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

COSE215: Theory of Computation

Jihyeok Park

PLRG

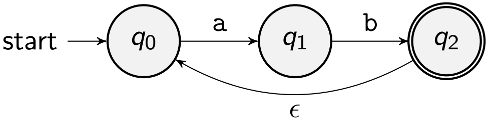
2023 Spring

Recall

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
     - DFA $\rightarrow$ NFA
     - DFA $\leftarrow$ NFA (Subset Construction)

Contents

ϵ-Transition



ab   abab   ababab $\cdots$

Definition of ϵ-NFA

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

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

$$\begin{array}{lll}
\delta(q_0, \mathtt{a}) = \{q_1\} & \delta(q_1, \mathtt{a}) = \varnothing & \delta(q_2, \mathtt{a}) = \varnothing \\
\delta(q_0, \mathtt{b}) = \varnothing & \delta(q_1, \mathtt{b}) = \{q_2\} & \delta(q_2, \mathtt{b}) = \varnothing \\
\delta(q_0, \epsilon) = \varnothing & \delta(q_1, \epsilon) = \varnothing & \delta(q_2, \epsilon) = \{q_0\}
\end{array}$$

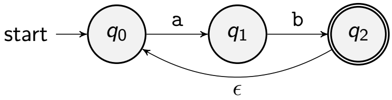
Definition of ϵ-NFA

```scala
// 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),
)
```

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

$$\delta(q_0, \mathtt{a}) = \{q_1\} \qquad \delta(q_1, \mathtt{a}) = \varnothing \qquad \delta(q_2, \mathtt{a}) = \varnothing$$
$$\delta(q_0, \mathtt{b}) = \varnothing \qquad \delta(q_1, \mathtt{b}) = \{q_2\} \qquad \delta(q_2, \mathtt{b}) = \varnothing$$
$$\delta(q_0, \epsilon) = \varnothing \qquad \delta(q_1, \epsilon) = \varnothing \qquad \delta(q_2, \epsilon) = \{q_0\}$$

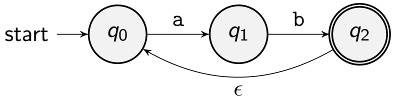
Transition Diagram



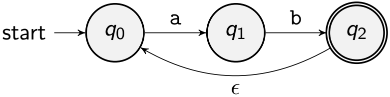
Transition Table

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

ϵ-Closures



ϵ-Closures



Definition (ϵ-Closures)

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

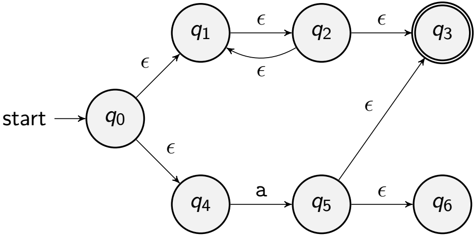
- **(Basis Case)** $q \in \text{EClose}(q)$
- **(Induction Case)** $q' \in \text{EClose}(q) \land q'' \in \delta(q', \epsilon) \Rightarrow q'' \in \text{EClose}(q)$

We sometimes need to define the ϵ-closure for a set of states S:

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

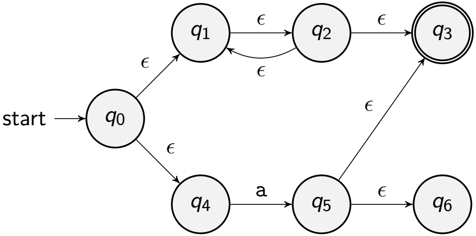
- $\text{EClose}(q_2) = \{q_0, q_2\}$

ϵ-Closures – Example



$$\text{EClose}(q_0) \quad = $$

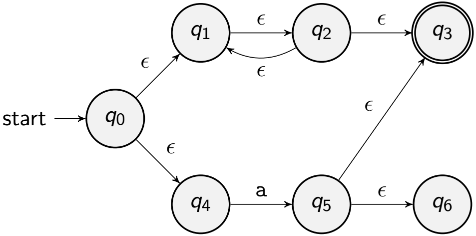
ϵ-Closures – Example



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

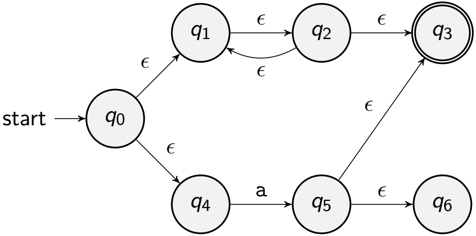
$$\mathsf{EClose}(q_2) =$$

ϵ-Closures – Example



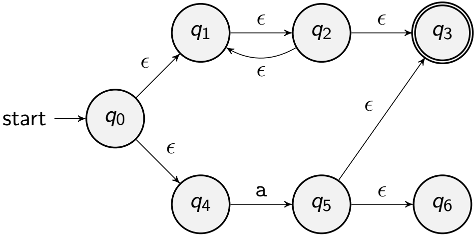
$$
\begin{aligned}
\mathrm{EClose}(q_0) &= \{q_0, q_1, q_2, q_3, q_4\} \\
\mathrm{EClose}(q_2) &= \{q_1, q_2, q_3\} \\
\mathrm{EClose}(q_5) &=
\end{aligned}
$$

ϵ-Closures – Example



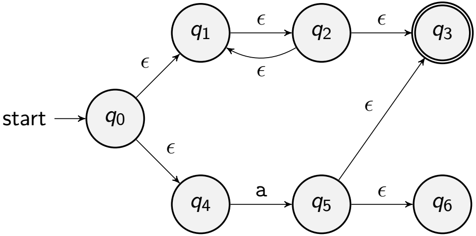
$$
\begin{aligned}
\mathrm{EClose}(q_0) &= \{q_0, q_1, q_2, q_3, q_4\} \\
\mathrm{EClose}(q_2) &= \{q_1, q_2, q_3\} \\
\mathrm{EClose}(q_5) &= \{q_3, q_5, q_6\} \\
\mathrm{EClose}(\{q_2, q_5\}) &=
\end{aligned}
$$

ϵ-Closures – Example



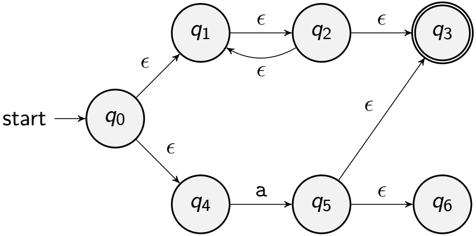
$$
\begin{aligned}
\mathsf{EClose}(q_0) &= \{q_0, q_1, q_2, q_3, q_4\} \\
\mathsf{EClose}(q_2) &= \{q_1, q_2, q_3\} \\
\mathsf{EClose}(q_5) &= \{q_3, q_5, q_6\} \\
\mathsf{EClose}(\{q_2, q_5\}) &= \mathsf{EClose}(q_2) \cup \mathsf{EClose}(q_5) \\
&=
\end{aligned}
$$

ϵ-Closures – Example



$$
\begin{aligned}
\mathrm{EClose}(q_0) &= \{q_0, q_1, q_2, q_3, q_4\} \\
\mathrm{EClose}(q_2) &= \{q_1, q_2, q_3\} \\
\mathrm{EClose}(q_5) &= \{q_3, q_5, q_6\} \\
\mathrm{EClose}(\{q_2, q_5\}) &= \mathrm{EClose}(q_2) \cup \mathrm{EClose}(q_5) \\
&= \{q_1, q_2, q_3\} \cup \{q_3, q_5, q_6\} \\
&=
\end{aligned}
$$

ϵ-Closures – Example

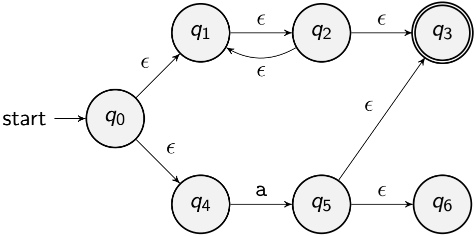


$$
\begin{aligned}
\mathrm{EClose}(q_0) &= \{q_0, q_1, q_2, q_3, q_4\} \\
\mathrm{EClose}(q_2) &= \{q_1, q_2, q_3\} \\
\mathrm{EClose}(q_5) &= \{q_3, q_5, q_6\} \\
\mathrm{EClose}(\{q_2, q_5\}) &= \mathrm{EClose}(q_2) \cup \mathrm{EClose}(q_5) \\
&= \{q_1, q_2, q_3\} \cup \{q_3, q_5, q_6\} \\
&= \{q_1, q_2, q_3, q_5, q_6\}
\end{aligned}
$$

ϵ-Closures

```
// 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)
```

ϵ-Closures

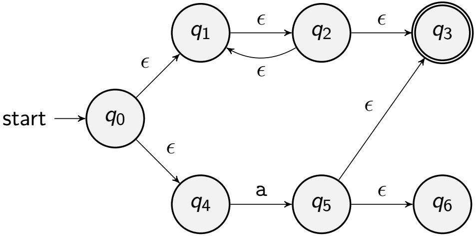


$$\text{EClose}(q_0) = \{q_0, q_1, q_2, q_3, q_4\}$$

```
// 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
```

ϵ-Closures



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

```scala
// 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())
```

Extended Transition Function

PLRG

Definition (Extended Transition Function)

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

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

```
// 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))
```

Language of ϵ-NFA

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 $\delta^*(q_0, w) \cap F \neq \varnothing$

```
// 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 ϵ-NFA)

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

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

Equivalence of DFA and ϵ-NFA

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. $L(D) = L(N_\epsilon)$.

Proof) Exercise (Refer to the previous lecture)

Theorem (ϵ-NFA to DFA – Subset Construction)

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

Proof) Exercise (Refer to the previous lecture)

DFA $\leftarrow$ ϵ-NFA (Subset Construction)

Theorem (ϵ-NFA to DFA – Subset Construction)

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

Proof) Define a DFA

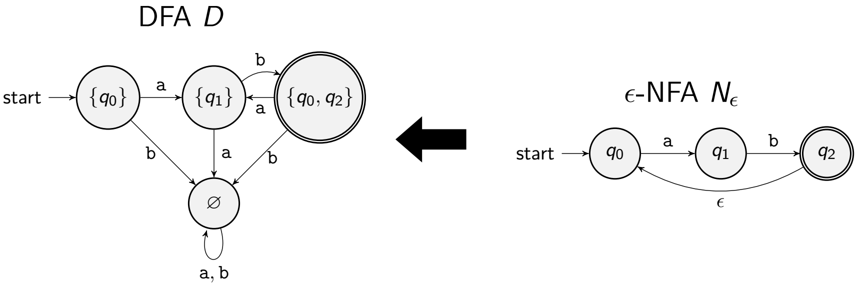
$$D = (Q_D, \Sigma, \delta_D, \mathrm{EClose}(q_0), F_D)$$

where

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

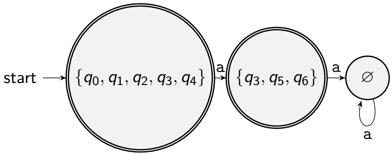
$$\delta_D(S, a) = \mathrm{EClose}\left(\bigcup_{q \in S} \delta_{N_\epsilon}(q, a)\right)$$

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

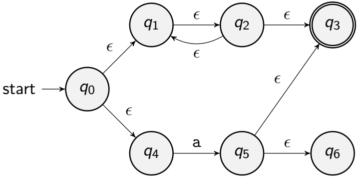


$$L(D) = L(N_\epsilon) = \{(\mathtt{ab})^n \mid n \geq 1\}$$

DFA D

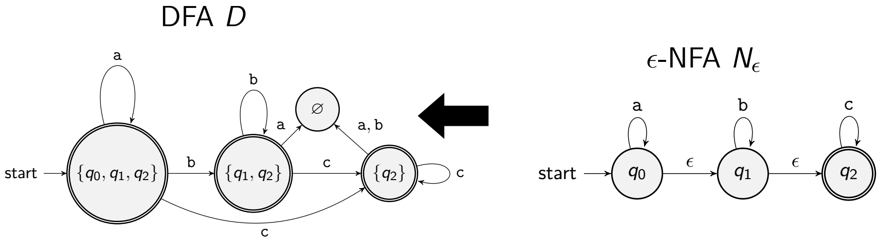


ϵ-NFA N_ϵ



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

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

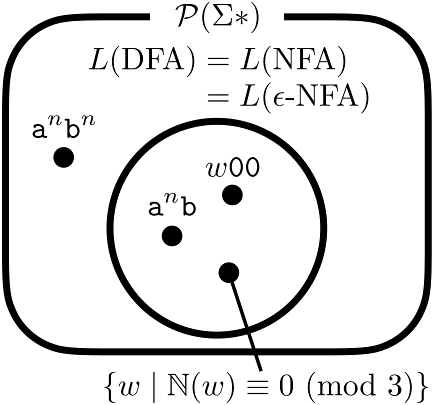


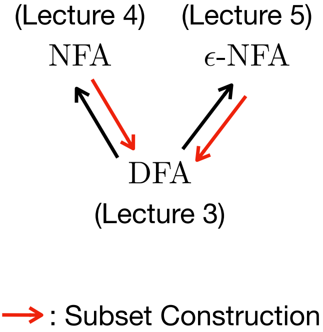
$$L(D) = L(N_\epsilon) = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i, j, k \geq 0\}$$

Summary

1. ϵ-Nondeterministic Finite Automata (ϵ-NFA)
   - ϵ-Transition
   - Definition
   - Transition Diagram and Transition Table
   - ϵ-Closures
   - Extended Transition Function
   - Language of ϵ-NFA
   - Equivalence of DFA and ϵ-NFA

Summary of Finite Automata





Next Lecture

- Regular Expressions and Languages

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