Kleene's Theorem

CS 2800: Discrete Structures, Spring 2015

Sid Chaudhuri

Pumping Lemma: Piazza @720

Structural Induction: Piazza @744

Kleene's Theorem

- A language is *regular*, i.e. it can be defined by a regular expression, if and only if it is recognized by a finite automaton

Kleene's Theorem

- A language is *regular*, i.e. it can be defined by a regular expression, if and only if it is recognized by a finite automaton
  - Regex has FA
    - Relatively simple construction

Kleene's Theorem

- A language is *regular*, i.e. it can be defined by a regular expression, if and only if it is recognized by a finite automaton
  - Regex has FA
    - Relatively simple construction
  - FA has regex
    - Tricky to prove

Regex → FA

- For every regular expression, there is a finite automaton that recognizes the same language

Regex → FA

- For every regular expression, there is a finite automaton that recognizes the same language
- We will construct an ε-NFA

Regex → FA

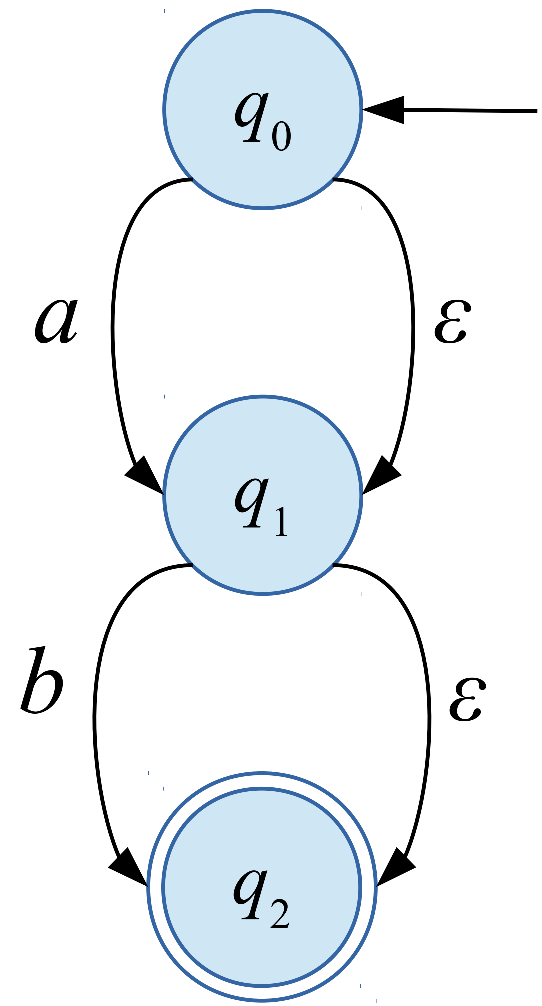
- For every regular expression, there is a finite automaton that recognizes the same language
- We will construct an ε-NFA
  - … which can be converted to an NFA

Regex → FA

- For every regular expression, there is a finite automaton that recognizes the same language
- We will construct an ε-NFA
  - … which can be converted to an NFA
  - … which can be converted to a DFA

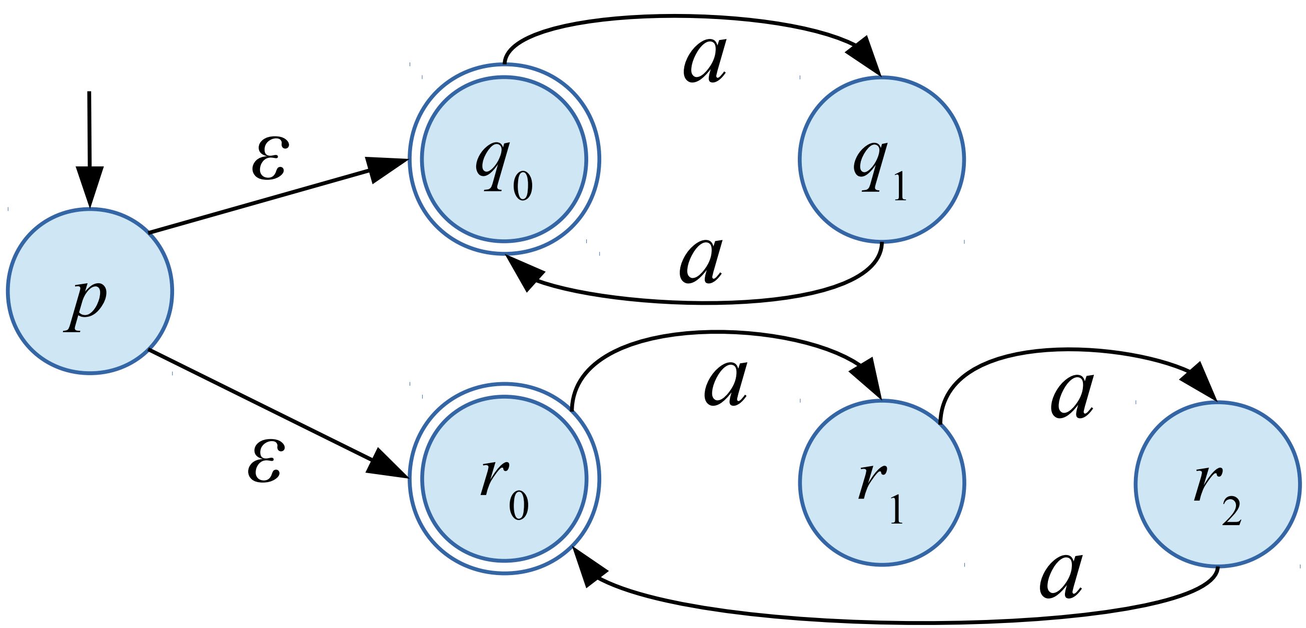
Recap: NFAs with epsilon transitions

- Just like ordinary NFAs, but…
  - Can "instantaneously" change state *without* reading an input symbol
  - Valid transitions of this type are shown by arcs labeled 'ε'
  - Note that ε does not suddenly become a member of the alphabet. Instead, we assume ε does not belong to *any* alphabet – it's a special symbol.



Why ε-NFAs?

- Suitable for representing "or" relations
- E.g. $L = \{ \, a^n \mid n \in \mathbf{N} \text{ is divisible by } 2 \text{ or } 3 \, \}$



- … but they're equivalent to NFAs and DFAs

ε-NFA to ordinary NFA

- The ε-closure of a state q is the set of states that can be reached from q following only ε-transitions

ε-NFA to ordinary NFA

- The ε-closure of a state q is the set of states that can be reached from q following only ε-transitions
  - The set includes q itself

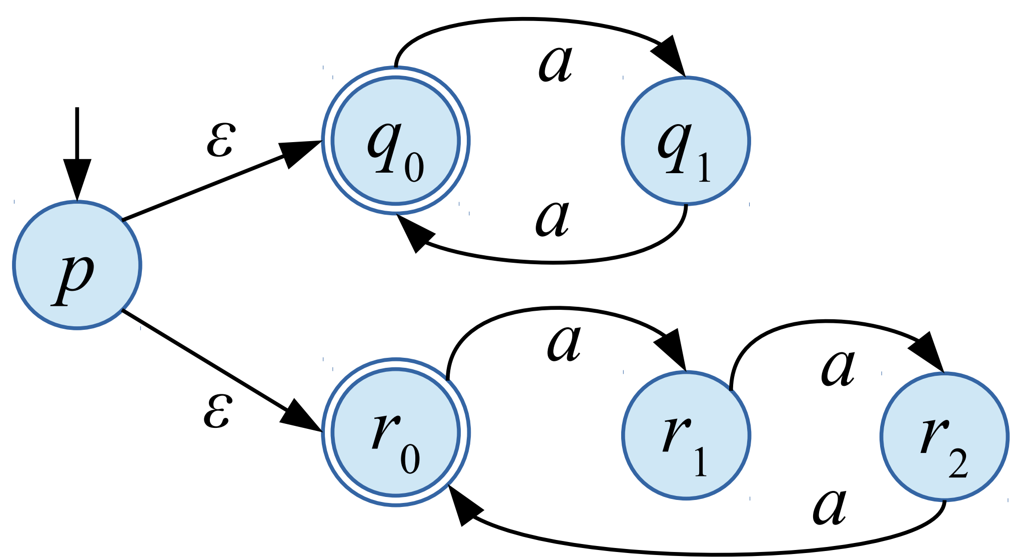
ε-NFA to ordinary NFA

- The ε-closure of a state q is the set of states that can be reached from q following only ε-transitions
  - The set includes q itself
  - We'll denote the set $ECLOSE(q)$

ε-NFA to ordinary NFA

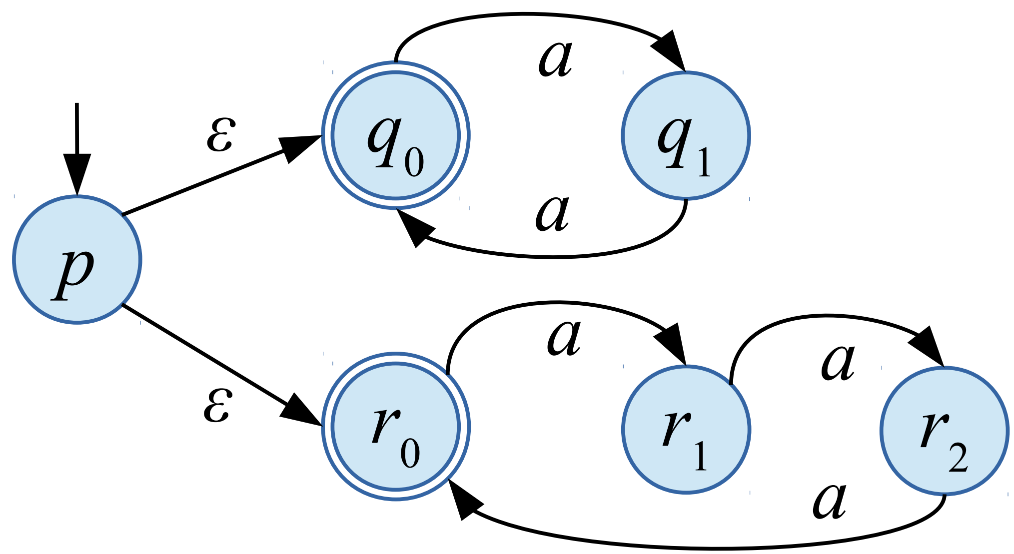
- The ε-closure of a state q is the set of states that can be reached from q following only ε-transitions
  - The set includes q itself
  - We'll denote the set $ECLOSE(q)$

$$ECLOSE(p) = ?$$



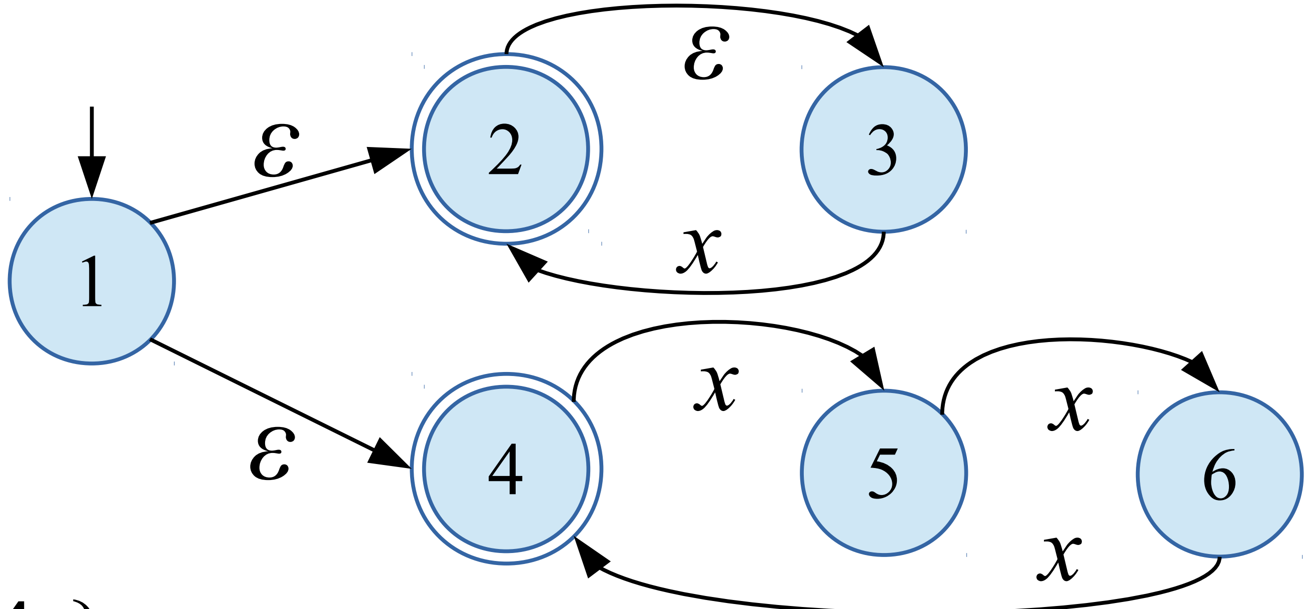
ε-NFA to ordinary NFA

- The ε-closure of a state q is the set of states that can be reached from q following only ε-transitions
  - The set includes q itself
  - We'll denote the set $ECLOSE(q)$



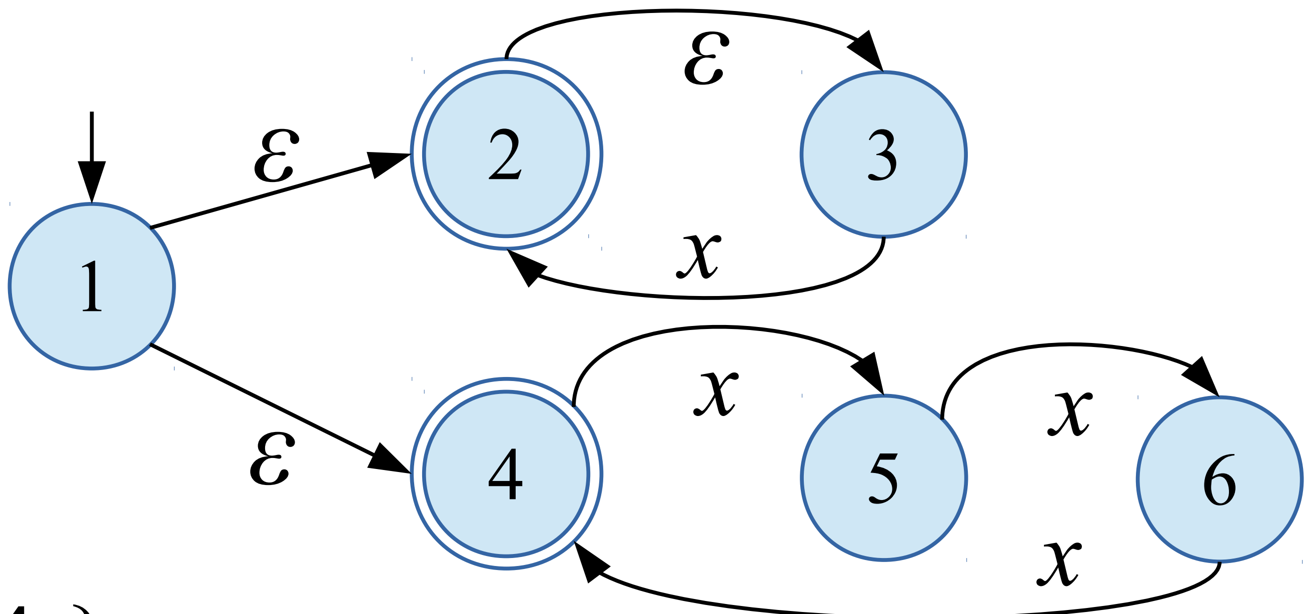
$$ECLOSE(p) = \{\, p, q_0, r_0 \,\}$$

$ECLOSE(1)?$



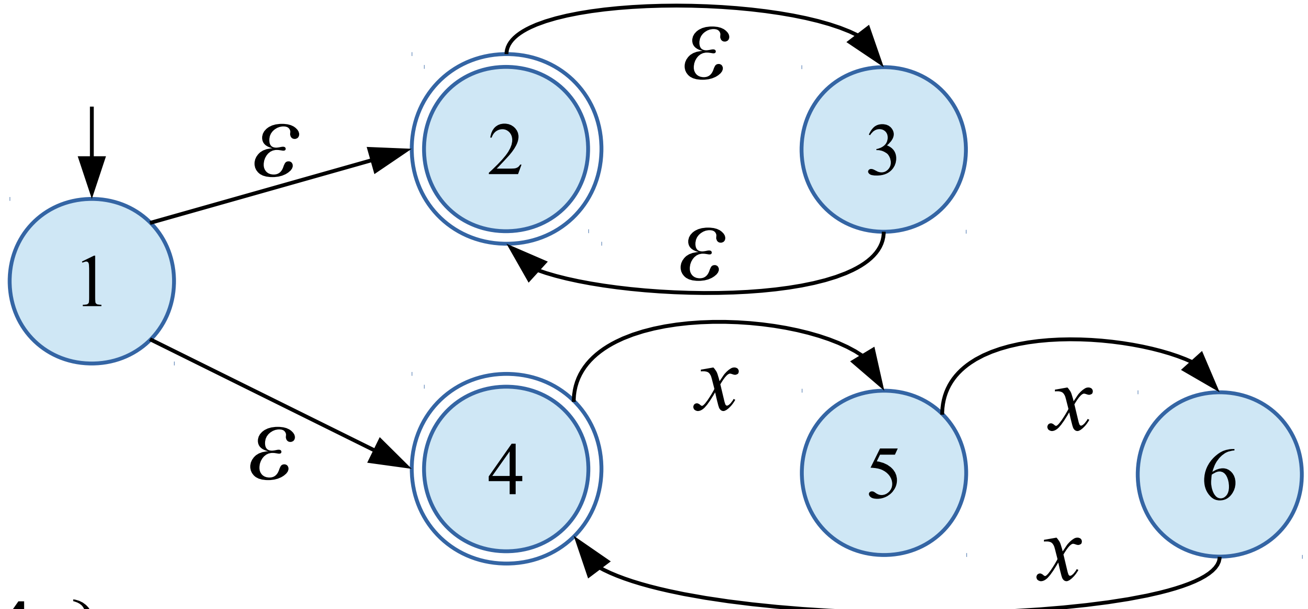
A) { 2, 4 }
B) { 1, 2, 4 }
C) { 1, 2, 3, 4 }
D) { 2, 3, 4 }

$ECLOSE(1)$?



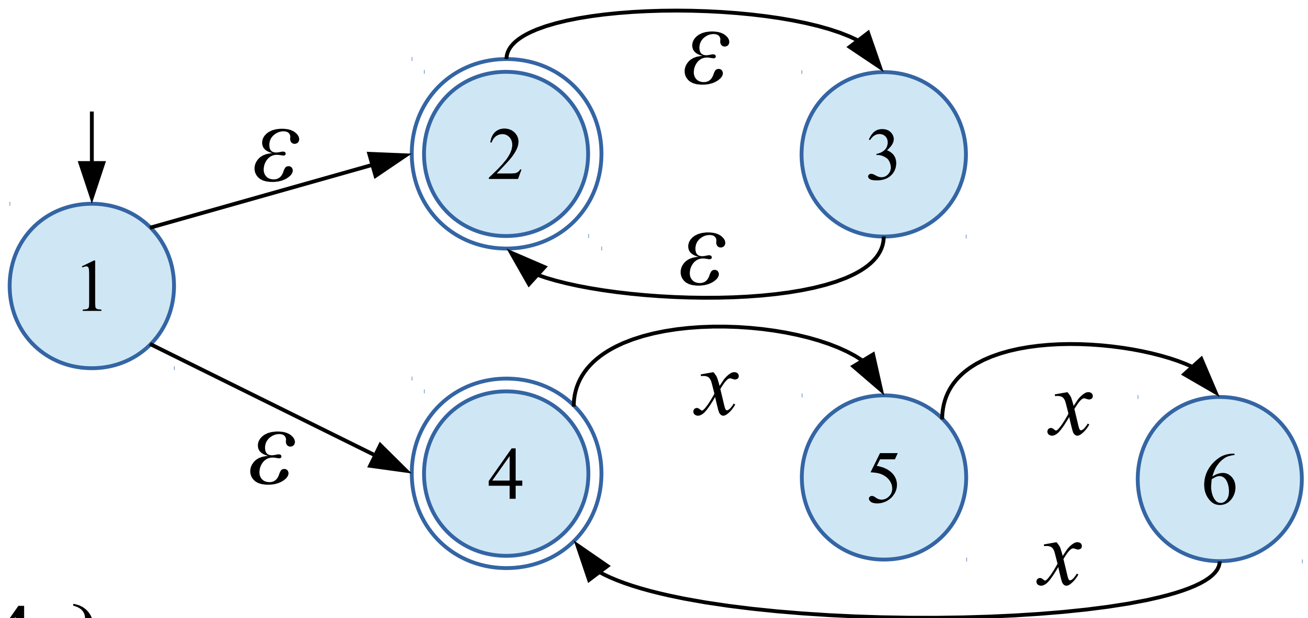
A) { 2, 4 }
B) { 1, 2, 4 }
C) **{ 1, 2, 3, 4 }**
D) { 2, 3, 4 }

$ECLOSE(1)$?



A) { 2, 4 }
B) { 1, 2, 4 }
C) { 1, 2, 3, 4 }
D) { 2, 3, 4 }

$ECLOSE(1)$?



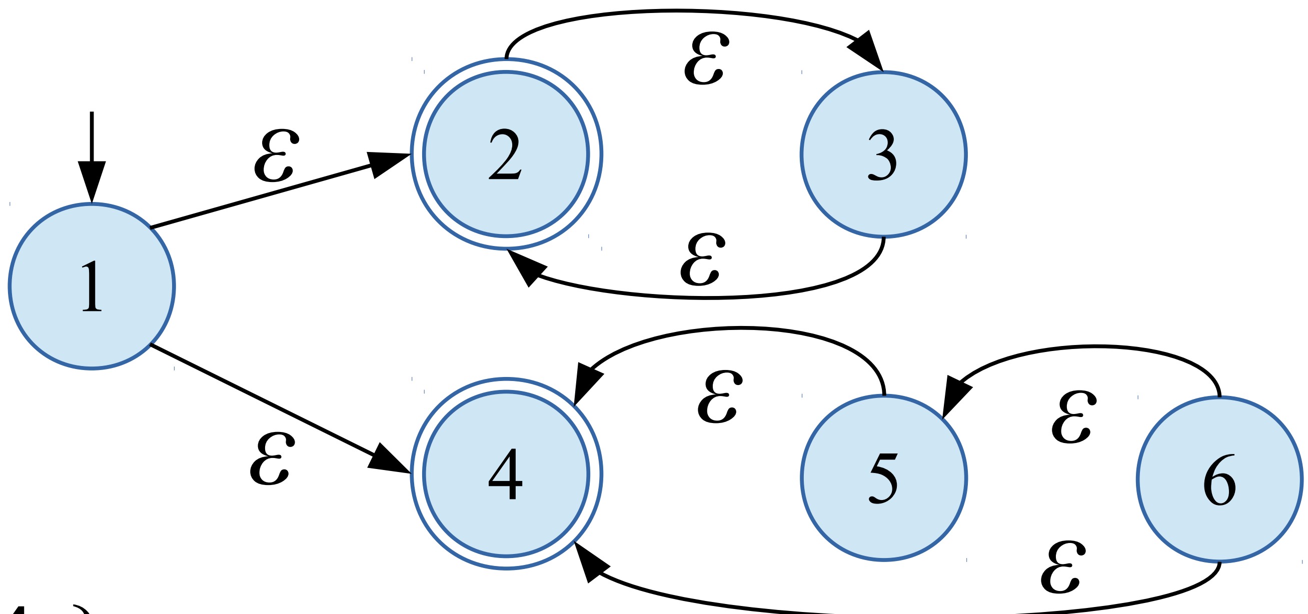
A) $\{ 2, 4 \}$

B) $\{ 1, 2, 4 \}$

C) $\mathbf{\{ 1, 2, 3, 4 \}}$

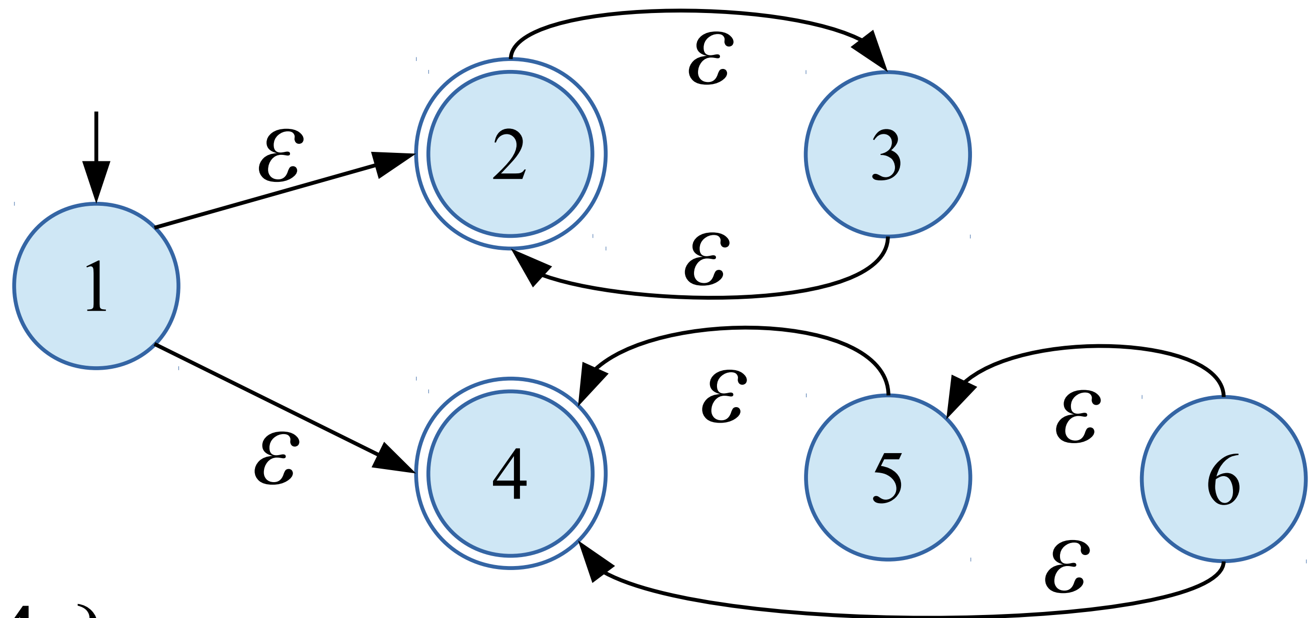
D) $\{ 2, 3, 4 \}$

$ECLOSE(1)?$



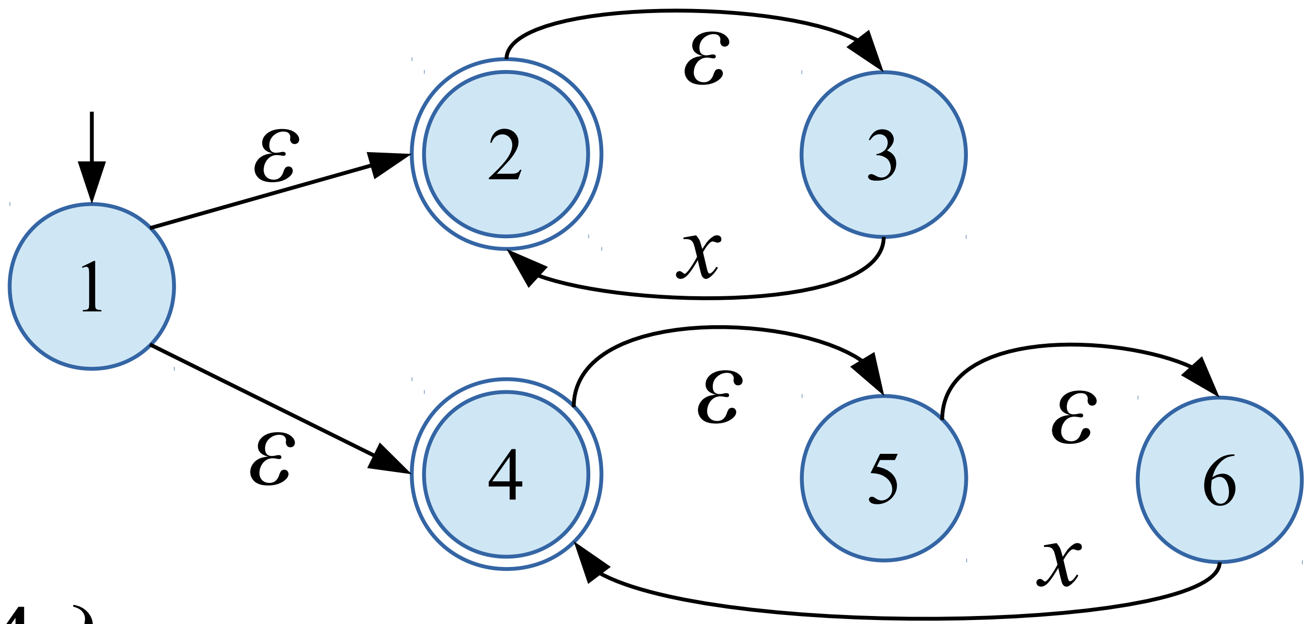
A) { 2, 4 }
B) { 1, 2, 4 }
C) { 1, 2, 3, 4 }
D) { 1, 2, 3, 4, 5, 6 }

$ECLOSE(1)$?



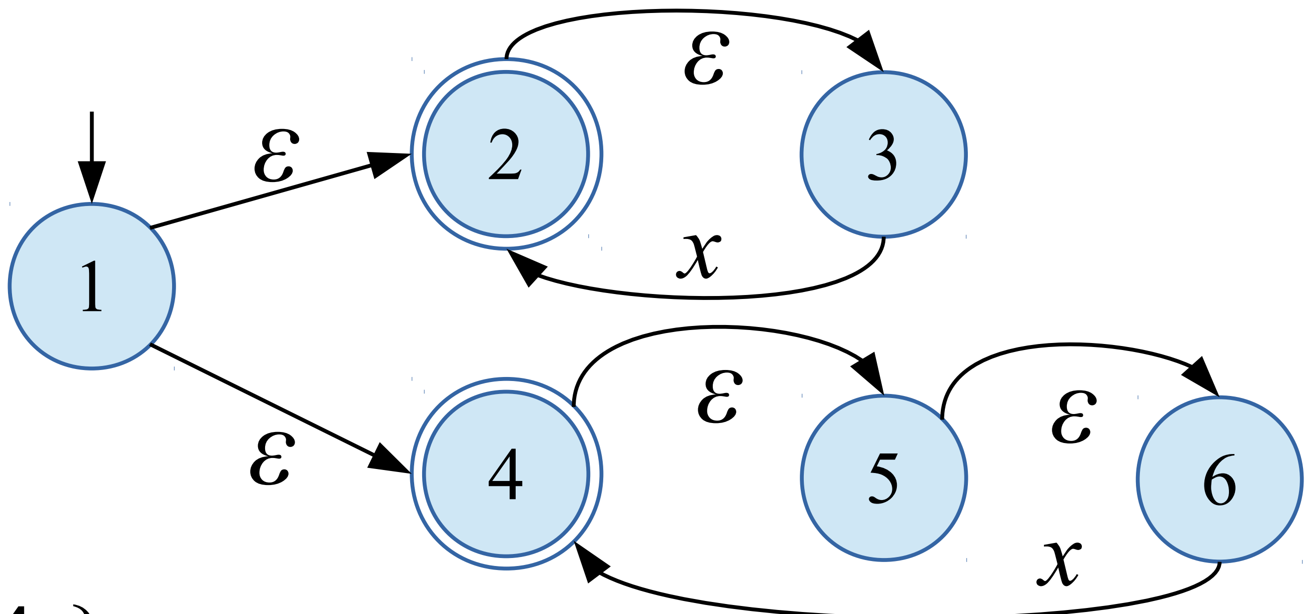
A) { 2, 4 }
B) { 1, 2, 4 }
C) **{ 1, 2, 3, 4 }**
D) { 1, 2, 3, 4, 5, 6 }

$ECLOSE(1)$?



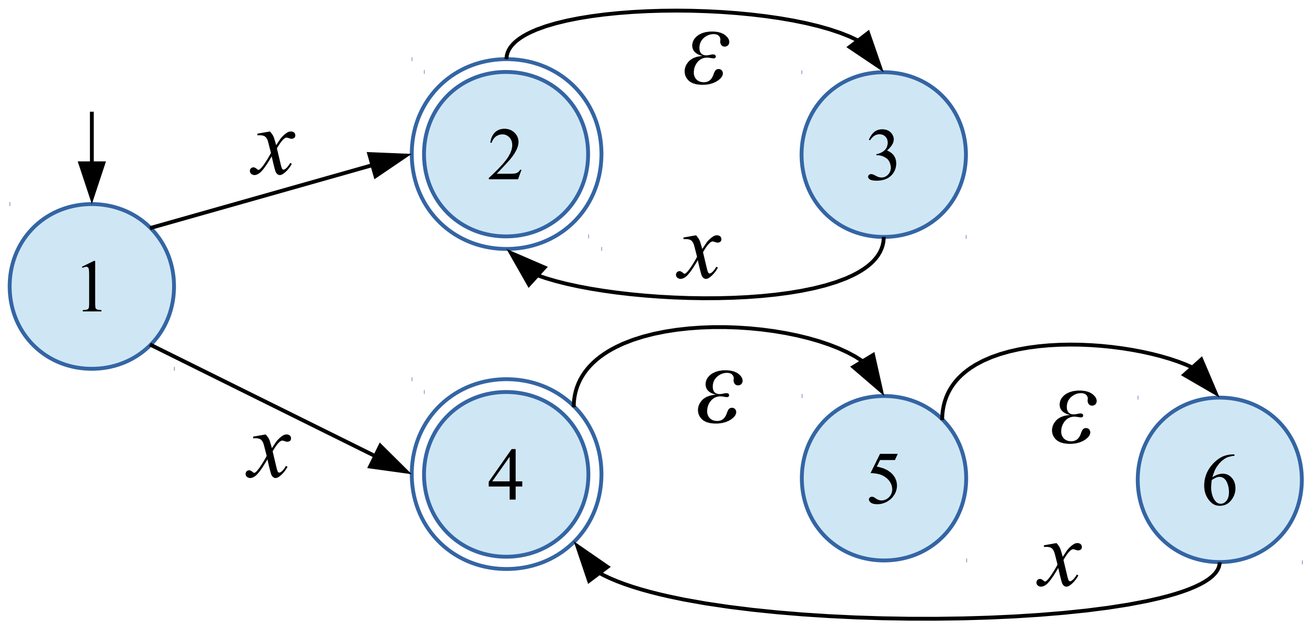
A) { 2, 4 }
B) { 1, 2, 4 }
C) { 1, 2, 3, 4 }
D) { 1, 2, 3, 4, 5, 6 }

$ECLOSE(1)$?



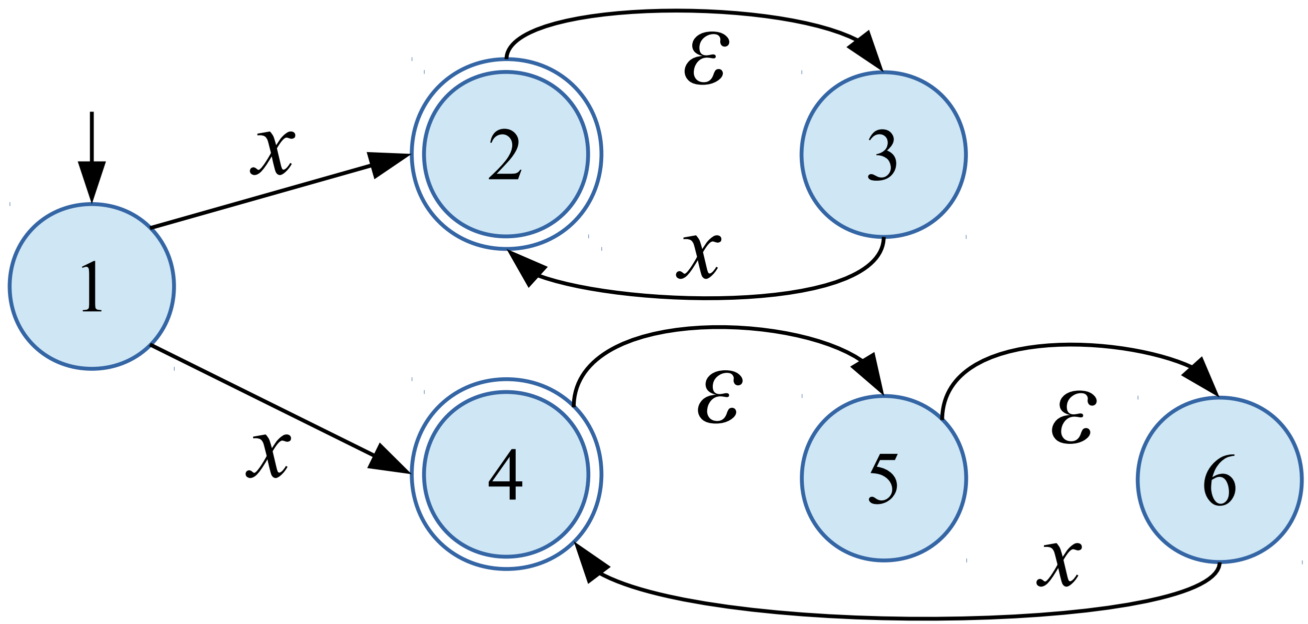
A) { 2, 4 }
B) { 1, 2, 4 }
C) { 1, 2, 3, 4 }
D) **{ 1, 2, 3, 4, 5, 6 }**

$ECLOSE(1)$?



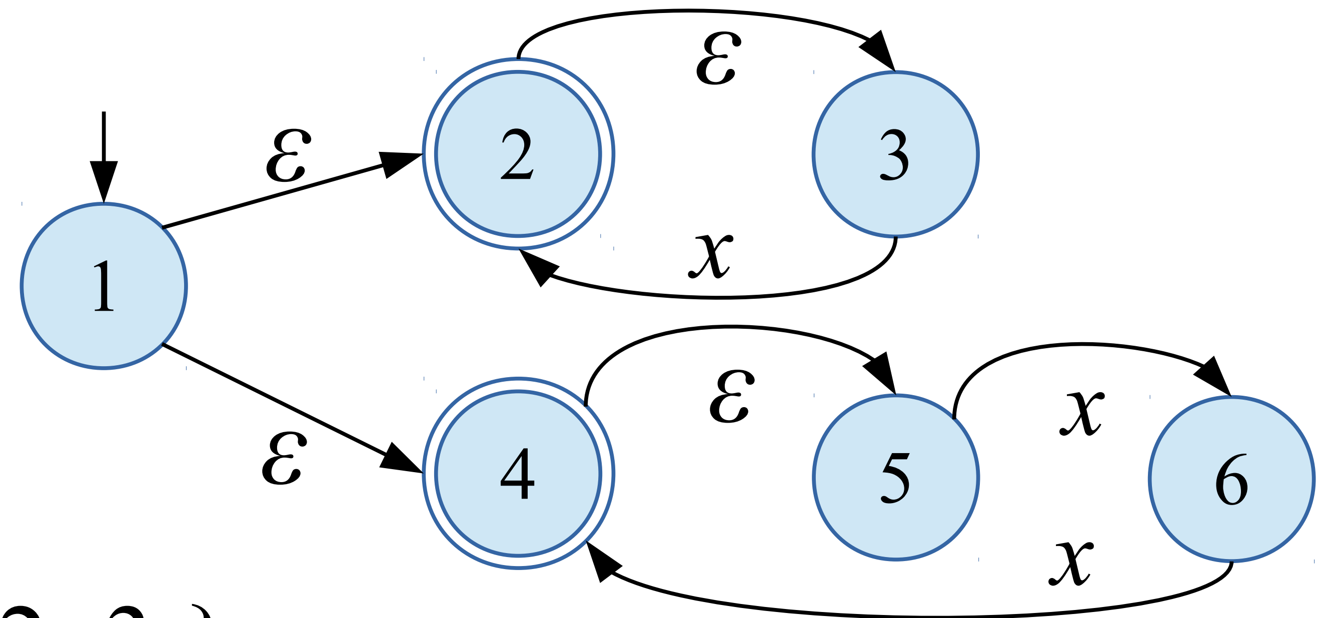
A) { 1 }
B) { 1, 2, 4 }
C) { 1, 2, 3, 4 }
D) { 1, 2, 3, 4, 5, 6 }

$ECLOSE(1)?$



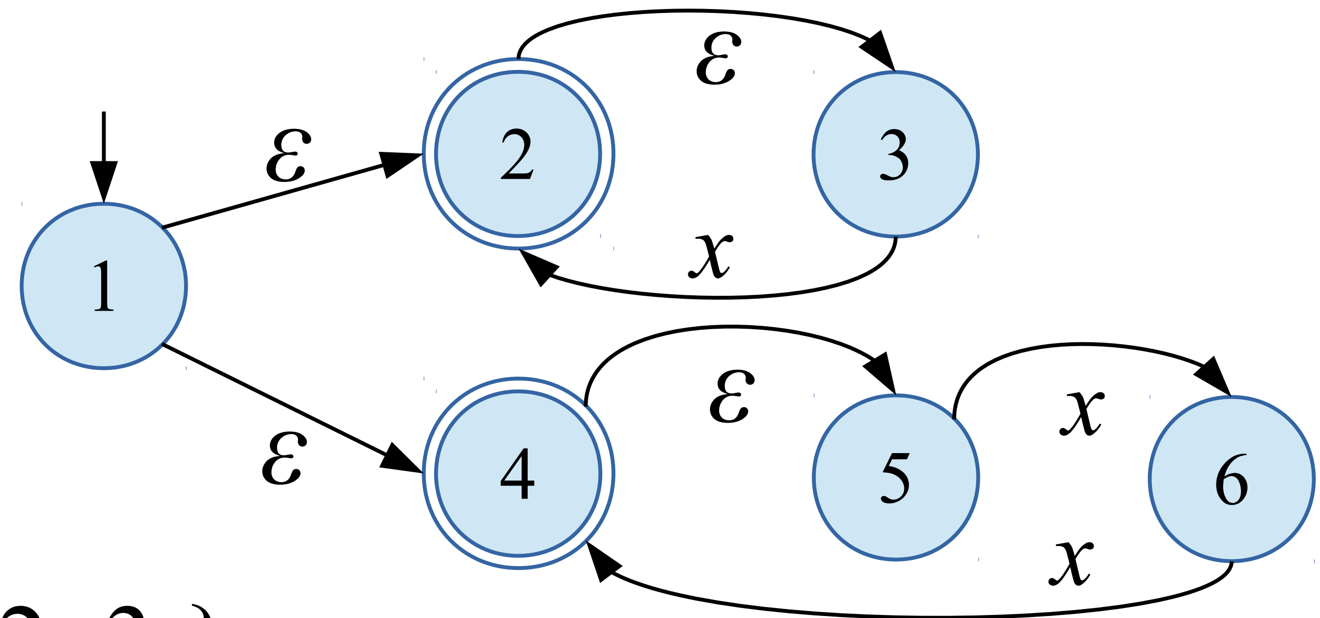
A) $\{ 1 \}$
B) $\{ 1, 2, 4 \}$
C) $\{ 1, 2, 3, 4 \}$
D) $\{ 1, 2, 3, 4, 5, 6 \}$

$ECLOSE(1)$?



A) { 1, 2, 3 }
B) { 1, 2, 4 }
C) { 1, 2, 3, 4 }
D) { 1, 2, 3, 4, 5, 6 }

$ECLOSE(1)$?



A) { 1, 2, 3 }
B) { 1, 2, 4 }
C) { 1, 2, 3, 4 }
D) { 1, 2, 3, 4, 5, 6 }

E) **{ 1, 2, 3, 4, 5 }**

ε-NFA to ordinary NFA

- Converting ε-NFA N_ε to ordinary NFA N (short-circuiting ε paths)

ε-NFA to ordinary NFA

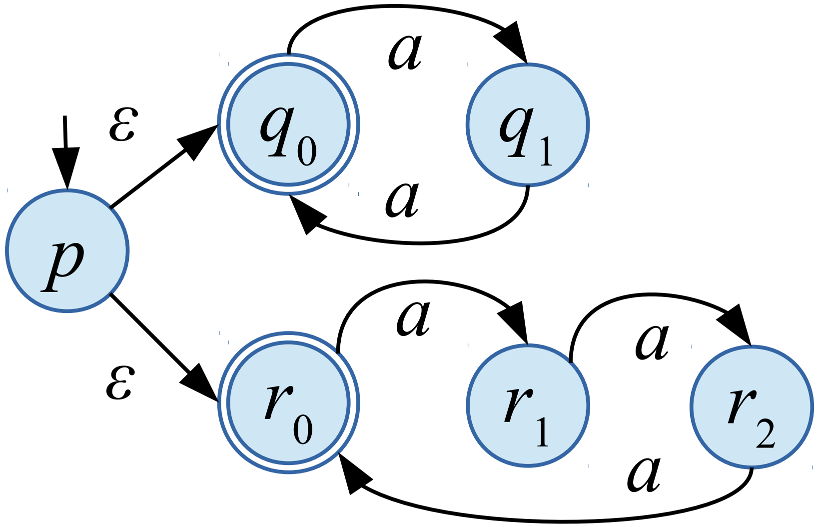
- Converting ε-NFA N_ε to ordinary NFA N (short-circuiting ε paths)
    1. Make p an accepting state of N iff $ECLOSE(p)$ contains an accepting state of N_ε

ε-NFA to ordinary NFA

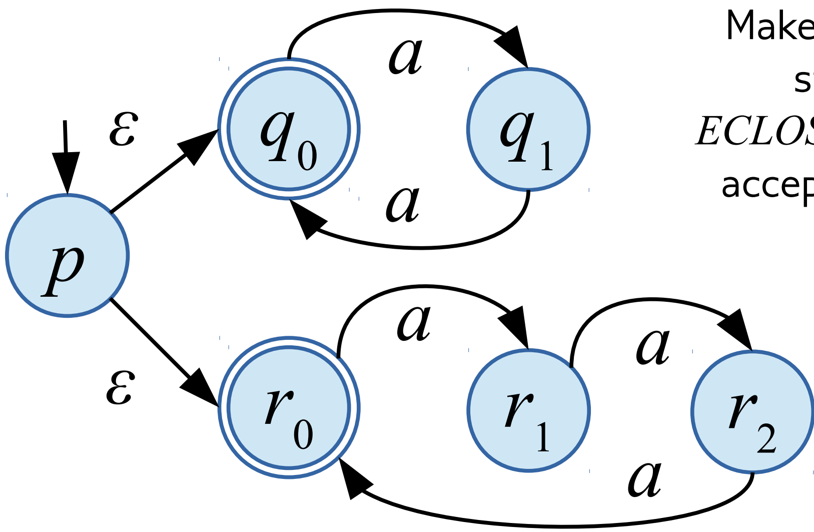
- Converting ε-NFA N_ε to ordinary NFA N (short-circuiting ε paths)
  1. Make p an accepting state of N iff $ECLOSE(p)$ contains an accepting state of N_ε
  2. Add an arc labeled a from p to q iff N_ε has an arc labeled a from some state in $ECLOSE(p)$ to q

ε-NFA to ordinary NFA

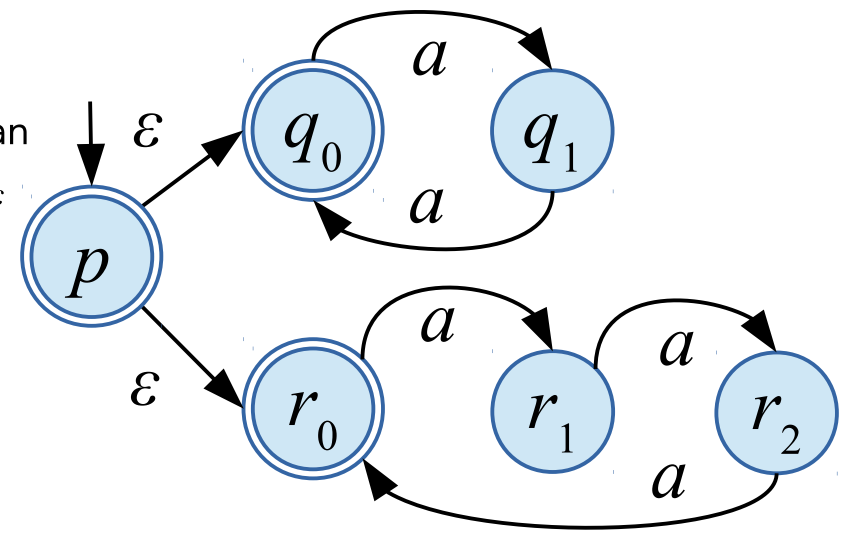
- Converting ε-NFA N_ε to ordinary NFA N (short-circuiting ε paths)
  1. Make p an accepting state of N iff $ECLOSE(p)$ contains an accepting state of N_ε
  2. Add an arc labeled a from p to q iff N_ε has an arc labeled a from some state in $ECLOSE(p)$ to q
  3. Delete all arcs labeled ε

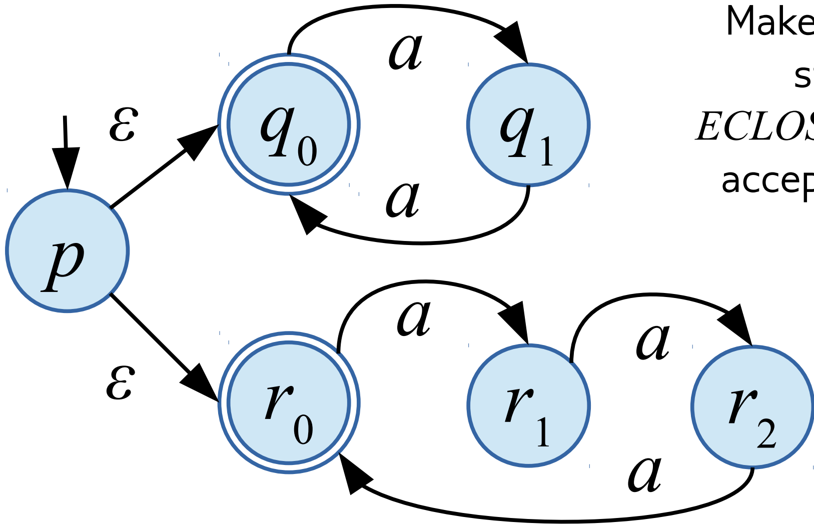


ε
ε
a
a
a
a
a
q0
q1
p
r0
r1
r2

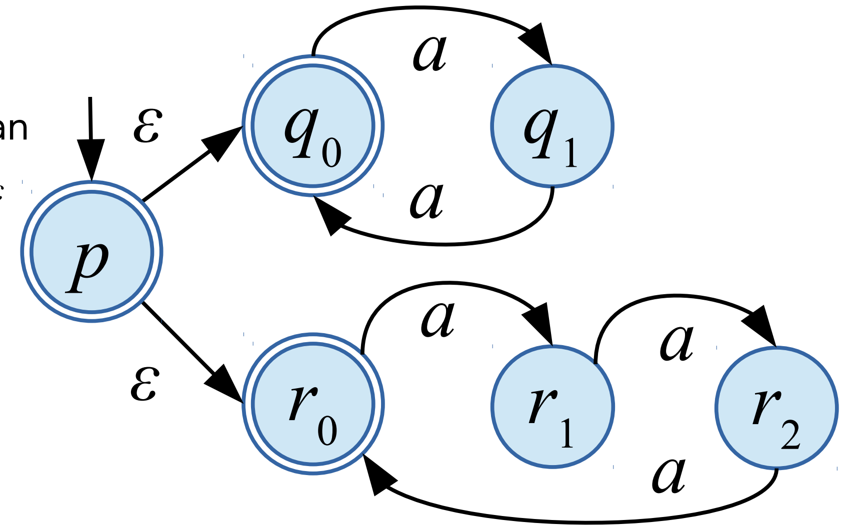


Make p an accepting state of N iff $ECLOSE(p)$ contains an accepting state of N_ε

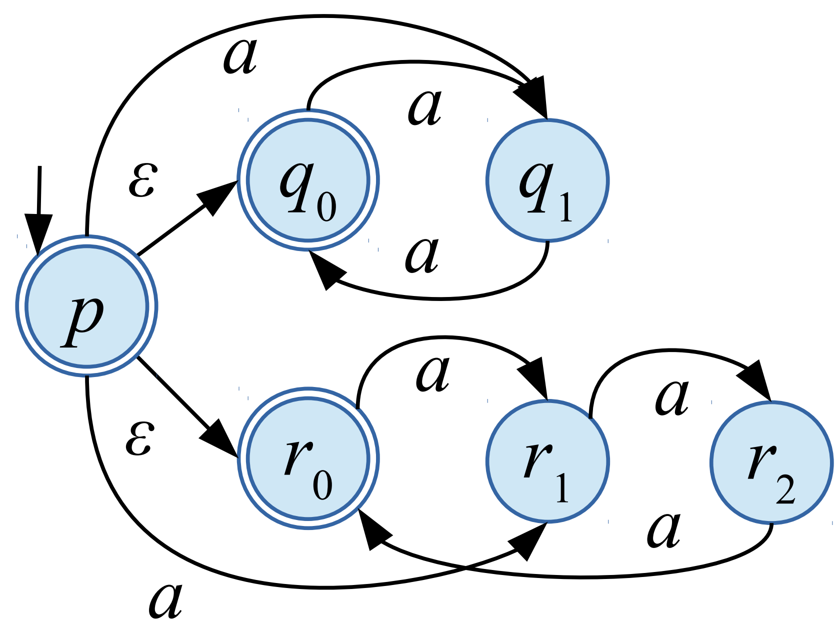


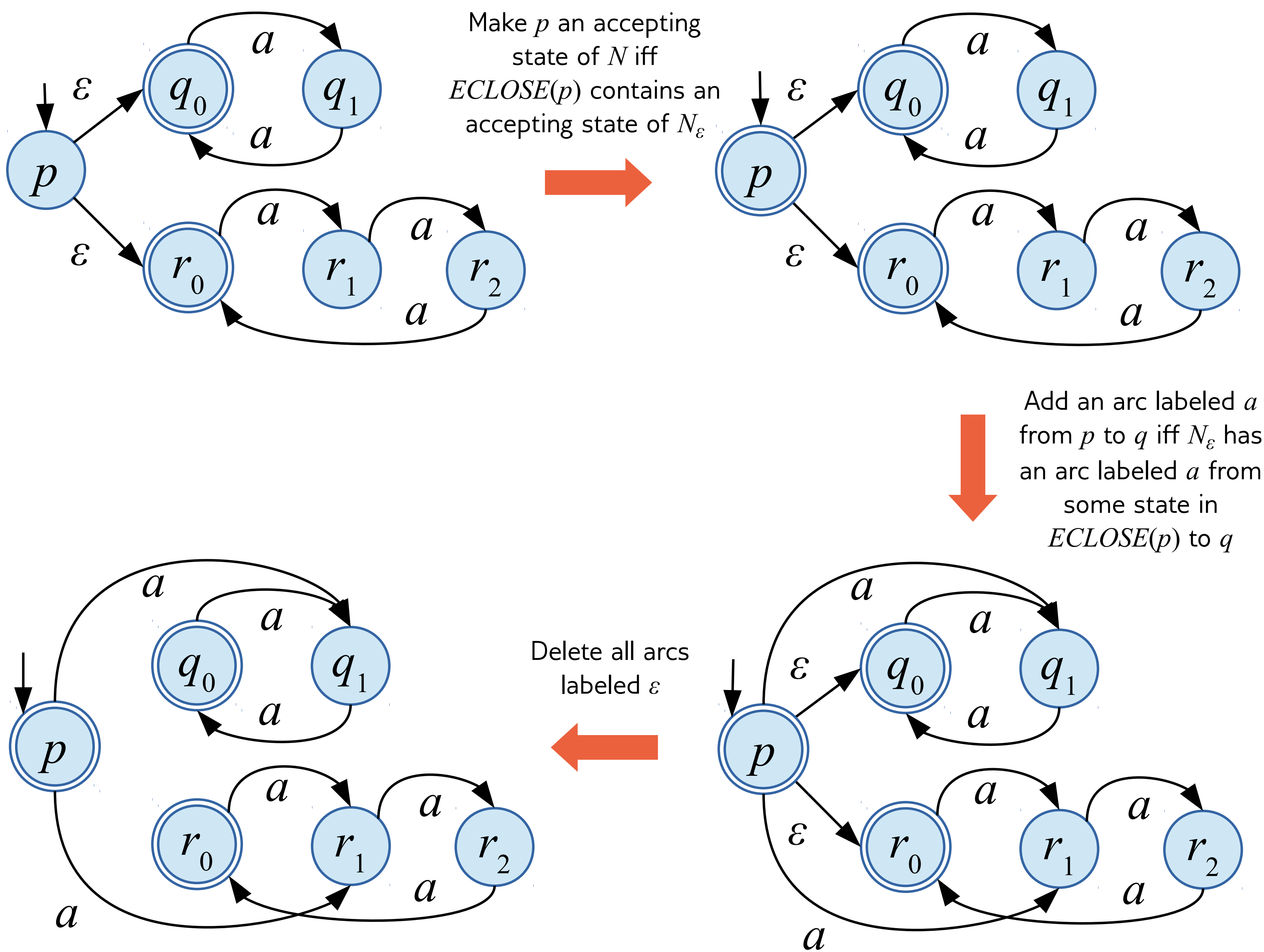


Make p an accepting state of N iff $ECLOSE(p)$ contains an accepting state of N_ε



Add an arc labeled a from p to q iff N_ε has an arc labeled a from some state in $ECLOSE(p)$ to q



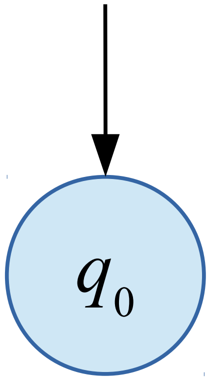

Make p an accepting state of N iff $ECLOSE(p)$ contains an accepting state of N_ε
Add an arc labeled a from p to q iff N_ε has an arc labeled a from some state in $ECLOSE(p)$ to q
Delete all arcs labeled ε
p
q_0
q_1
r_0
r_1
r_2
a
ε

Regular expression to ε-NFA

- Structural induction on regex
  - Construct simple automata for base cases
  - For every higher-order construction, construct equivalent ε-NFA from smaller ε-NFAs

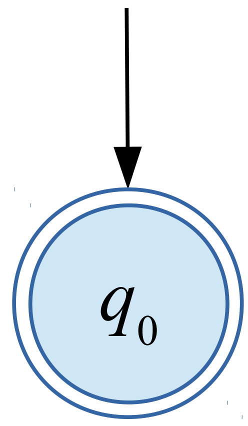
Empty set

Regex: $\varnothing$



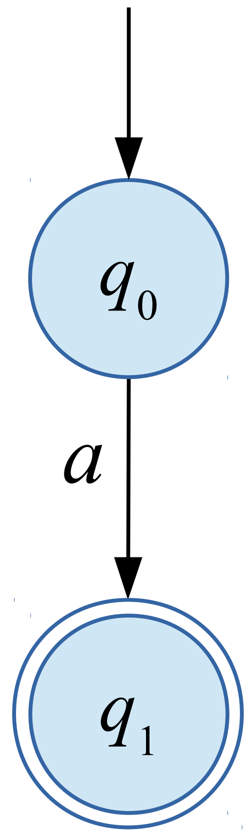
Empty string

Regex: $\boldsymbol{\varepsilon}$



Literal character

Regex: **a**

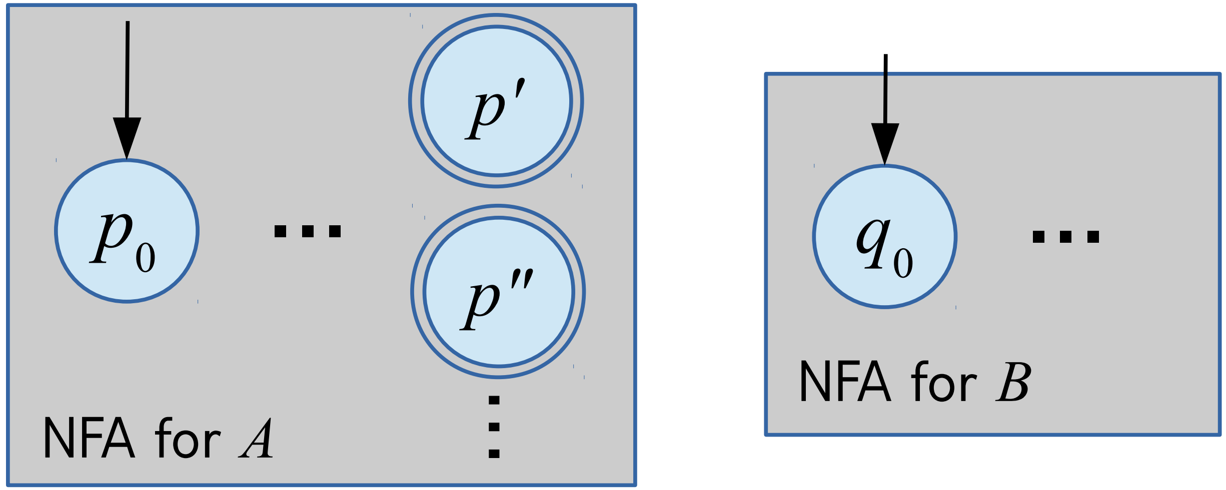


Concatenation

Regex: AB

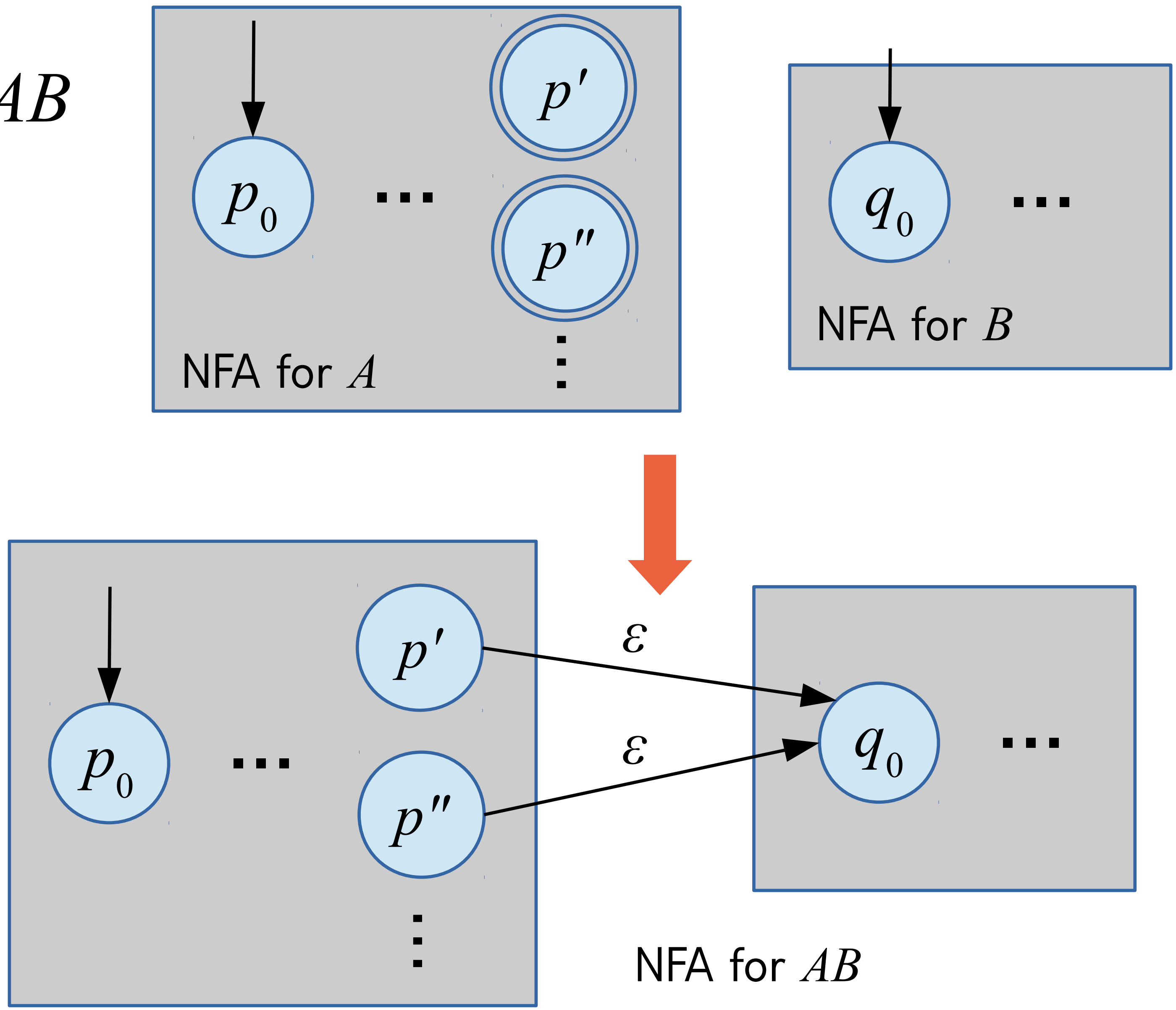
Concatenation

Regex: AB



Concatenation

Regex: AB

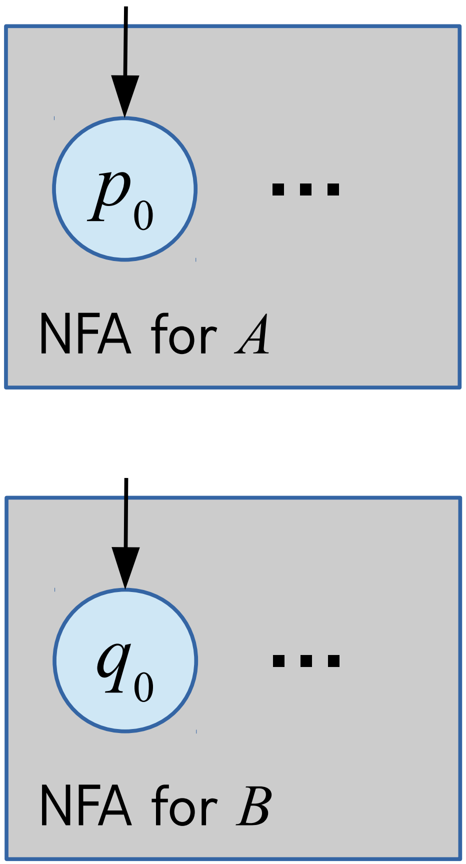


Alternation

Regex: $A|B$

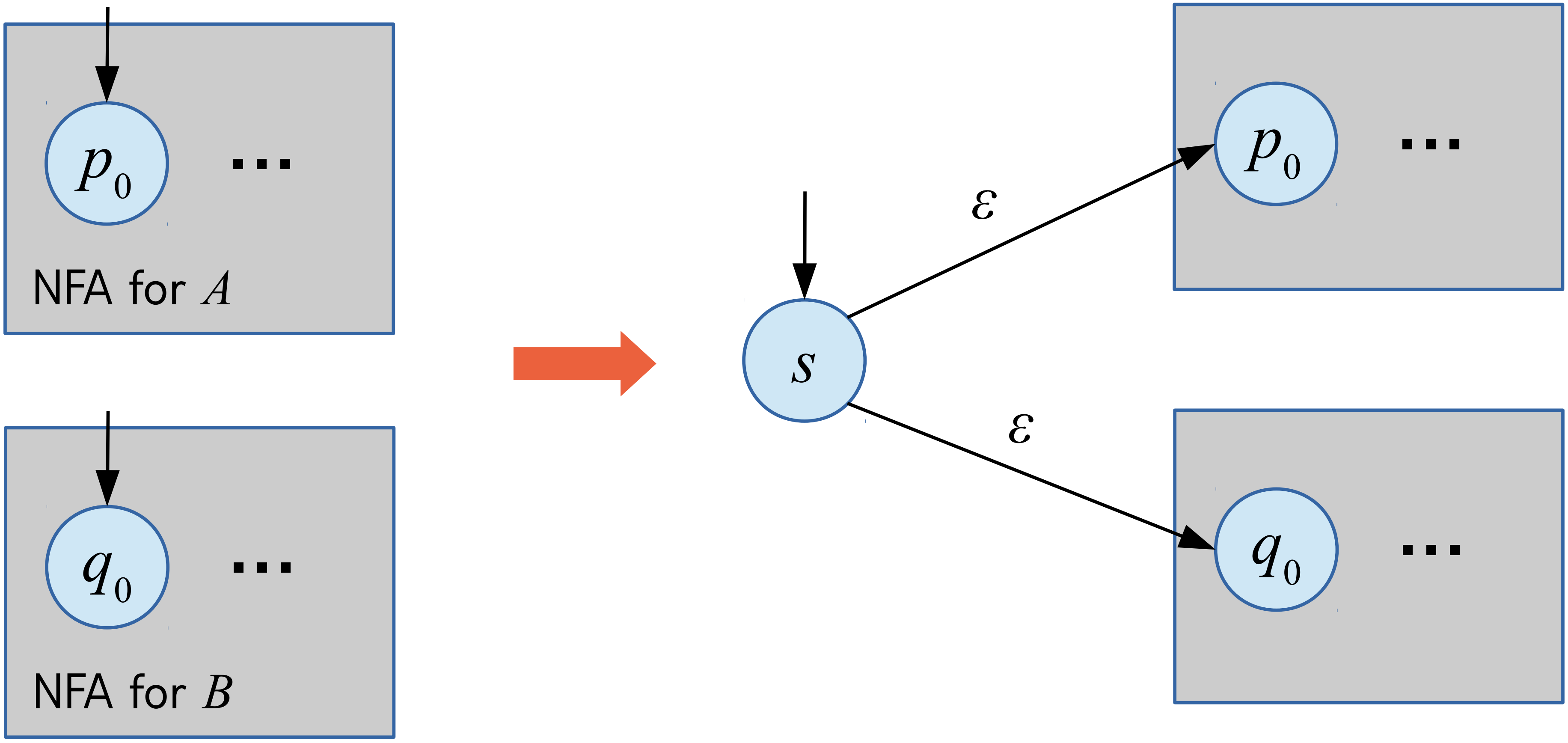
Alternation

Regex: $A|B$



Alternation

Regex: $A|B$



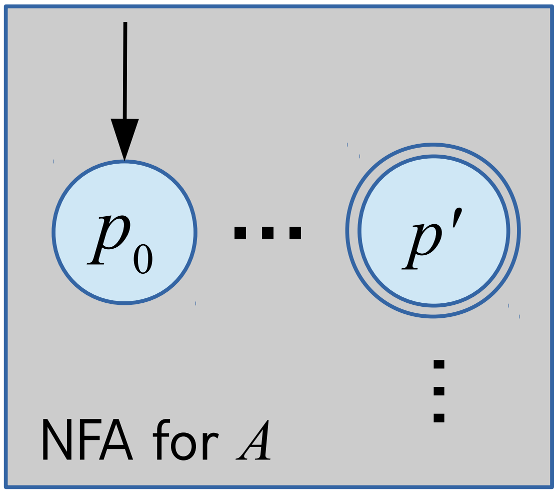
NFA for $A|B$

Kleene star

Regex: $A*$

Kleene star

Regex: $A*$



Kleene star

Regex: $A*$