

# Lecture 5 – $\epsilon$ -Nondeterministic Finite Automata ( $\epsilon$ -NFA)

## COSE215: Theory of Computation

Jihyeok Park



2023 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
- Equivalence of DFA and NFA
  - $\text{DFA} \rightarrow \text{NFA}$
  - $\text{DFA} \leftarrow \text{NFA}$  (Subset Construction)

## 1. $\epsilon$ -Nondeterministic Finite Automata ( $\epsilon$ -NFA)

- $\epsilon$ -Transition

- Definition

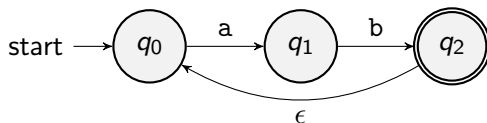
- Transition Diagram and Transition Table

- $\epsilon$ -Closures

- Extended Transition Function

- Language of  $\epsilon$ -NFA

- Equivalence of DFA and  $\epsilon$ -NFA



ab      abab      ababab      ...

## Definition ( $\epsilon$ -Nondeterministic Finite Automaton ( $\epsilon$ -NFA))

An  $\epsilon$ -nondeterministic finite automaton is a 5-tuple:

$$N_{\epsilon} = (Q, \Sigma, \delta, q_0, F)$$

- $Q$  is a finite set of **states**
- $\Sigma$  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**

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

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

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

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

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

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

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

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

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

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

```
// The type definitions of states and symbols
type State = Int
type Symbol = Char
// 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 enfa: ENFA = ENFA(
  states = Set(0, 1, 2),
  symbols = Set('a', 'b'),
  trans = Map(
    (0, Some('a')) -> Set(1), (1, Some('a')) -> Set(), (2, Some('a')) -> Set(),
    (0, Some('b')) -> Set(), (1, Some('b')) -> Set(2), (2, Some('b')) -> Set(),
    (0, None)       -> Set(), (1, None)       -> Set(), (2, None)       -> Set(0),
  ),
  initState = 0,
  finalStates = Set(2),
)
```

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

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

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

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

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

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

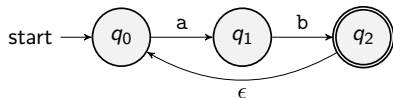
$$\delta(q_2, b) = \emptyset$$

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

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

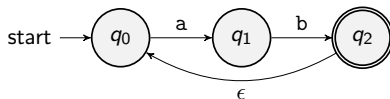
$$\delta(q_2, \epsilon) = \{q_0\}$$

**Transition Diagram**



**Transition Table**

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



## Definition ( $\epsilon$ -Closures)

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

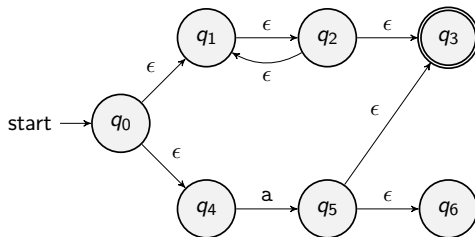
- **(Basis Case)**  $q \in \text{EClose}(q)$
- **(Induction Case)**  $q' \in \text{EClose}(q) \wedge q'' \in \delta(q', \epsilon) \Rightarrow q'' \in \text{EClose}(q)$

We sometimes need to define the  $\epsilon$ -closure for a set of states  $S$ :

$$\text{EClose}(S) = \bigcup_{q \in S} \text{EClose}(q)$$

- $\text{EClose}(q_2) = \{q_0, q_2\}$





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

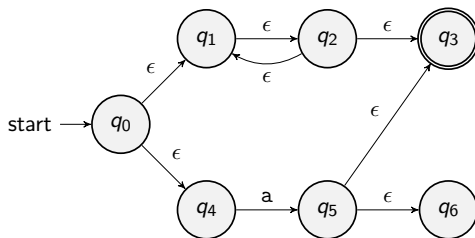
$$\text{EClose}(q_2) = \{q_1, q_2, q_3\}$$

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

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

```
// Another example of epsilon-NFA
val enfa2: ENFA = ENFA(
  states = Set(0, 1, 2, 3, 4, 5, 6),
  symbols = Set('a'),
  trans = Map(
    (0, Some('a')) -> Set(), (0, None) -> Set(1, 4),
    (1, Some('a')) -> Set(), (1, None) -> Set(2),
    (2, Some('a')) -> Set(), (2, None) -> Set(1, 3),
    (3, Some('a')) -> Set(), (3, None) -> Set(),
    (4, Some('a')) -> Set(5), (4, None) -> Set(),
    (5, Some('a')) -> Set(), (5, None) -> Set(3, 6),
    (6, Some('a')) -> Set(), (6, None) -> Set(),
  ),
  initState = 0,
  finalStates = Set(3),
)

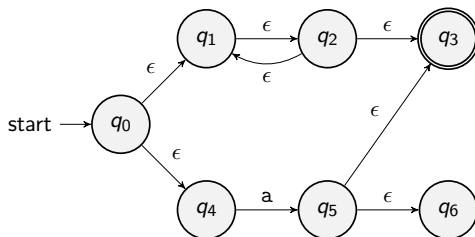
// The definitions of epsilon-closures
def eclose(enfa: ENFA)(q: State): Set[State] = ???
// The epsilon-closures for state 0, 2, and 5
eclose(enfa2)(0) // Set(0, 1, 2, 3, 4)
eclose(enfa2)(2) // Set(1, 2, 3)
eclose(enfa2)(5) // Set(3, 5, 6)
```



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

```
// The WRONG definitions of epsilon-closures because of infinite loop
def wrongEClose(enfa: ENFA)(q: State): Set[State] =
  val basis = Set(q) // Basis Case
  val induc = enfa.trans(q, None) // Induction Case
  .flatMap(q => eclose(enfa)(q))
  basis ++ induc

wrongEClose(enfa2)(5) // Set(3, 5, 6)
wrongEClose(enfa2)(2) // infinite loop
```



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

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

## Definition (Extended Transition Function)

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

- **(Basis Case)**  $\delta^*(q, \epsilon) = \text{EClose}(q)$
- **(Induction Case)**  $\delta^*(q, aw) = \bigcup_{q' \in \text{EClose}(q)} \bigcup_{q'' \in \delta(q', a)} \delta^*(q'', w)$

```
// The type definition of words
type Word = String
// A helper function to extract first symbol and rest of word
object `<|` { def unapply(w: Word) = w.headOption.map((_, w.drop(1))) }
// The extended transition function of epsilon-NFA
def extTrans(enfa: ENFA)(q: State, w: Word): Set[State] = w match
  case "" => eclose(enfa)(q)
  case a <| x => eclose(enfa)(q)
    .flatMap(q => enfa.trans(q, Some(a)))
    .flatMap(q => extTrans(enfa)(q, x))
```

### Definition (Acceptance of a Word)

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

```
// The acceptance of a word by epsilon-NFA
def accept(enfa: ENFA)(w: Word): Boolean =
  val curStates: Set[State] = extTrans(enfa)(enfa.initState, w)
  curStates.intersect(enfa.finalStates).nonEmpty
```

### Definition (Language of $\epsilon$ -NFA)

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

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

## Theorem (Equivalence of DFA and $\epsilon$ -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  $\epsilon$ -NFA  $N_\epsilon$ .*

**Proof)** By the following two theorems.

## Theorem (DFA to $\epsilon$ -NFA)

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

**Proof)** Exercise (Refer to the previous lecture)

## Theorem ( $\epsilon$ -NFA to DFA – Subset Construction)

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

**Proof)** Exercise (Refer to the previous lecture)

**Theorem ( $\epsilon$ -NFA to DFA – Subset Construction)**

For a given  $\epsilon$ -NFA  $N_\epsilon = (Q, \Sigma, \delta_{N_\epsilon}, q_0, F)$ ,  $\exists$  DFA  $D$ .  $L(D) = L(N_\epsilon)$ .

**Proof)** Define a DFA

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

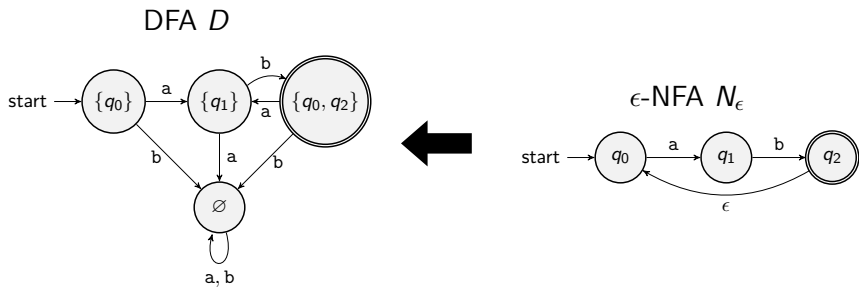
where

- $Q_D = \{S \subseteq Q \mid S = \text{EClose}(S)\}$
- $\forall S \in Q_D. \forall a \in \Sigma.$

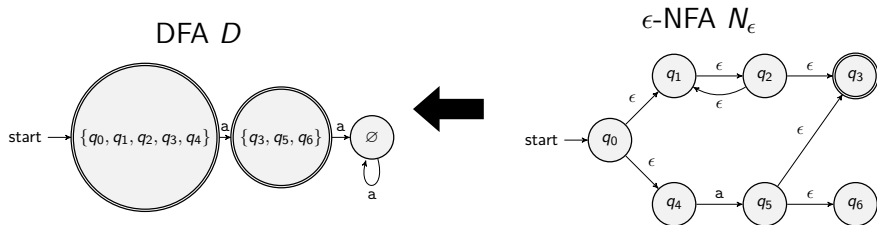
$$\delta_D(S, a) = \text{EClose} \left( \bigcup_{q \in S} \delta_{N_\epsilon}(q, a) \right)$$

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

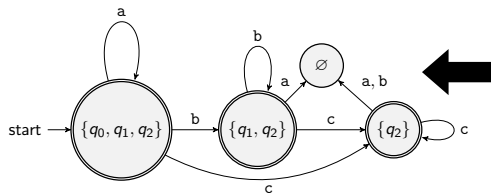
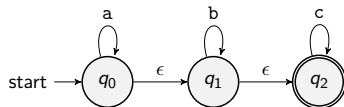




$$L(D) = L(N_\epsilon) = \{(ab)^n \mid n \geq 1\}$$



$$L(D) = L(N_\epsilon) = \{\epsilon, a\}$$

DFA  $D$  $\epsilon$ -NFA  $N_\epsilon$ 

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

## 1. $\epsilon$ -Nondeterministic Finite Automata ( $\epsilon$ -NFA)

- $\epsilon$ -Transition

- Definition

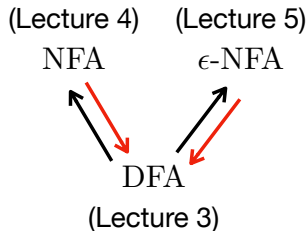
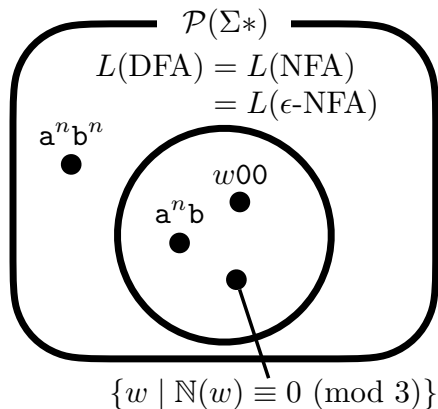
- Transition Diagram and Transition Table

- $\epsilon$ -Closures

- Extended Transition Function

- Language of  $\epsilon$ -NFA

- Equivalence of DFA and  $\epsilon$ -NFA



→ : Subset Construction

- Regular Expressions and Languages

Jihyeok Park

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

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