbols**
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ is the **transition function**
- $q_0 \in Q$ is the **initial state**
- $F \subseteq Q$ is the set of **final states**

Definition (ϵ -Nondeterministic Finite Automaton (ϵ -NFA))

An ϵ -**nondeterministic finite automaton** is a 5-tuple:

$$N^\epsilon = (Q, \Sigma, \delta, q_0, F)$$

- Q is a finite set of **states**
- Σ is a finite set of **symbols**
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ is the **transition function**
- $q_0 \in Q$ is the **initial state**
- $F \subseteq Q$ is the set of **final states**

$$N_1^\epsilon = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$$

$$\delta(q_0, a) = \{q_1\}$$

$$\delta(q_1, a) = \emptyset$$

$$\delta(q_2, a) = \emptyset$$

$$\delta(q_0, b) = \emptyset$$

$$\delta(q_1, b) = \{q_2\}$$

$$\delta(q_2, b) = \emptyset$$

$$\delta(q_0, \epsilon) = \emptyset$$

$$\delta(q_1, \epsilon) = \emptyset$$

$$\delta(q_2, \epsilon) = \{q_0\}$$

```
// The definition of epsilon-NFA
case class ENFA(
  states: Set[State],
  symbols: Set[Symbol],
  trans: Map[(State, Option[Symbol]), Set[State]],
  initState: State,
  finalStates: Set[State],
)
```

```
// The definition of epsilon-NFA
case class ENFA(
  states: Set[State],
  symbols: Set[Symbol],
  trans: Map[(State, Option[Symbol]), Set[State]],
  initState: State,
  finalStates: Set[State],
)
```

```
// An example of epsilon-NFA
val enfa1: ENFA = ENFA(
  states = Set(0, 1, 2),
  symbols = Set('a', 'b'),
  trans = Map(
    (0, Some('a')) -> Set(1),      // (0, a) -> 1
    (1, Some('b')) -> Set(2),      // (1, b) -> 2
    (2, None)       -> Set(0),      // (2,  $\epsilon$ ) -> 0
  ).withDefaultValue(Set()),
  initState = 0,
  finalStates = Set(2),
)
```

$$N_1^\epsilon = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$$

$$\delta(q_0, a) = \{q_1\}$$

$$\delta(q_1, a) = \emptyset$$

$$\delta(q_2, a) = \emptyset$$

$$\delta(q_0, b) = \emptyset$$

$$\delta(q_1, b) = \{q_2\}$$

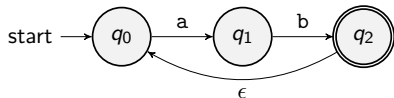
$$\delta(q_2, b) = \emptyset$$

$$\delta(q_0, \epsilon) = \emptyset$$

$$\delta(q_1, \epsilon) = \emptyset$$

$$\delta(q_2, \epsilon) = \{q_0\}$$

Transition Diagram



Transition Table

q	a	b	ϵ
$\rightarrow q_0$	$\{q_1\}$	\emptyset	\emptyset
q_1	\emptyset	$\{q_2\}$	\emptyset
$*q_2$	\emptyset	\emptyset	$\{q_0\}$

Definition (ϵ -Closures)

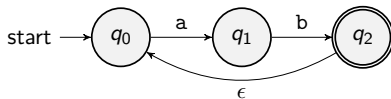
The ϵ -**closure** $\text{EClo}(q)$ for a state q is the set of all reachable states from q defined as follows:

- **(Basis Case)** $q \in \text{EClo}(q)$
- **(Induction Case)** $(q' \in \delta(q, \epsilon) \wedge q'' \in \text{EClo}(q')) \Rightarrow q'' \in \text{EClo}(q)$

Definition (ϵ -Closures)

The ϵ -**closure** $\text{EClo}(q)$ for a state q is the set of all reachable states from q defined as follows:

- **(Basis Case)** $q \in \text{EClo}(q)$
- **(Induction Case)** $(q' \in \delta(q, \epsilon) \wedge q'' \in \text{EClo}(q')) \Rightarrow q'' \in \text{EClo}(q)$

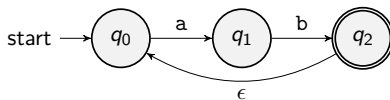


$$\text{EClo}(q_2) = \{q_0, q_2\}$$

Definition (ϵ -Closures)

The ϵ -**closure** $\text{EClo}(q)$ for a state q is the set of all reachable states from q defined as follows:

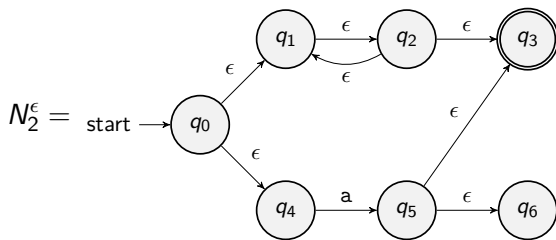
- **(Basis Case)** $q \in \text{EClo}(q)$
- **(Induction Case)** $(q' \in \delta(q, \epsilon) \wedge q'' \in \text{EClo}(q')) \Rightarrow q'' \in \text{EClo}(q)$



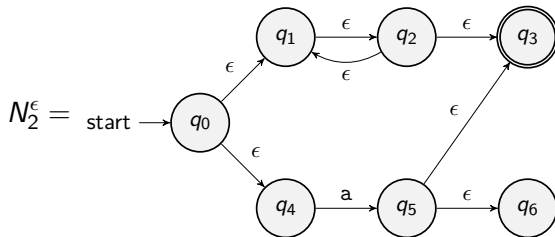
$$\text{EClo}(q_2) = \{q_0, q_2\}$$

We sometimes need to define the ϵ -closure for a **set of states** $S \subseteq Q$:

$$\forall S \subseteq Q. \text{EClo}(S) = \bigcup_{q \in S} \text{EClo}(q)$$

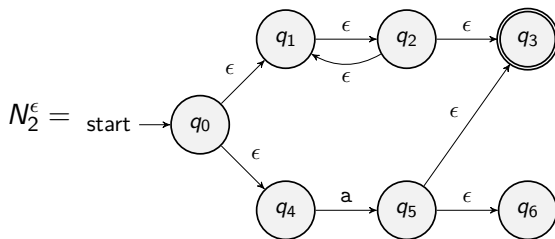


$\text{EClo}(q_0) =$



$$\text{EClo}(q_0) = \{q_0, q_1, q_2, q_3, q_4\}$$

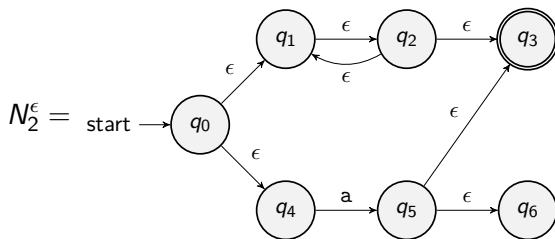
$$\text{EClo}(q_2) =$$



$$\text{EClo}(q_0) = \{q_0, q_1, q_2, q_3, q_4\}$$

$$\text{EClo}(q_2) = \{q_1, q_2, q_3\}$$

$$\text{EClo}(q_5) =$$

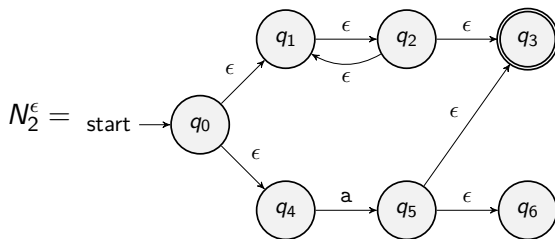


$$\text{EClo}(q_0) = \{q_0, q_1, q_2, q_3, q_4\}$$

$$\text{EClo}(q_2) = \{q_1, q_2, q_3\}$$

$$\text{EClo}(q_5) = \{q_3, q_5, q_6\}$$

$$\text{EClo}(\{q_2, q_5\}) =$$

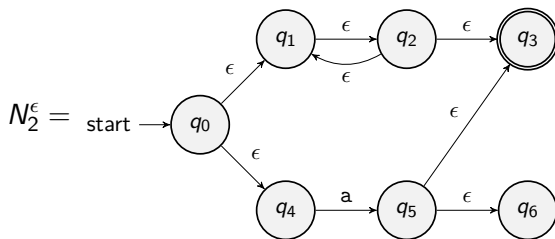


$$\text{EClo}(q_0) = \{q_0, q_1, q_2, q_3, q_4\}$$

$$\text{EClo}(q_2) = \{q_1, q_2, q_3\}$$

$$\text{EClo}(q_5) = \{q_3, q_5, q_6\}$$

$$\begin{aligned} \text{EClo}(\{q_2, q_5\}) &= \text{EClo}(q_2) \cup \text{EClo}(q_5) \\ &= \end{aligned}$$

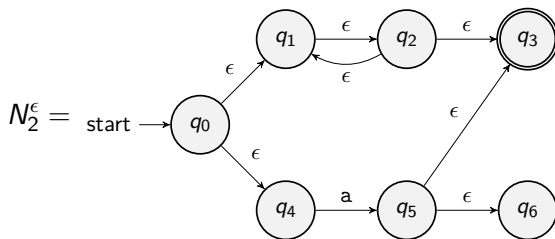


$$\text{EClo}(q_0) = \{q_0, q_1, q_2, q_3, q_4\}$$

$$\text{EClo}(q_2) = \{q_1, q_2, q_3\}$$

$$\text{EClo}(q_5) = \{q_3, q_5, q_6\}$$

$$\begin{aligned} \text{EClo}(\{q_2, q_5\}) &= \text{EClo}(q_2) \cup \text{EClo}(q_5) \\ &= \{q_1, q_2, q_3\} \cup \{q_3, q_5, q_6\} \\ &= \end{aligned}$$



$$\text{EClo}(q_0) = \{q_0, q_1, q_2, q_3, q_4\}$$

$$\text{EClo}(q_2) = \{q_1, q_2, q_3\}$$

$$\text{EClo}(q_5) = \{q_3, q_5, q_6\}$$

$$\begin{aligned} \text{EClo}(\{q_2, q_5\}) &= \text{EClo}(q_2) \cup \text{EClo}(q_5) \\ &= \{q_1, q_2, q_3\} \cup \{q_3, q_5, q_6\} \\ &= \{q_1, q_2, q_3, q_5, q_6\} \end{aligned}$$

```
// Another example of epsilon-NFA  
val enfa2: ENFA = ENFA(...)
```

The ϵ -closures for states 0, 2, 5, and $\{2, 5\}$ are as follows:

```
enfa2.eclo(0)           // Set(0, 1, 2, 3, 4)  
enfa2.eclo(2)           // Set(1, 2, 3)  
enfa2.eclo(5)           // Set(3, 5, 6)  
enfa2.eclo(Set(2, 5)) // Set(1, 2, 3, 5, 6)
```



```
// Another example of epsilon-NFA
val enfa2: ENFA = ENFA(...)
```

The €-closures for states 0, 2, 5, and {2, 5} are as follows:

```
enfa2.eclo(0)           // Set(0, 1, 2, 3, 4)
enfa2.eclo(2)           // Set(1, 2, 3)
enfa2.eclo(5)           // Set(3, 5, 6)
enfa2.eclo(Set(2, 5)) // Set(1, 2, 3, 5, 6)
```

Then, how to implement the eclo method for €-closure?

```
case class ENFA(...):
  ...

  // The epsilon-closure of a state
  def eclo(q: State): Set[State] = ???

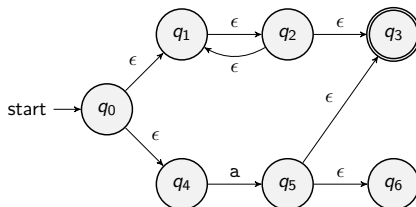
  // The epsilon-closure of a set of states
  def eclo(qs: Set[State]): Set[State] = qs.flatMap(eclo)
```

```
case class ENFA(...):  
  ...  
  // The epsilon-closure of a state  
  def wrongEClo(q: State): Set[State] =  
    val basis = Set(q) // Basis Case  
    val induc = enfa.trans(q, None).flatMap(wrongEClo) // Induction Case  
    basis ++ induc
```

```
case class ENFA(...):
  ...
  // The epsilon-closure of a state
  def wrongEClo(q: State): Set[State] =
    val basis = Set(q) // Basis Case
    val induc = enfa.trans(q, None).flatMap(wrongEClo) // Induction Case
    basis ++ induc
```

The above implementation is **WRONG** because of **infinite loop**:

```
enfa2.wrongEClo(5) // Set(3, 5, 6)
enfa2.wrongEClo(2) // INFINITE LOOP -- cycle between states 1 and 2
```

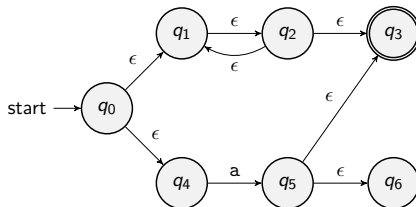


$$\text{EClo}(q_5) = \{q_3, q_5, q_6\}$$

$$\text{EClo}(q_2) = \{q_0, q_1, q_2, q_3, q_4\}$$

We can resolve the infinite loop issue by keeping the **visited states**:

```
case class ENFA(...):
  ...
  // The definitions of epsilon-closures
  def eclo(q: State): Set[State] =
    def aux(rest: List[State], visited: Set[State]): Set[State] = rest match
      case Nil          => visited
      case p :: targets => aux(
        rest      = (trans((p, None)) -- visited).toList ++ targets,
        visited   = visited + p,
      )
    aux(List(q), Set())
```



$$EClo(q_5) = \{q_3, q_5, q_6\}$$

$$EClo(q_2) = \{q_0, q_1, q_2, q_3, q_4\}$$

Definition (Extended Transition Function)

For a given ϵ -NFA $N^\epsilon = (Q, \Sigma, \delta, q_0, F)$, the **extended transition function** $\delta^* : \mathcal{P}(Q) \times \Sigma^* \rightarrow \mathcal{P}(Q)$ is defined as follows:

- **(Basis Case)** $\delta^*(S, \epsilon) = \text{EClo}(S)$
- **(Induction Case)** $\delta^*(S, xw) = \delta^*(\bigcup_{q \in \text{EClo}(S)} \delta(q, x), w)$

```
case class ENFA(...):
  ...

// The extended transition function of epsilon-NFA
def extTrans(qs: Set[State], w: Word): Set[State] = w match
  case "" => eclo(qs)
  case x <| w => extTrans(eclo(qs).flatMap(q => trans(q, Some(x))), w)
```

Definition (Acceptance of a Word)

For a given ϵ -NFA $N^\epsilon = (Q, \Sigma, \delta, q_0, F)$, we say that N^ϵ **accepts** a word $w \in \Sigma^*$ if and only if $\delta^*(q_0, w) \cap F \neq \emptyset$

Definition (Acceptance of a Word)

For a given ϵ -NFA $N^\epsilon = (Q, \Sigma, \delta, q_0, F)$, we say that N^ϵ **accepts** a word $w \in \Sigma^*$ if and only if $\delta^*(q_0, w) \cap F \neq \emptyset$

```
case class ENFA(...):  
  ...  
  
  // The acceptance of a word by epsilon-NFA  
  def accept(w: Word): Boolean =  
    extTrans(Set(initState), w).intersect(finalStates).nonEmpty
```

Definition (Acceptance of a Word)

For a given ϵ -NFA $N^\epsilon = (Q, \Sigma, \delta, q_0, F)$, we say that N^ϵ **accepts** a word $w \in \Sigma^*$ if and only if $\delta^*(q_0, w) \cap F \neq \emptyset$

```
case class ENFA(...):  
  ...  
  
  // The acceptance of a word by epsilon-NFA  
  def accept(w: Word): Boolean =  
    extTrans(Set(initState), w).intersect(finalStates).nonEmpty
```

Definition (Language of ϵ -NFA)

For a given ϵ -NFA $N^\epsilon = (Q, \Sigma, \delta, q_0, F)$, the **language** of N^ϵ is defined as follows:

$$L(N^\epsilon) = \{w \in \Sigma^* \mid N^\epsilon \text{ accepts } w\}$$

1. ϵ -Nondeterministic Finite Automata (ϵ -NFA)

ϵ -Transition

Definition

Transition Diagram and Transition Table

ϵ -Closures

Extended Transition Function

Language of ϵ -NFA

2. Equivalence of DFA and ϵ -NFA

DFA \leftarrow ϵ -NFA (Subset Construction)

Theorem (Equivalence of DFA and ϵ -NFA)

A language L is the language $L(D)$ of a DFA D if and only if L is the language $L(N^\epsilon)$ of an ϵ -NFA N^ϵ .

Theorem (Equivalence of DFA and ϵ -NFA)

A language L is the language $L(D)$ of a DFA D if and only if L is the language $L(N^\epsilon)$ of an ϵ -NFA N^ϵ .

Proof) By the following two theorems.

Theorem (DFA to ϵ -NFA)

For a given DFA $D = (Q, \Sigma, \delta, q, F)$, \exists ϵ -NFA N^ϵ . $L(D) = L(N^\epsilon)$.

Theorem (ϵ -NFA to DFA – Subset Construction)

For a given ϵ -NFA $N^\epsilon = (Q, \Sigma, \delta, q_0, F)$, \exists DFA D . $L(D) = L(N^\epsilon)$.

The formal proofs are exercises for you

Theorem (Equivalence of DFA and ϵ -NFA)

A language L is the language $L(D)$ of a DFA D if and only if L is the language $L(N^\epsilon)$ of an ϵ -NFA N^ϵ .

Proof) By the following two theorems.

Theorem (DFA to ϵ -NFA)

For a given DFA $D = (Q, \Sigma, \delta, q, F)$, \exists ϵ -NFA N^ϵ . $L(D) = L(N^\epsilon)$.

Theorem (ϵ -NFA to DFA – Subset Construction)

For a given ϵ -NFA $N^\epsilon = (Q, \Sigma, \delta, q_0, F)$, \exists DFA D . $L(D) = L(N^\epsilon)$.

The formal proofs are exercises for you

Let's see **examples** of the second theorem (DFA \leftarrow ϵ -NFA)

Theorem (ϵ -NFA to DFA – Subset Construction)

For a given ϵ -NFA $N^\epsilon = (Q_{N^\epsilon}, \Sigma, \delta_{N^\epsilon}, q_0, F_{N^\epsilon})$, \exists DFA D . $L(D) = L(N^\epsilon)$.

Proof) Define a DFA

$$D = (Q_D, \Sigma, \delta_D, \text{EClo}(q_0), F_D)$$

where

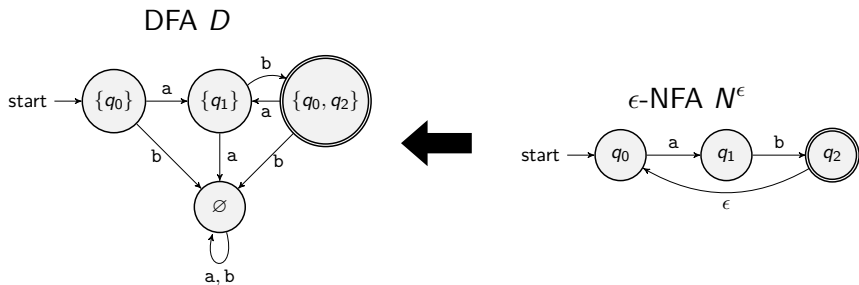
- $Q_D = \{S \subseteq Q_{N^\epsilon} \mid S = \text{EClo}(S)\}$
- $\forall S \in Q_D. \forall x \in \Sigma.$

$$\delta_D(S, x) = \text{EClo} \left(\bigcup_{q \in S} \delta_{N^\epsilon}(q, x) \right)$$

- $F_D = \{S \in Q_D \mid S \cap F \neq \emptyset\}$

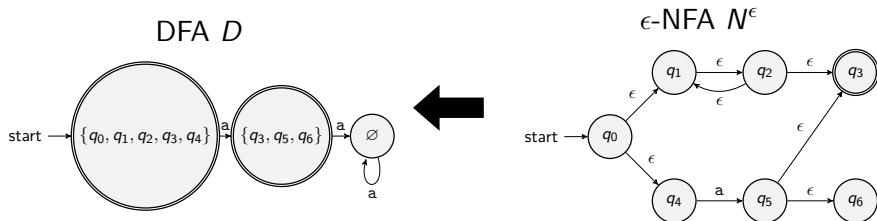
The states of D are the sets of states of N^ϵ closed under ϵ -closure.
And, the ϵ -closure is **idempotent** (i.e., $\text{EClo}(\text{EClo}(S)) = \text{EClo}(S)$)

DFA \leftarrow ϵ -NFA (Subset Construction) – Examples



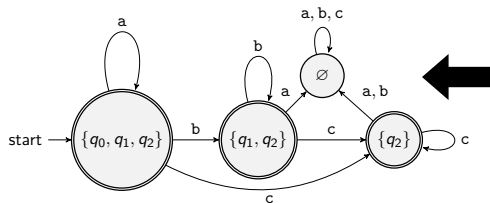
$$L(D) = L(N^\epsilon) = \{(ab)^n \mid n \geq 1\}$$

DFA \leftarrow ϵ -NFA (Subset Construction) – Examples

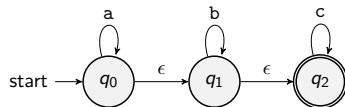


DFA \leftarrow ϵ -NFA (Subset Construction) – Examples

DFA D



ϵ -NFA N^ϵ



$$L(D) = L(N^\epsilon) = \{a^i b^j c^k \mid i, j, k \geq 0\}$$

1. ϵ -Nondeterministic Finite Automata (ϵ -NFA)

- ϵ -Transition

- Definition

- Transition Diagram and Transition Table

- ϵ -Closures

- Extended Transition Function

- Language of ϵ -NFA

2. Equivalence of DFA and ϵ -NFA

- DFA \leftarrow ϵ -NFA (Subset Construction)

- Please see this document on GitHub:

<https://github.com/ku-plrg-classroom/docs/tree/main/cose215/fa-examples>

- The due date is 23:59 on Apr. 3 (Wed.).
- Please only submit `Implementation.scala` file to **Blackboard**.

- Regular Expressions and Languages

Jihyeok Park

jihyeok_park@korea.ac.kr

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