Chapter 2
Scanning

한양대학교 컴퓨터공학부

컴파일러

2014년 2학기

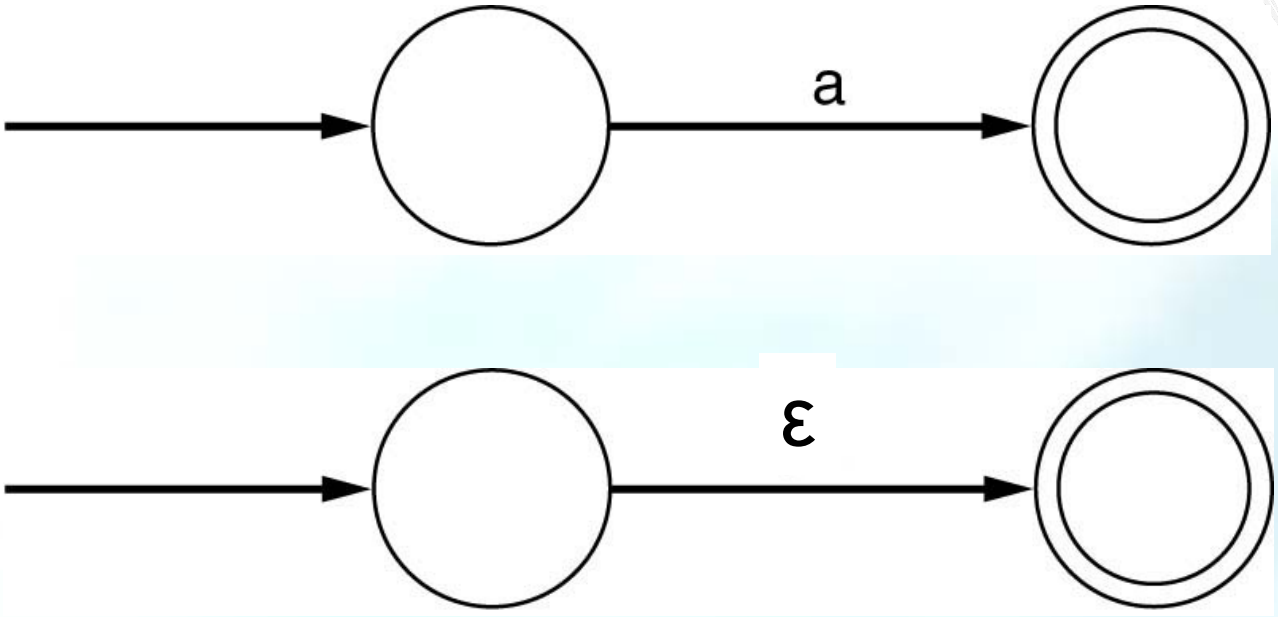
Scanner Construction

Regular expression → NFA

- **Thompson's construction**
- Regular expressions
  - Basic regular expressions
  - Concatenation
  - Choice
  - Repetition

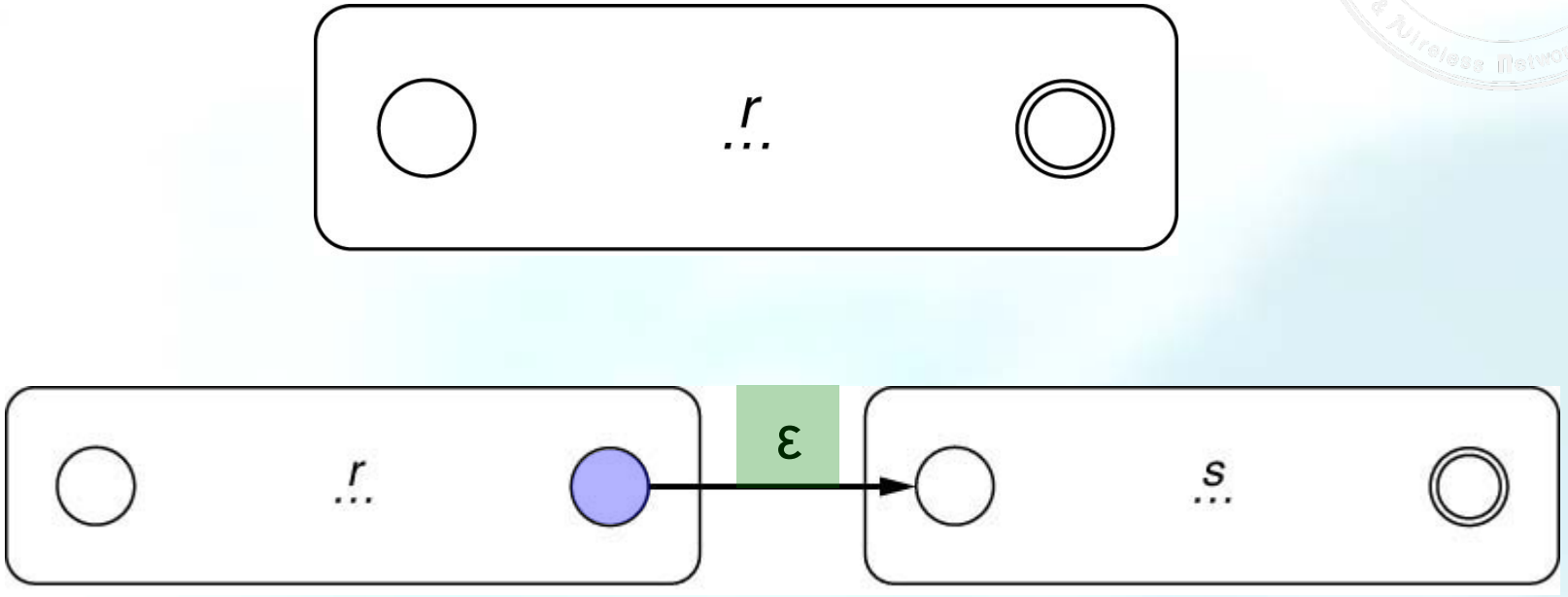
Regular expression → NFA

- Basic regular expressions
  - a, ε, **Φ**



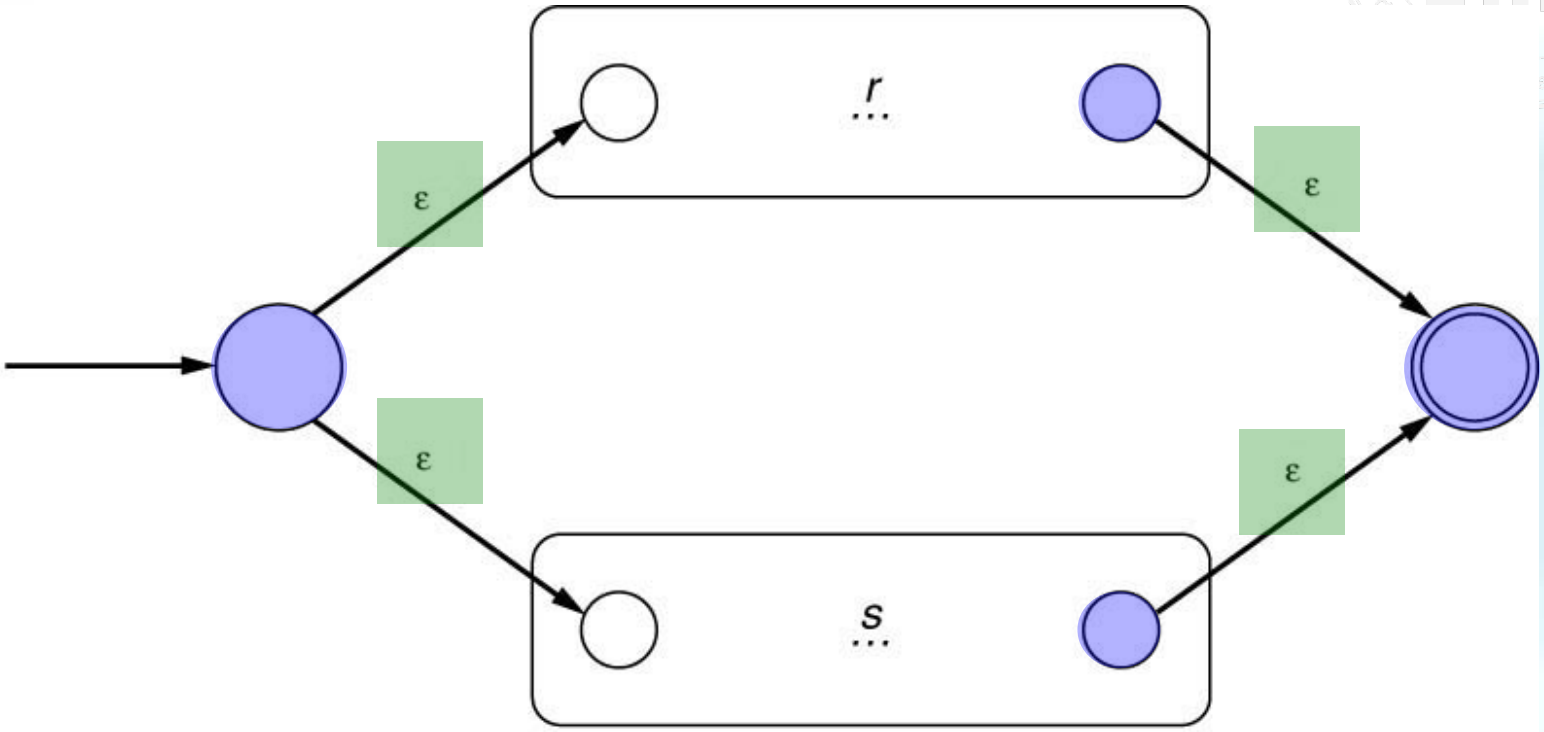
Regular expression → NFA

- Concatenation
  - *rs*



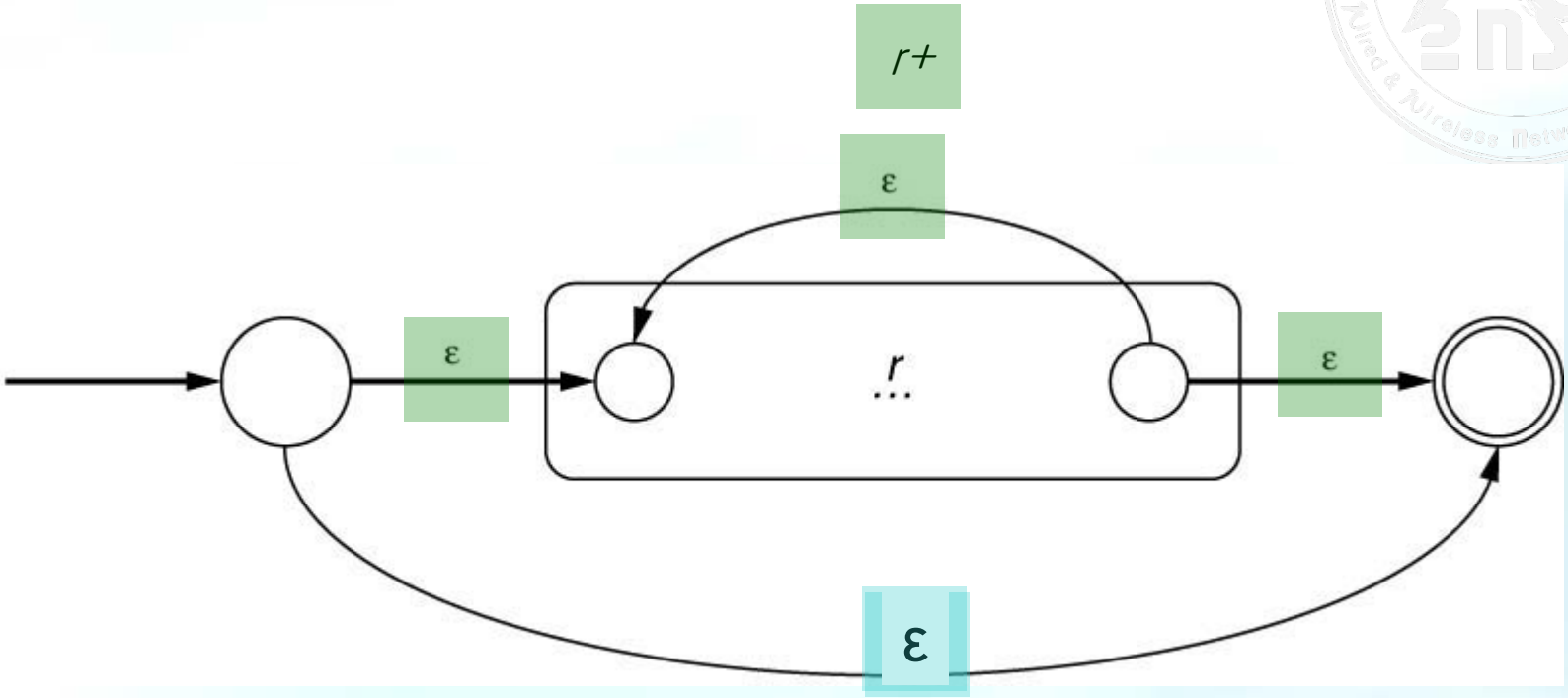
Regular expression → NFA

- Choice
  - $r|s$



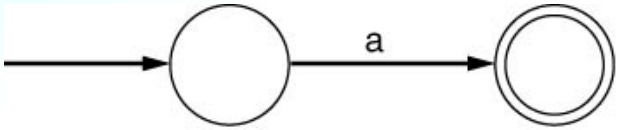
Regular expression → NFA

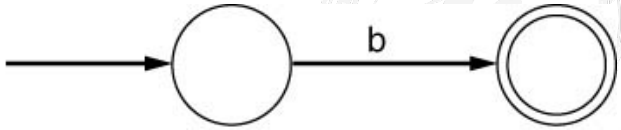
- Repetition
  - r^*

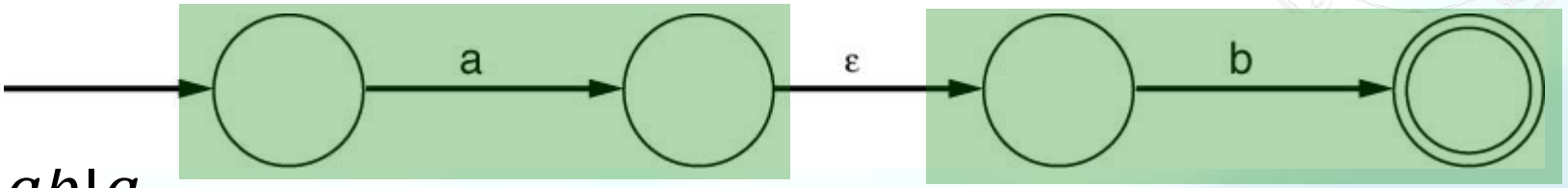


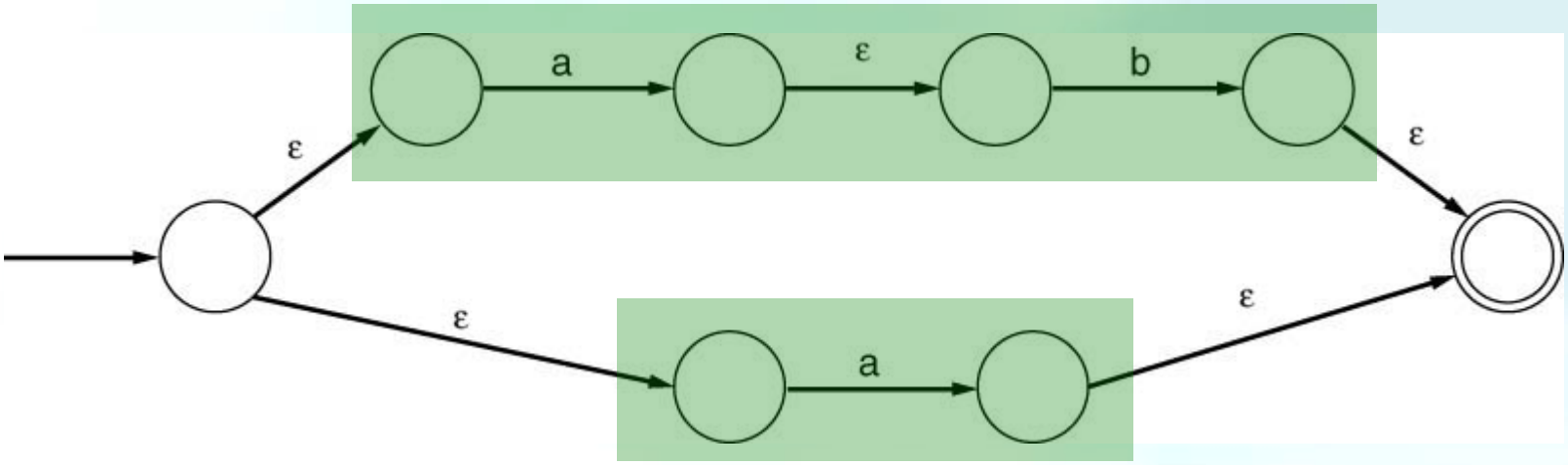
Regular expression → NFA

- $ab|a$
  - a, b
  - ab
  - $ab|a$



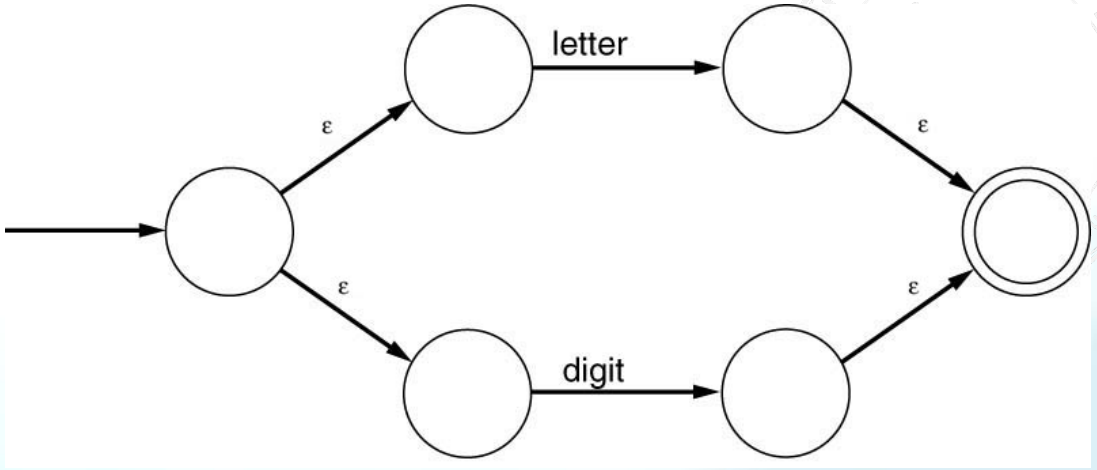




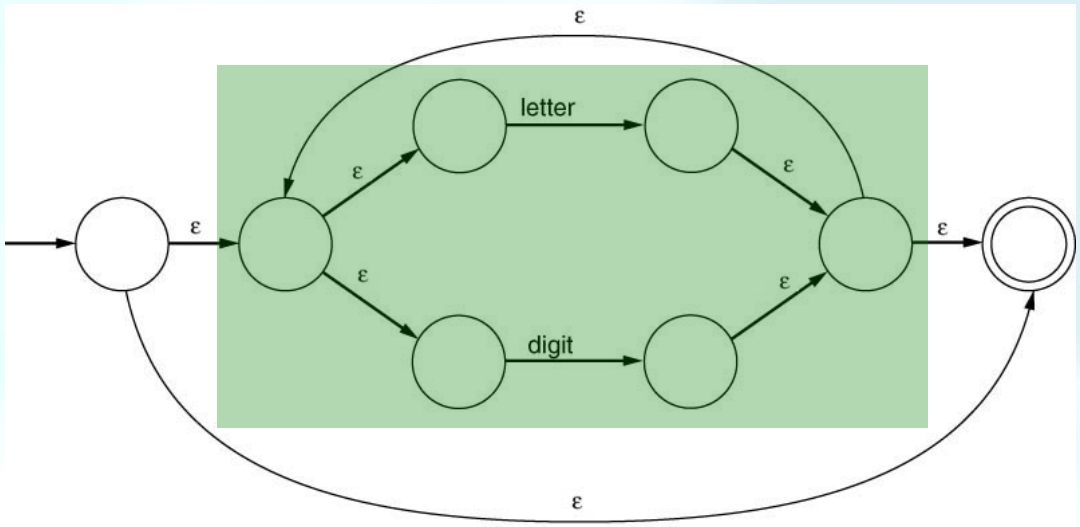


Regular expression → NFA

- *letter*(*letter*|*digit*)*
  - *letter*|*digit*

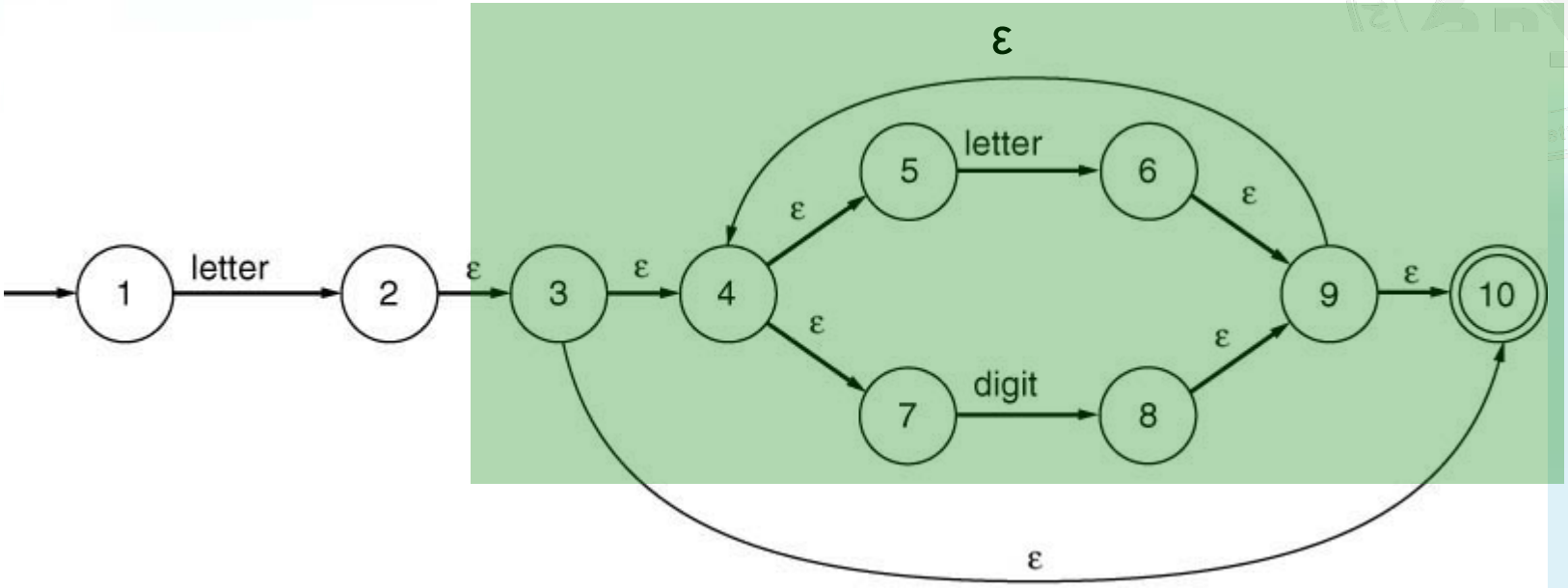


  - (*letter*|*digit*)*



Regular expression → NFA

- *letter*(*letter*|*digit*)*



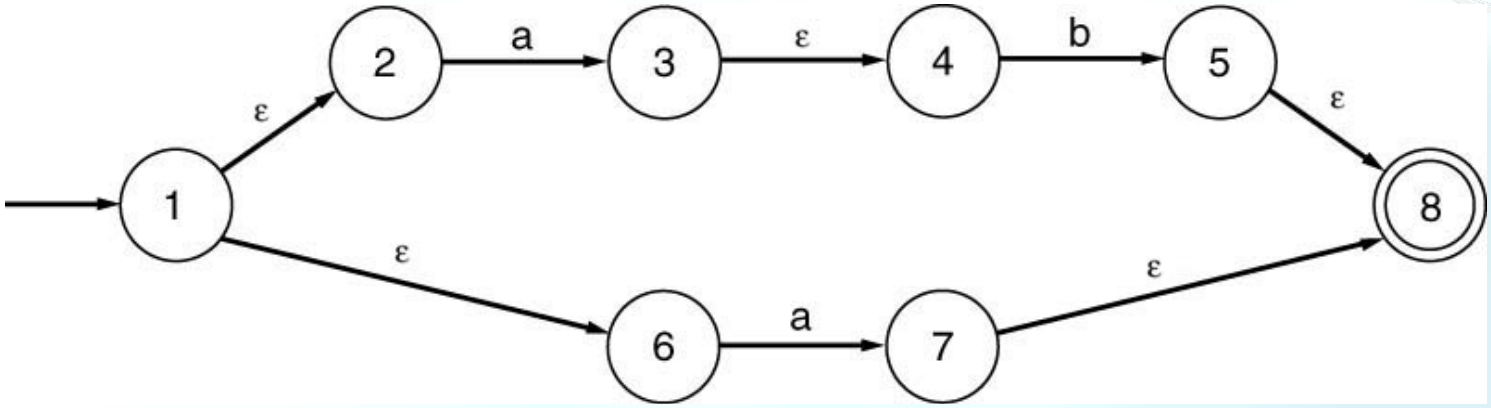
Examples

- (a | b)* a

- (a | bb)*

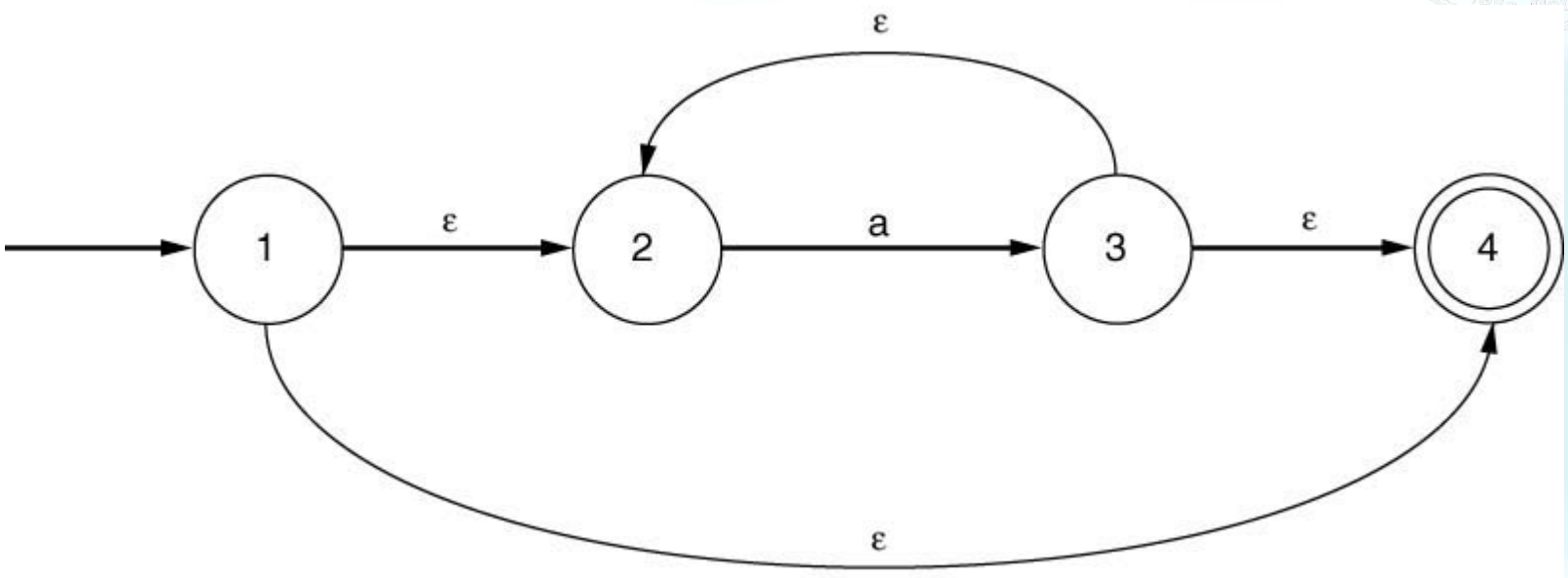
NFA → DFA

- The **subset construction** with ε – transitions.



NFA → DFA

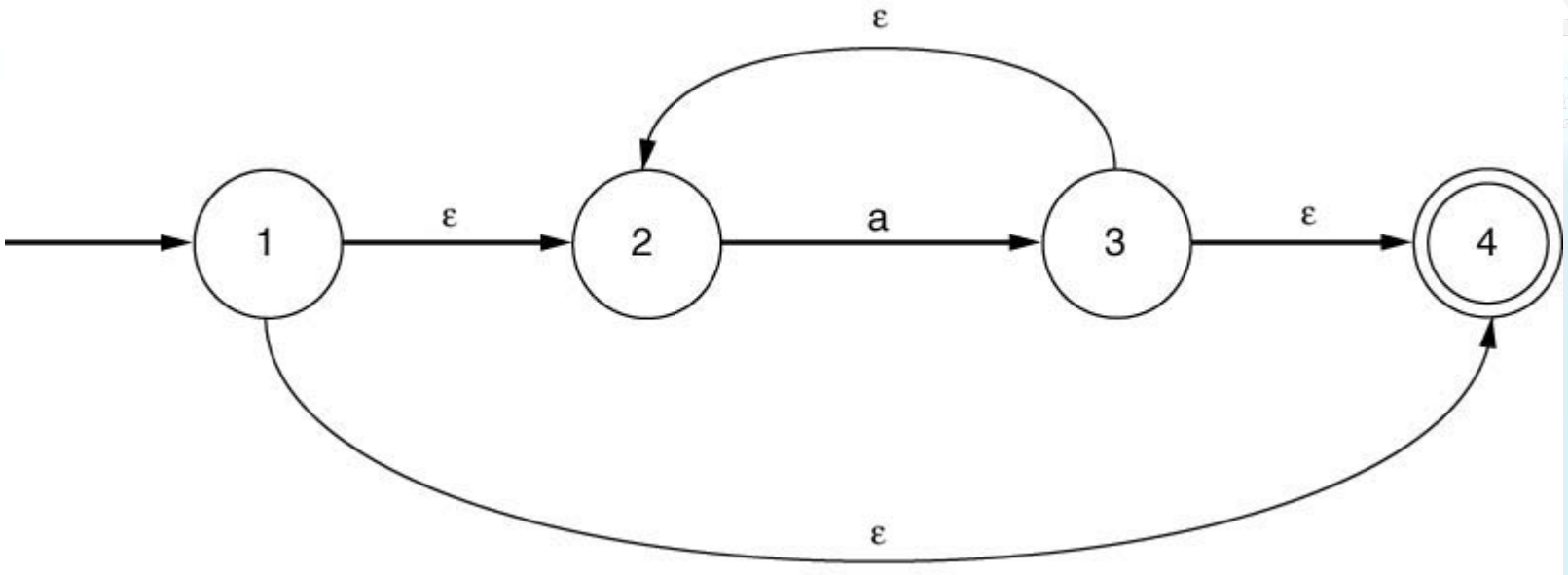
- ε – closure of a state
  - The ε – closure of a single state *s*, denoted by *s*, is the set of states reachable by only zero or more ε-transitions.



$\overline{1}$ = { } $\overline{2}$={ } $\overline{3}$={ } $\overline{4}$={ }

NFA → DFA

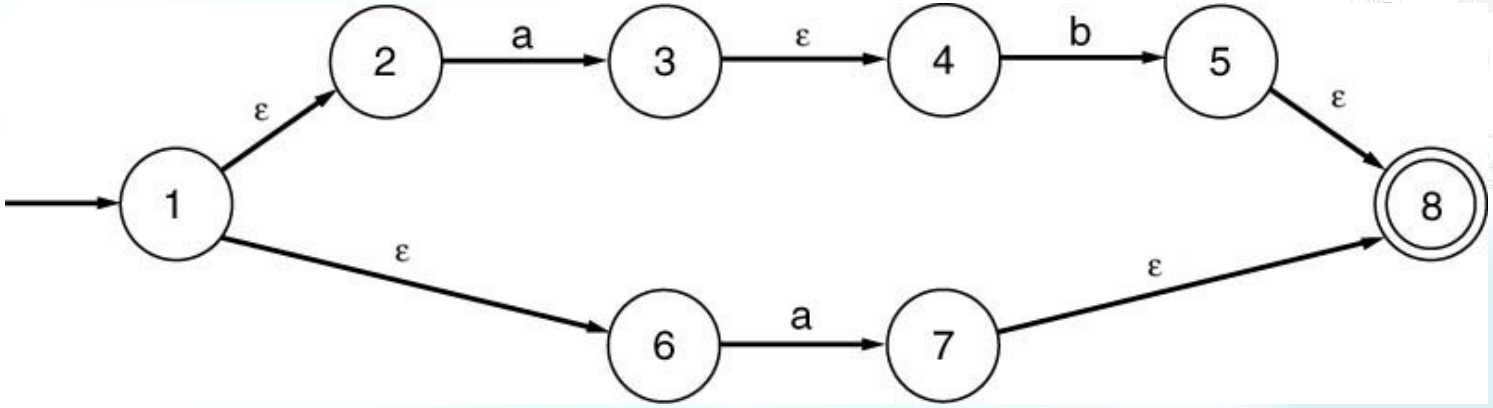
- ε – closure of some states
  - The union of the ε – closures of each state.



$$\overline{\{1,3\}} = \overline{1} \cup \overline{3} = \{1,2,4\} \cup \{2,3,4\} = \{1,2,3,4\}$$

NFA → DFA

- The Subset construction



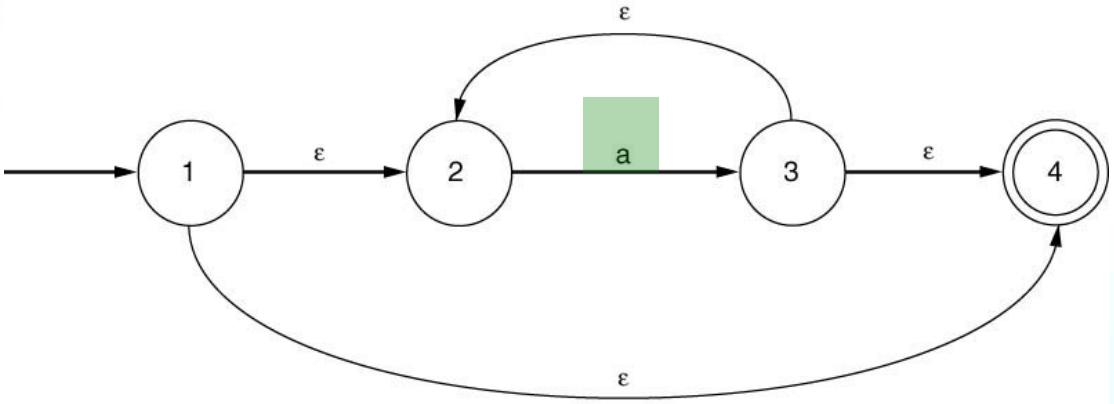
string: ab

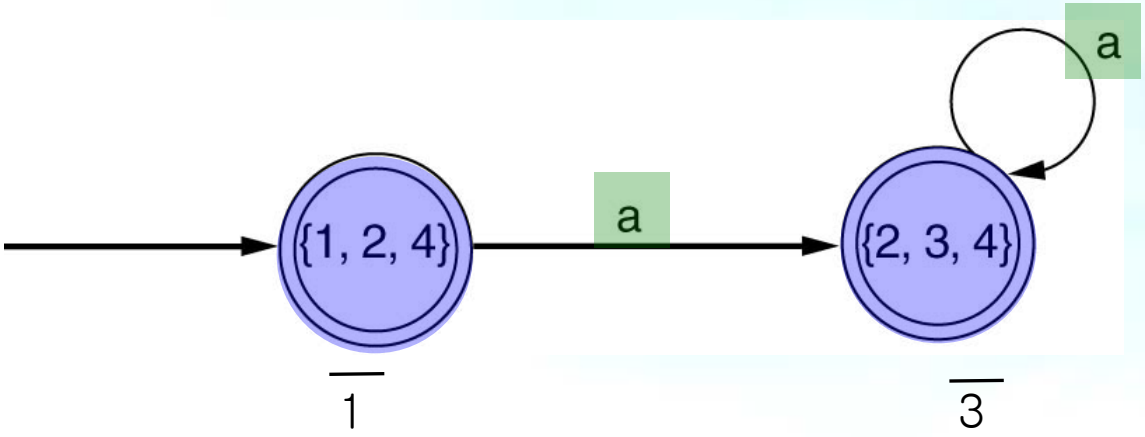
$\overline{1}$ = {1,2,6}, $\overline{2}$ = {2}, $\overline{3}$={3,4}, $\overline{4}$={4}, $\overline{5}$={5,8}, $\overline{6}$={6}, $\overline{7}$={7,8}, $\overline{8}$={8}

$\overline{\{1\}}$ = {1,2,6} —a→ $\overline{\{3,7\}}$ = { } —b→ $\overline{\{5\}}$ = { }

NFA → DFA

- NFA → DFA (a*)





$\overline{1} = \{1,2,4\}$

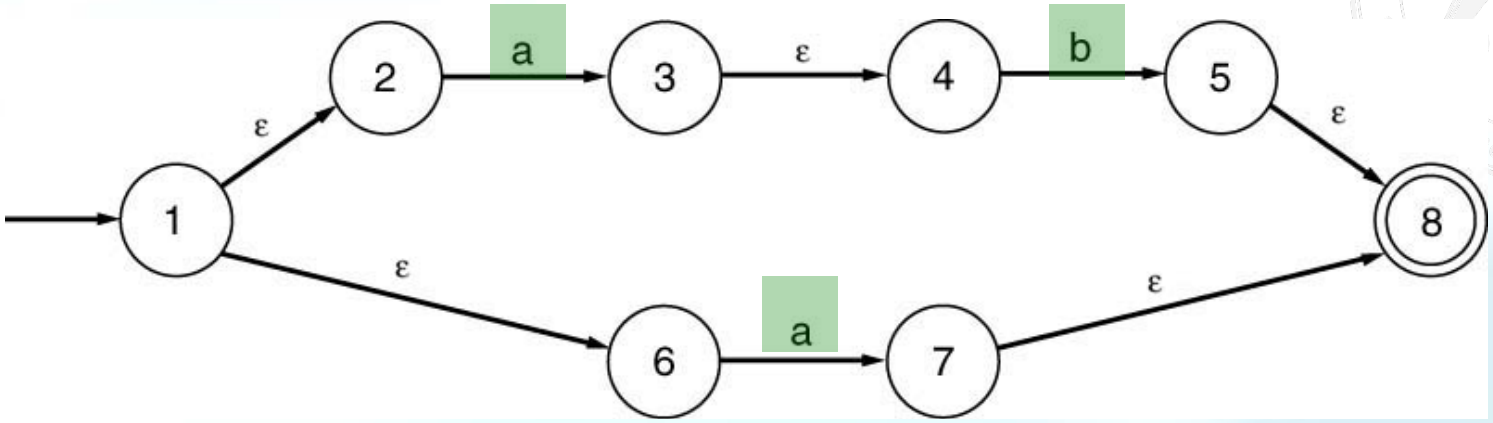
$\overline{2} = \{2\}$

$\overline{3} = \{2,3,4\}$

$\overline{4} = \{4\}$

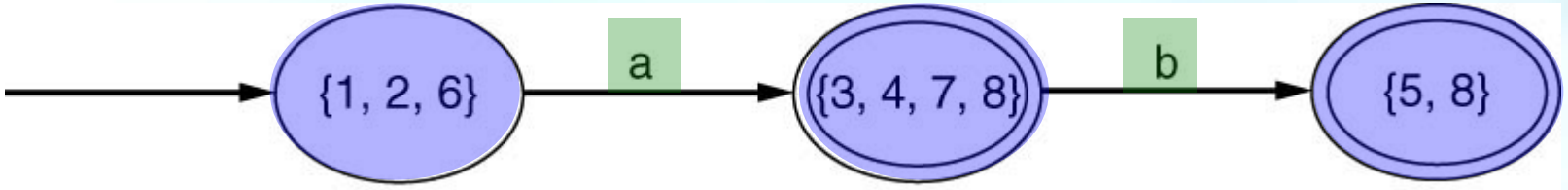
NFA → DFA

- ab|a



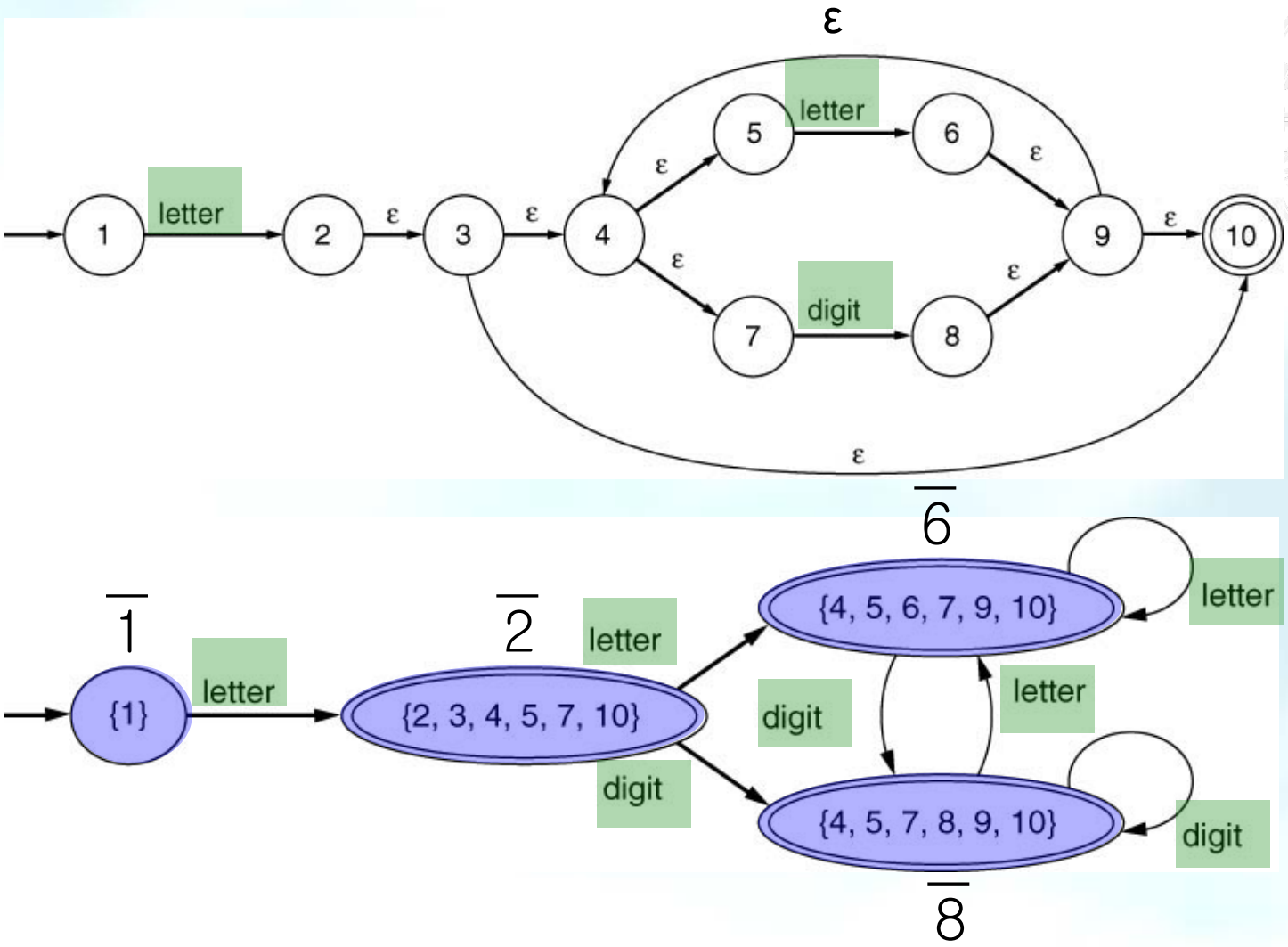
$\overline{1} = \{1,2,6\}$, $\overline{2} = \{2\}$, $\overline{3}=\{3,4\}$, $\overline{4}=\{4\}$, $\overline{5}=\{5,8\}$, $\overline{6}=\{6\}$, $\overline{7}=\{7,8\}$, $\overline{8}=\{8\}$

$\overline{1}$ $\overline{3} \cup \overline{7}$ $\overline{5}$

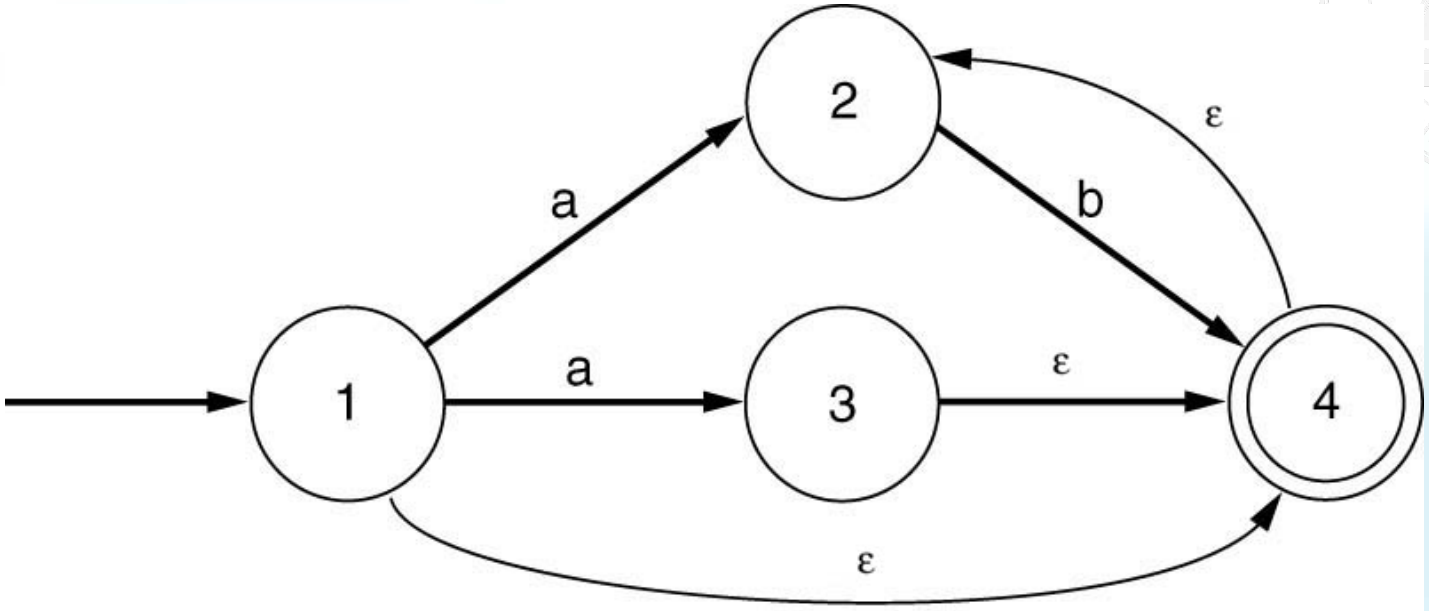


NFA → DFA

- *letter*(*letter*|*digit*)*

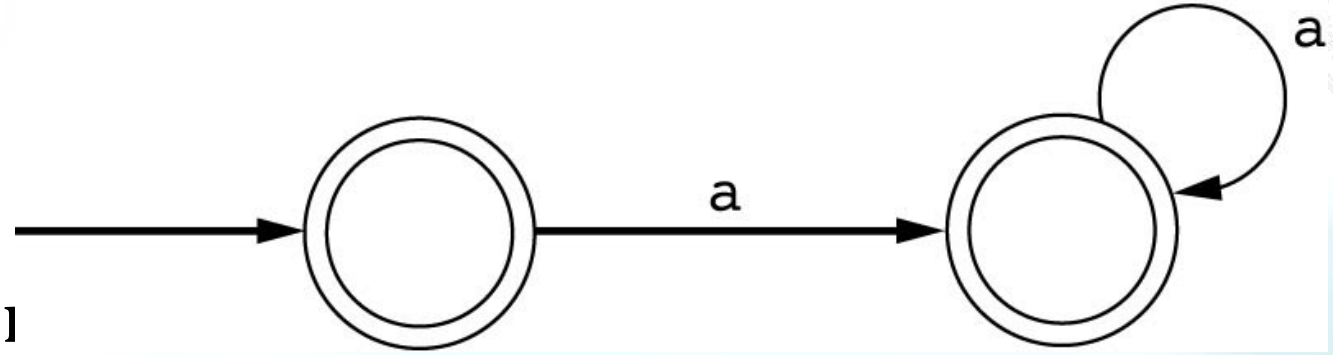


Example 2.10

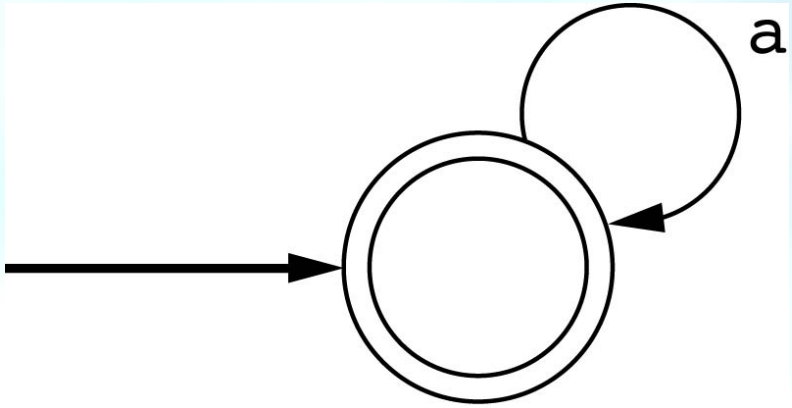


Minimizing DFA

- DFA for a* that is constructed by subset construction

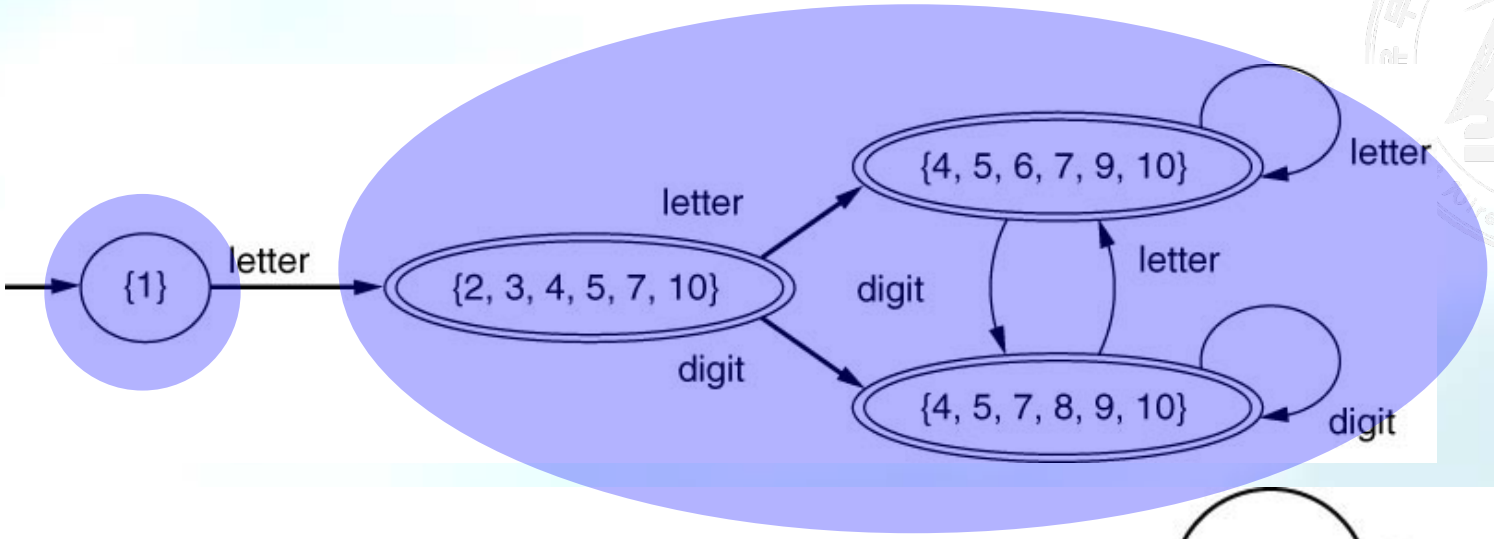


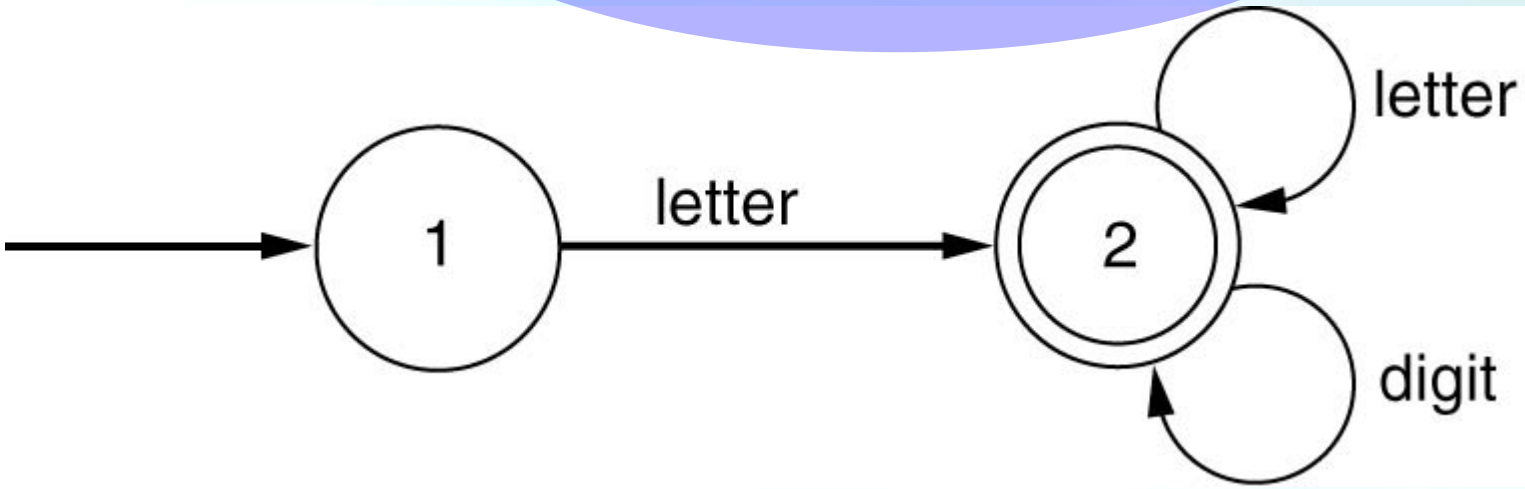
- can



Minimizing DFA

- *letter*(*letter*|*digit*)*



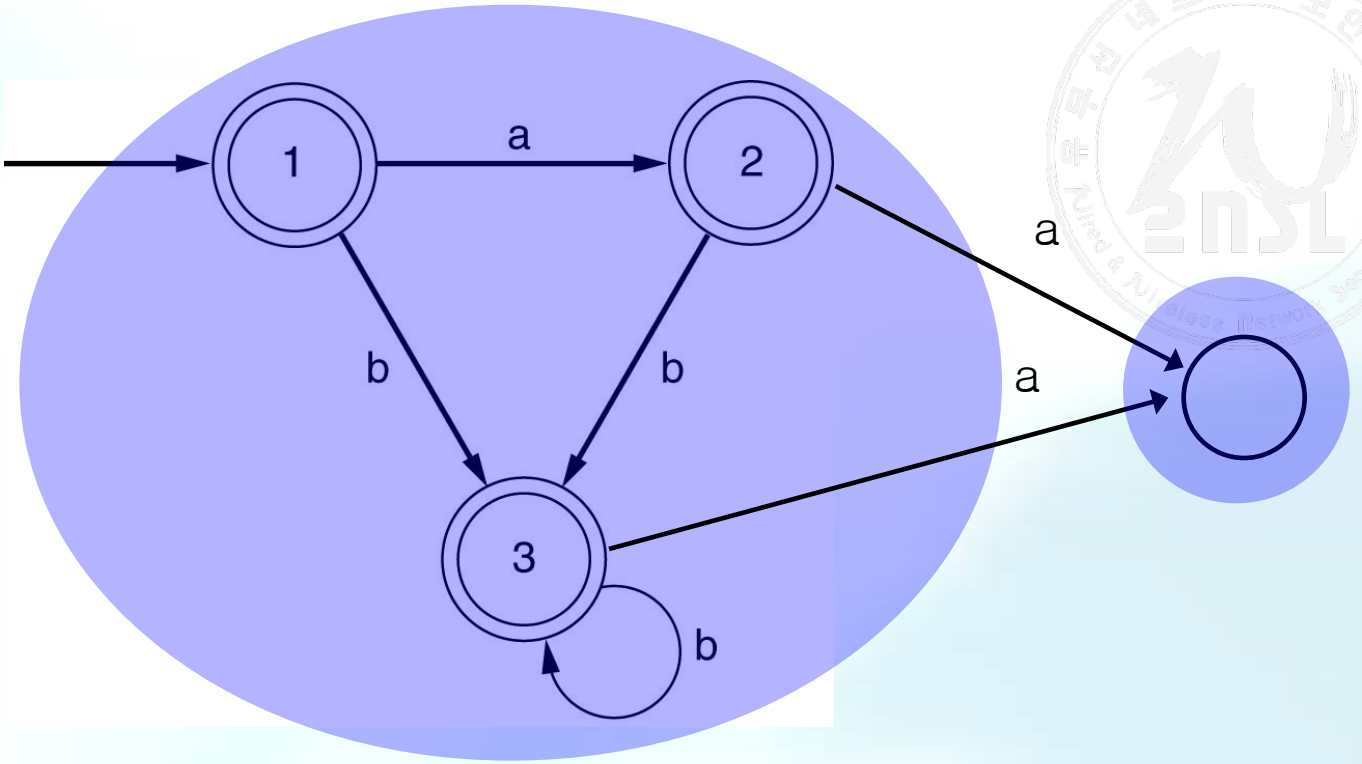


Algorithm

1. unified into single states
    1. one with all the accepting states
    2. the other with all the nonaccepting states
2. Consider transitions on each character *a* of the alphabet
3. if there are two accepting/nonaccepting states that have transitions on *a* that land in different states, *a* **distinguishes** the states and the set of states must be split.

Minimizing DFA

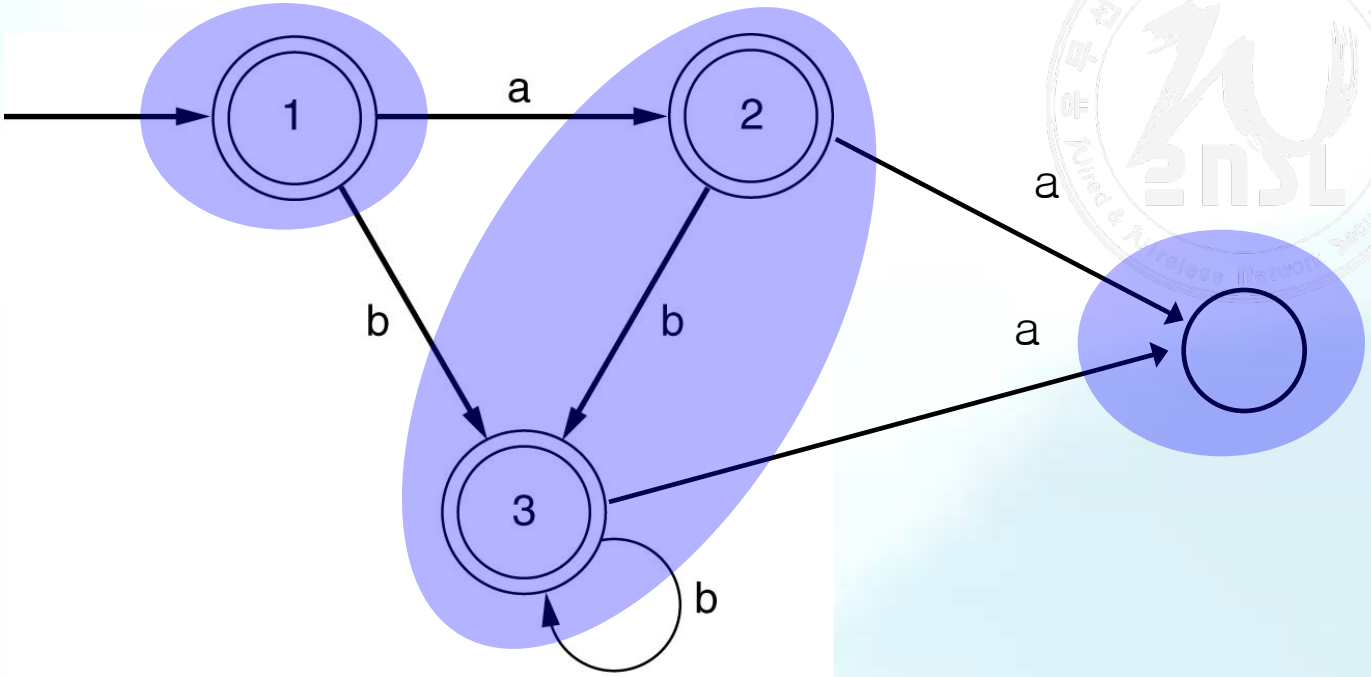
- (a|ε)b*

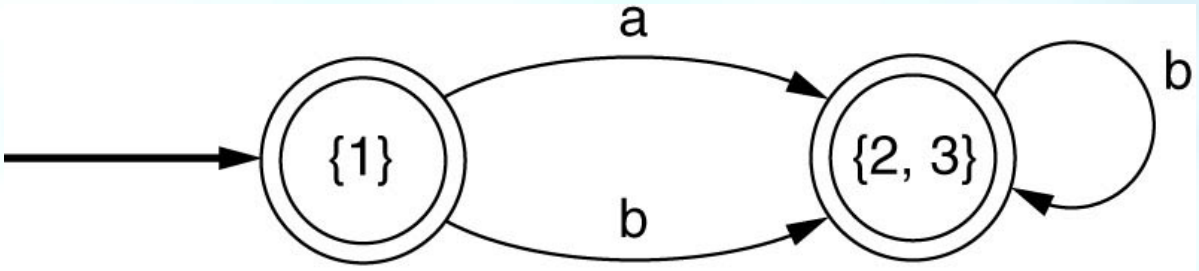


- T(1,a)≠ T(2,a)

Minimizing DFA

- (a|ε)b*





Use of Lex to generate a scanner automatically

- Lex program
  - Input: a text file containing
    - Regular expressions
    - Actions to be taken when each expression is matched
  - Output: C source code (lex.yy.c or lexyy.c)
    - Defining a procedure yylex
      - that is a table-driven implementation of a DFA
      - that operates like a getToken procedure

Metacharacter conventions in Lex

- Table 2.2

Pattern	Meaning
`a`	the character *a*
`"a"`	the character *a*, even if *a* is a metacharacter
`\a`	the character *a* when *a* is a metacharacter
`a*`	zero or more repetitions of *a*
`a+`	one or more repetitions of *a*
`a?`	an optional *a*
`a|b`	*a* or *b*
`(a)`	*a* itself
`[abc]`	any of the characters *a*, *b*, or *c*
`[a-d]`	any of the characters *a*, *b*, *c*, or *d*
`[^ab]`	any character except *a* or *b*
`.`	any character except a newline
`{xxx}`	the regular expression that the name *xxx* represents

Format of a Lex input file

{definitions}

%%

{rules}

%%

{auxiliary routines}

Table 2.3 Some Lex internal names

Lex Internal Name	Meaning/Use
lex.yy.c or **lexyy.c**	Lex output file name
yylex	Lex scanning routine
yytext	string matched on current action
yyin	Lex input file (default: **stdin**)
yyout	Lex output file (default: **stdout**)
input	Lex buffered input routine
ECHO	Lex default action (print **yytext** to **yyout**)

Homework #1

- Exercises
  - 2.1, 2.2, 2.12, 2.13, 2.16
- 주의 사항
  - Handwriting으로 제출
  - Cover page는 생략. 첫 번째 페이지에 homework 번호, 학번, 이름 기입