

# Chapter 3

## 2. 유한 오토마타 : Part I

# 목차

01 형식언어

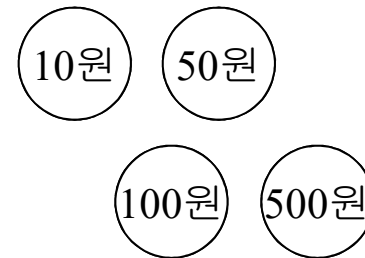
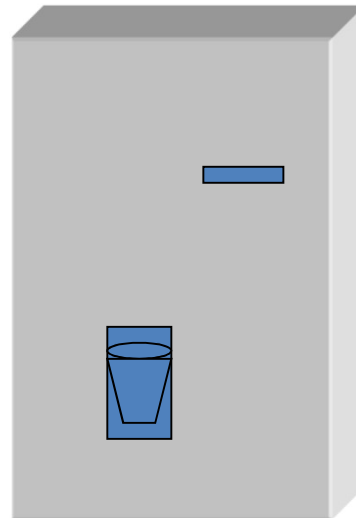
02 정규 표현

**03 유한 오토마타**

# 커피 자판기는 어떻게 동작할까?

---

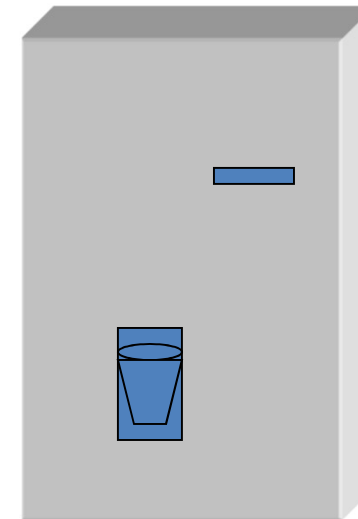
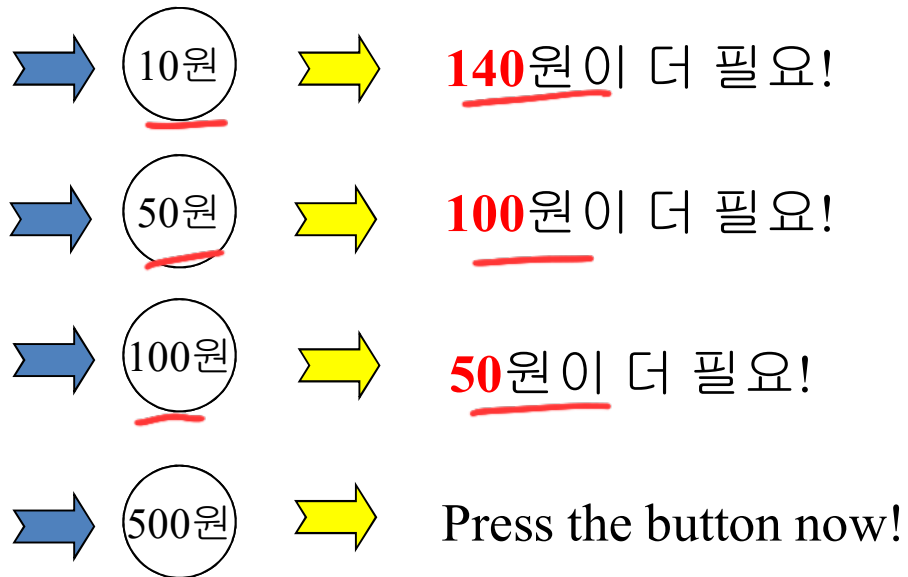
*coffee vending machine*



밀크 커피 1잔=150원

# 자판기는 입력에 따라 상태가 바뀐다

*coffee vending machine*



밀크 커피  
1잔=150원

# Finite State Automata

---

## ■ A model of computation

- **A set of *states*** and **how to get** from some state to other states

## ■ Computation model 이란?

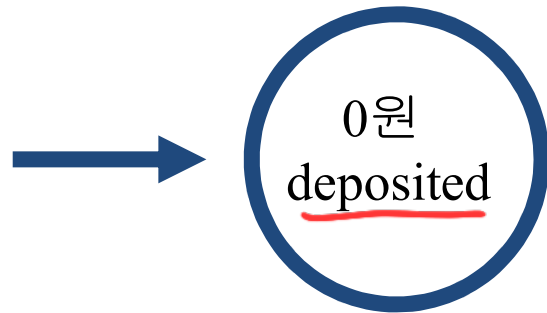
- 시스템 내부를 상세히 만들지 않았지만
  - S/W를 사용하여 시스템 동작을 확인

Automata 는 복수, Automaton 은 단수

## An initial state

### ■ Start with 0원 deposited.

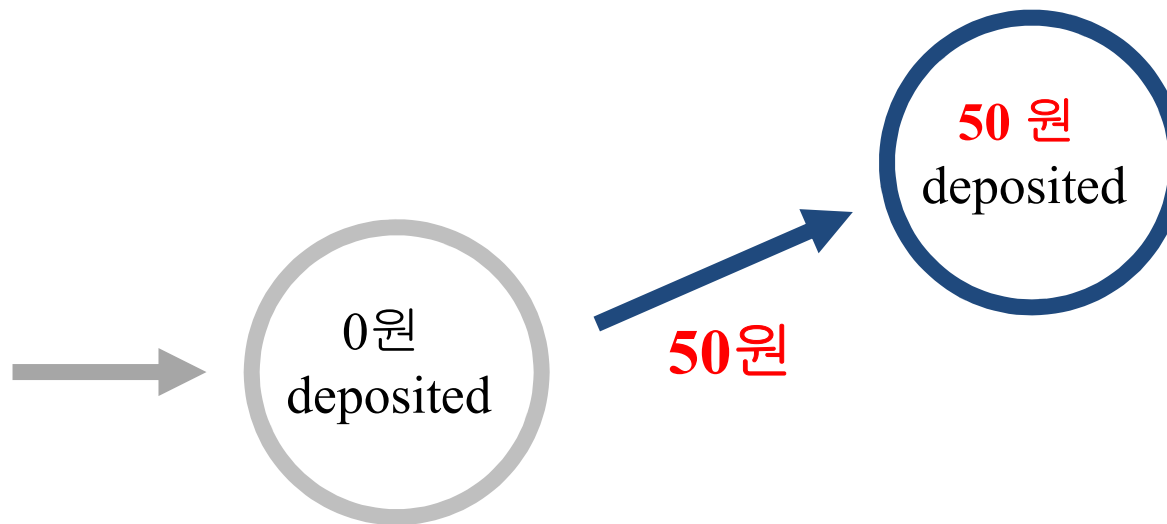
- Called the **initial state** of the machine.
- A circle is drawn for the state



- An arrow pointing to this state marks it as the initial state.

## Depositing a 50원...

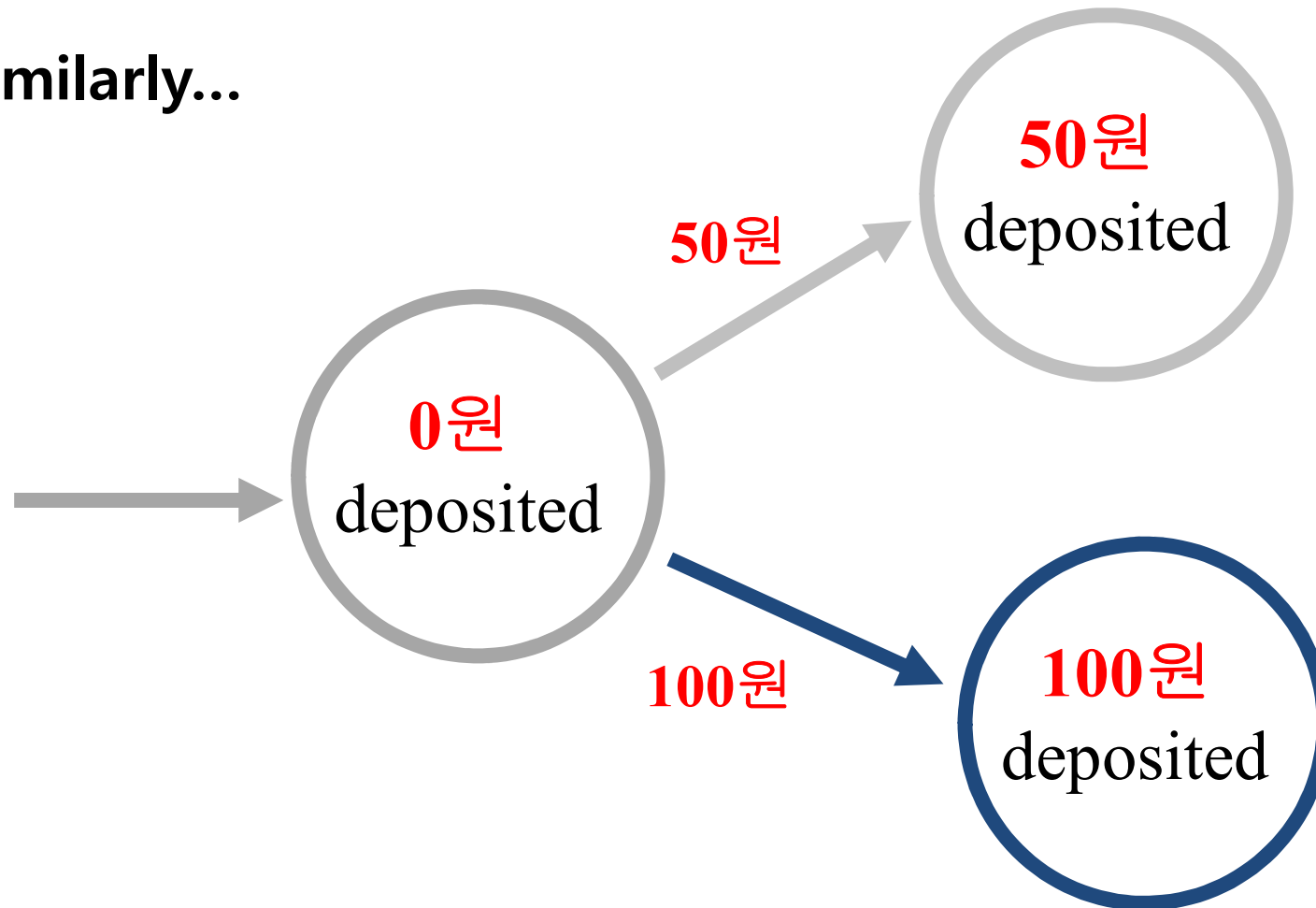
- Takes us to a new state: 50원 deposited
- Add an arrow for the transition
  - Label it with the **input** we gave to the machine



## Depositing a 100원...

---

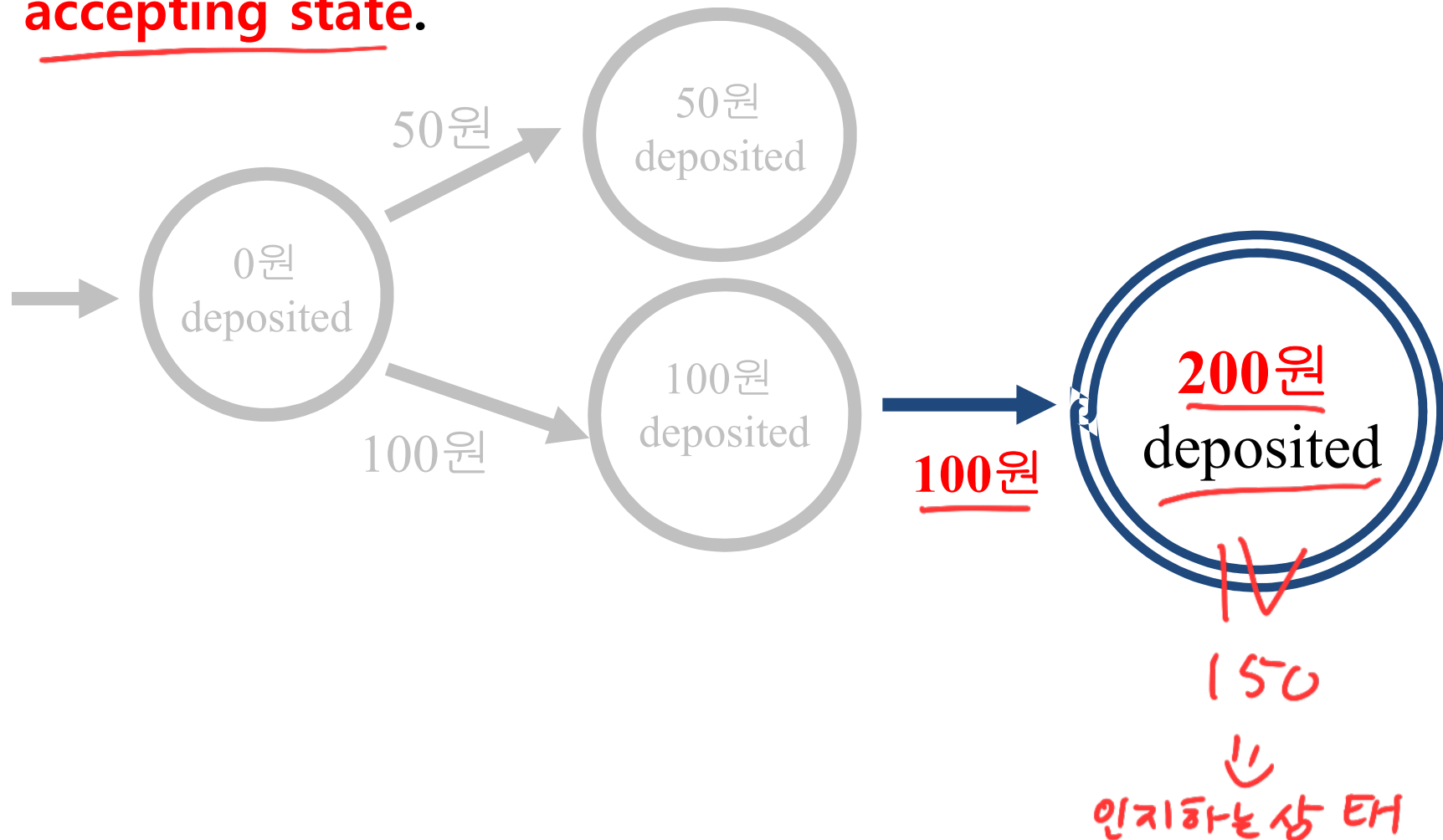
### ■ Similarly...





## Accepting states (or Final States)

- Add new state with double circle to denote that it's an accepting state.



# How to are tokens recognized?

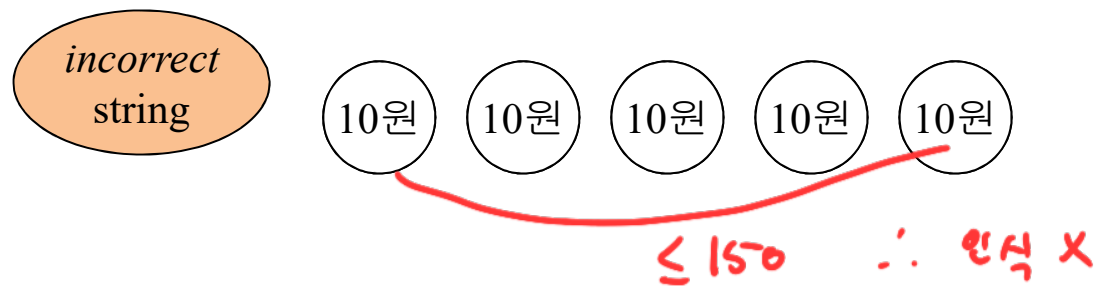
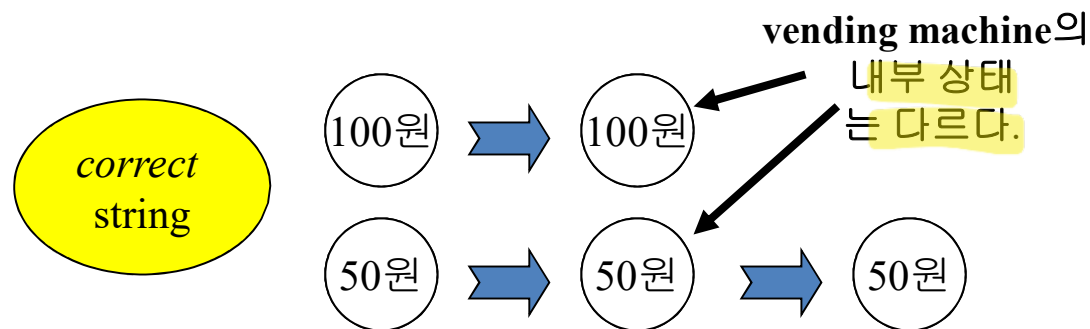
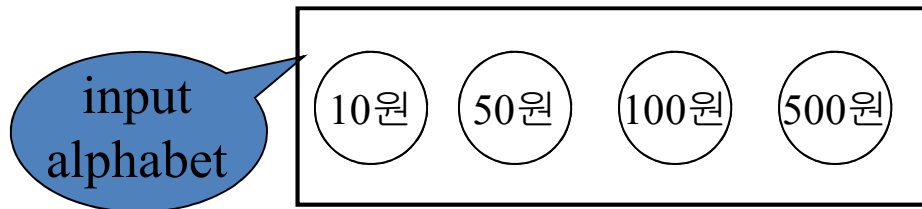
---

## ■ 유한 오토마타(FA, Finite Automata)를 사용

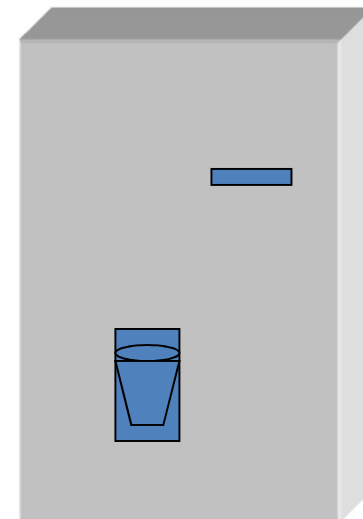
- 유한(finite) 개의 상태(state)를 갖고 있다.
- 입력에 따라 상태가 바뀐다.
  - 윗판의 말은 윗을 던진 결과에 따라 움직인다.
- 시작 상태와 종결 상태가 있다.

✱ 정규 표현은 유한 오토마타(recognizer)로 변환할 수 있다.

# 유한 오토마타 동작 예



coffee vending machine



밀크 커피 1잔=150원

# Finite Automata (1/2)

## ■ 가상 기계(Hypothetical machine)

### ■ 입력 알파벳

- {10원, 50원, 100원, 500원}

### ■ 상태 수가 유한 $\rightarrow$ Finite-*State* Automata

- 상태 : 누적 금액

- 모두 16개의 상태가 필요  $\{S_0, S_1, S_2, \dots, S_{15}\}$

–  $S_0$ (0원),  $S_1$ (10원),  $S_2$ (20원), ...,  $S_{15}$ (150원 이상)  $\rightarrow$  총 16개

$\Downarrow$   
 $2^4$

## Finite Automata (2/2)

### ■ 가상 기계(Hypothetical machine)

- 유한 상태 집합  $\{S_0, S_1, S_2, \dots, S_{15}\}$
- An initial state and final state S  $\Rightarrow$  시작 상태는 하나지만 종결 상태는 여러개 가능
  - 초기 상태( $S_0$ , 0원), 종결 상태( $S_{15}$ , 150원 이상)
- 외부 입력에 따라 상태(state)가 바뀐다.
  - 현재 상태( $S_1$ , 10원)  $\rightarrow$  100원 투입  $\rightarrow$  다음 상태( $S_{11}$ , 110원)

# Deterministic Finite Automata (DFA)

■ **DFA**  $M = ( \Sigma, S, T, s_0, A ) = \text{tuple}$

▪ Input Alphabet :  $\Sigma$

▪ A set of finite states :  $S$

▪ A starting(initial) state :  $s_0 \in S$

▪ A set of accepting(final) states :  $A \subset S$

$\delta$  : 델타로 읽음

▪ state transition functions (상태 전이 함수) :  $T$  (또는  $\delta$ )

$$T : (S \times \Sigma) \rightarrow S$$

$T$  ( 현재 상태, 입력 기호 ) = 다음 상태

$$T(S_i, a) = S_j \quad \text{단, } S_i, S_j \in S, a \in \Sigma$$

# Deterministic Finite Automata

## ■ DFA : 결정적 유한 오토마타

- DFA는 다음 2 가지 조건을 모두 만족해야 한다.
  - $\epsilon$ -transition이 없다.
  - 어떤 상태에서 하나의 입력 기호에 대해 다음 상태는 **하나**  
-  $\delta : S \times \Sigma \rightarrow S$
- 위 조건 중 어느 하나도 만족하지 않으면
  - NFA (비결정적 유한 오토마타)
- $\epsilon$  - transition :  $\epsilon$  에 의한 상태 전이

**DFA** (*Deterministic* Finite Automata)  
**NFA** (*Non-deterministic* FA)

## DFA 예(1/3)

$$M = (\overset{\text{입력 알파벳}}{\Sigma}, \overset{\text{유한 상태 집합}}{\underline{S}}, \overset{\text{상태 전이}}{\overset{\text{시작 상태}}{T}}, \overset{\text{종결 상태}}{\overset{\text{}}{s_0}}, \overset{\text{}}{\overset{\text{}}{A}}) = (\{a, b\}, \{q_0, q_1, q_2\}, \delta, \underline{q_0}, \{\underline{q_2}\})$$

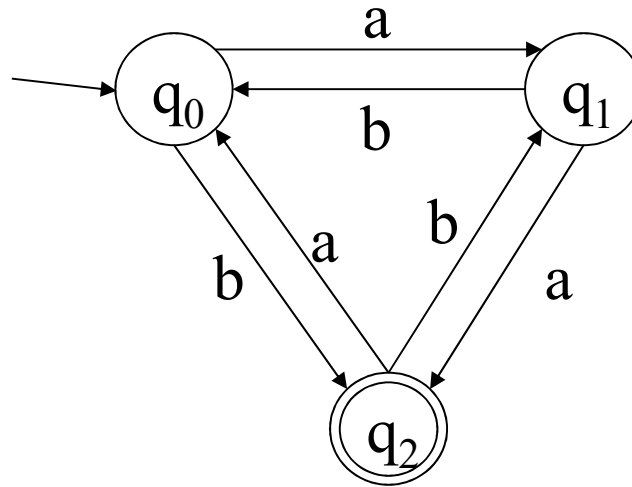
$$\begin{array}{ll} \delta(q_0, a) = q_1 & \delta(q_0, b) = q_2 \\ \delta(q_1, a) = q_2 & \delta(q_1, b) = q_0 \\ \delta(q_2, a) = q_0 & \delta(q_2, b) = q_1 \end{array}$$

상태 전이



## DFA 예(2/3)

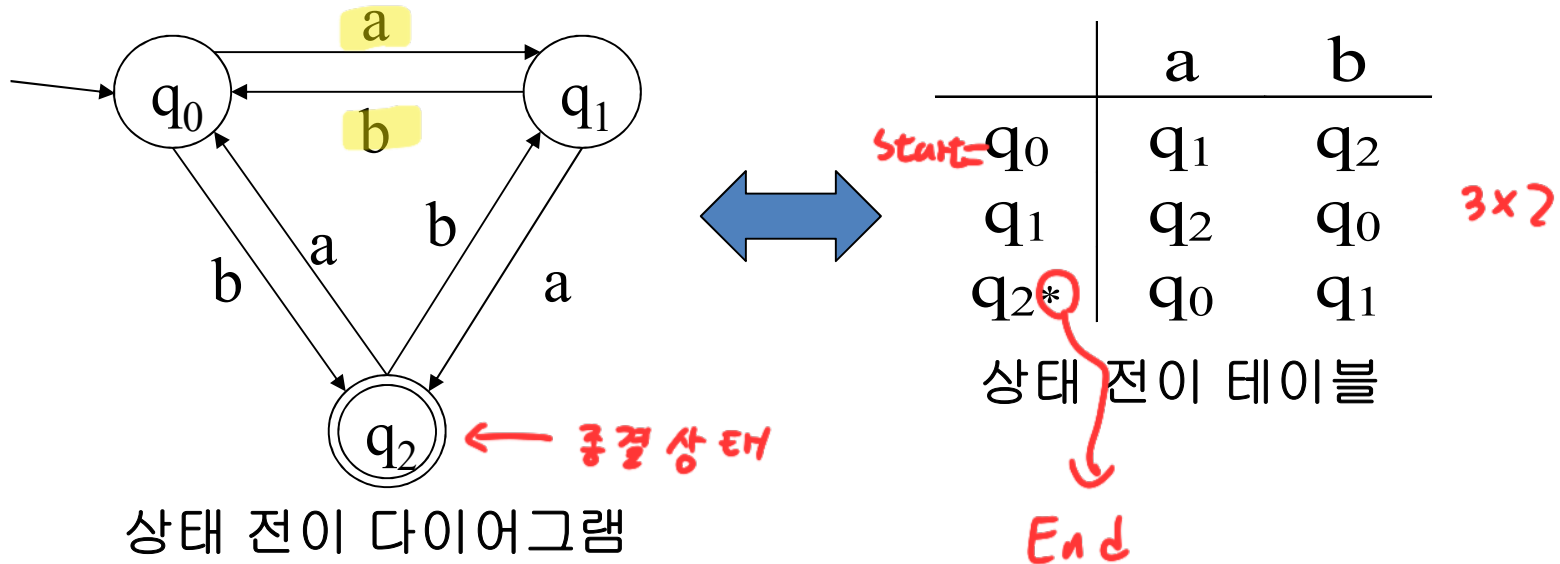
$$M = (\Sigma, S, T, s_0, A) = (\{a, b\}, \{q_0, q_1, q_2\}, \delta, q_0, \{q_2\})$$



상태 전이 다이어그램  
(state transition diagram)

## DFA 예(3/3)

$$M = (\Sigma, S, T, s_0, A) = (\{a, b\}, \{q_0, q_1, q_2\}, \delta, q_0, \{q_2\})$$



## $L(M)$ : DFA $M$ 이 인식하는 언어

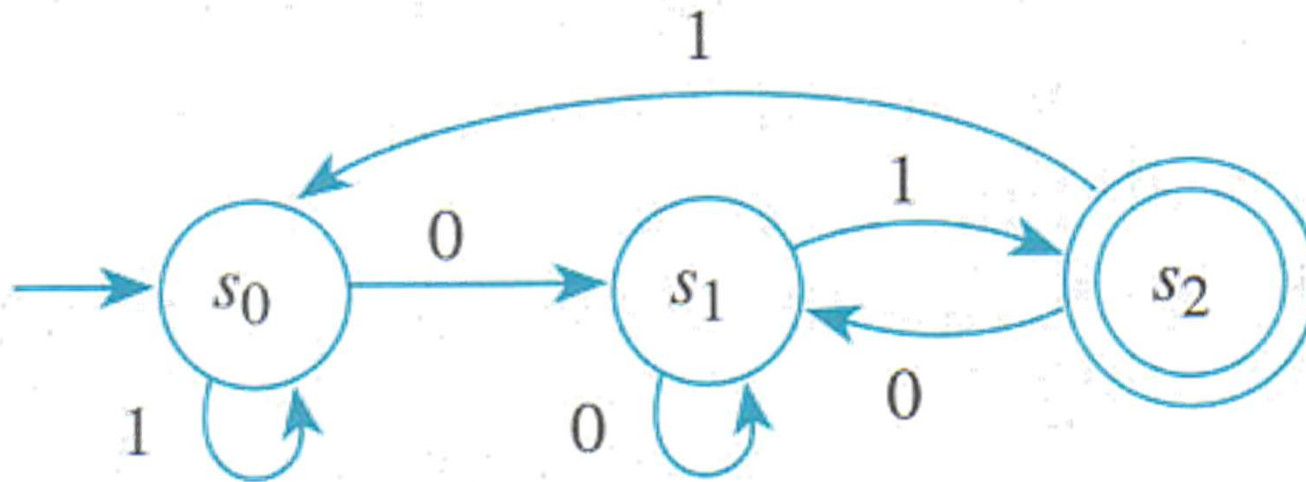
■  $M$  이 인식하는 언어  $L(M)$ 은 아래 조건을 만족하는 문자열  $c_1 c_2 \dots c_n$ 의 집합 ( $c_i \in \Sigma$ )

- $s_1 = \pi(s_0, c_1), s_2 = \pi(s_1, c_2), \dots, s_n = \pi(s_{n-1}, c_n)$ 인 상태들이 존재
- 단,  $s_n$ 은  $A$ 의 원소

## Practice #1 (1/2)

- Initial state?
- Accepting state?

DF4

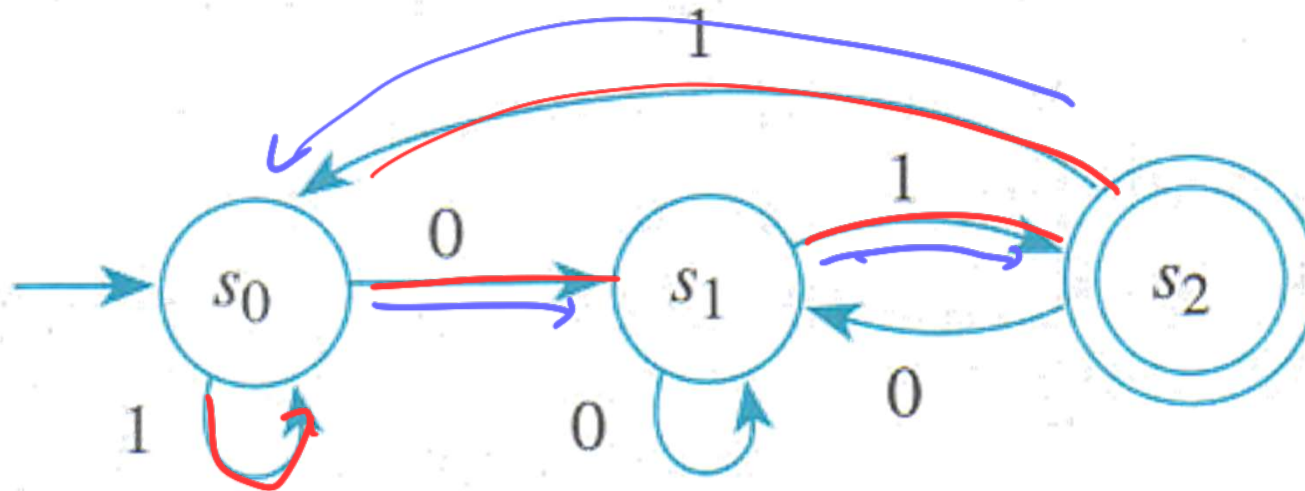


$\Sigma: \{0, 1\}$      $S = \{s_0, s_1, s_2\}$

$L(M) = \{01, 0101, \dots\} \Rightarrow$  이로 끝나는 문자열을 인식하는 automata

## Practice #1 (2/2)

- What happens if we input 011? 인식 X  $s_0$ 로 돌아와 - 선
- What about 0111? - 선 인식 X



## DFA 인식 과정 예(1/2)

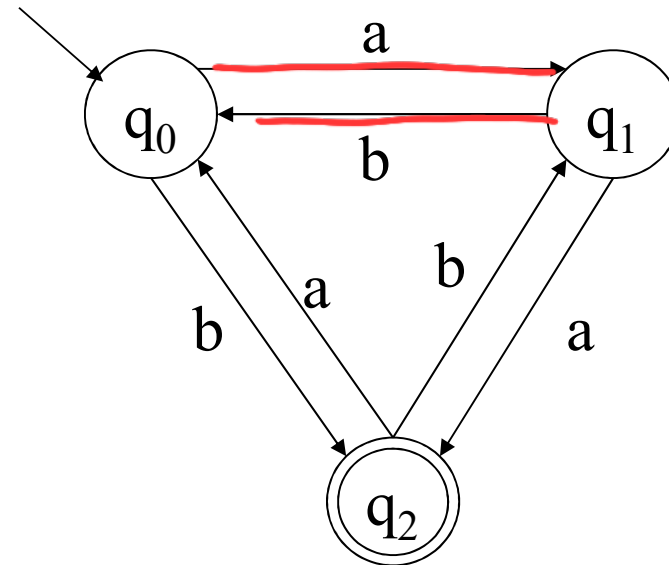
입력 문자열 =  $c_1 c_2 c_3 = aba$

$$\delta(q_0, c_1) = \delta(q_0, a) = q_1,$$

$$\delta(q_1, c_2) = \delta(q_1, b) = q_0,$$

$$\delta(q_0, c_3) = \delta(q_0, a) = q_1 \neq q_2$$

$\therefore$  인식하지 못함 *end: q<sub>1</sub>*



## DFA 인식 과정 예(2/2)

입력 = ababb

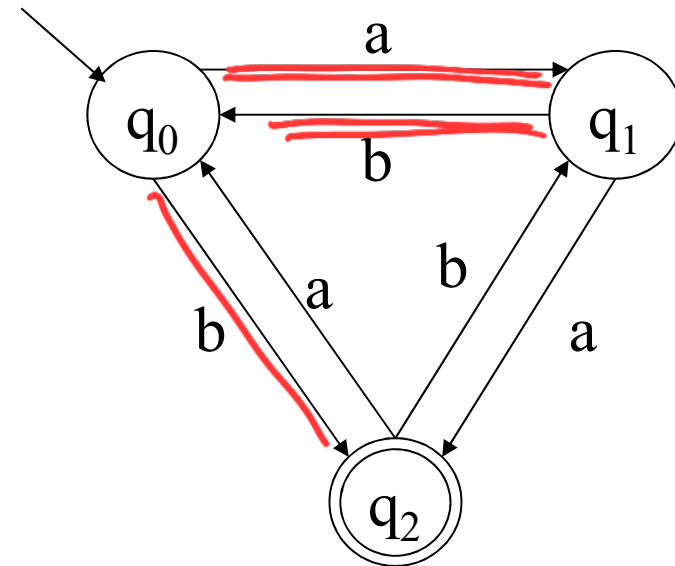
$\delta(q_0, a) = q_1$ ,  $\delta(q_1, b) = q_0$

$\delta(q_0, a) = q_1$ ,  $\delta(q_1, b) = q_0$

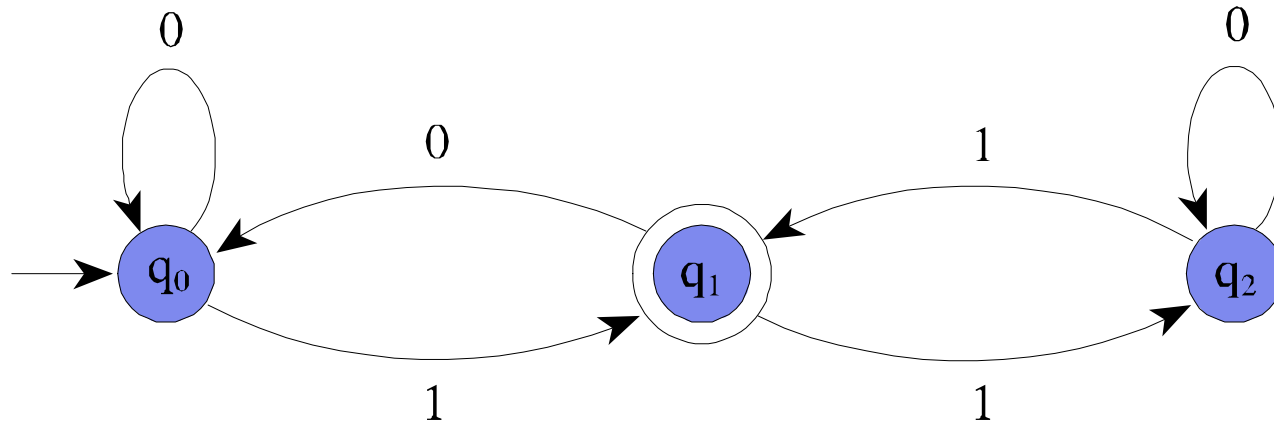
$\delta(q_0, b) = q_2$

$\therefore$  인식

DFA (목적지 하나!)



## Practice #3



Q1. Identify 5 tuple of DFA M.

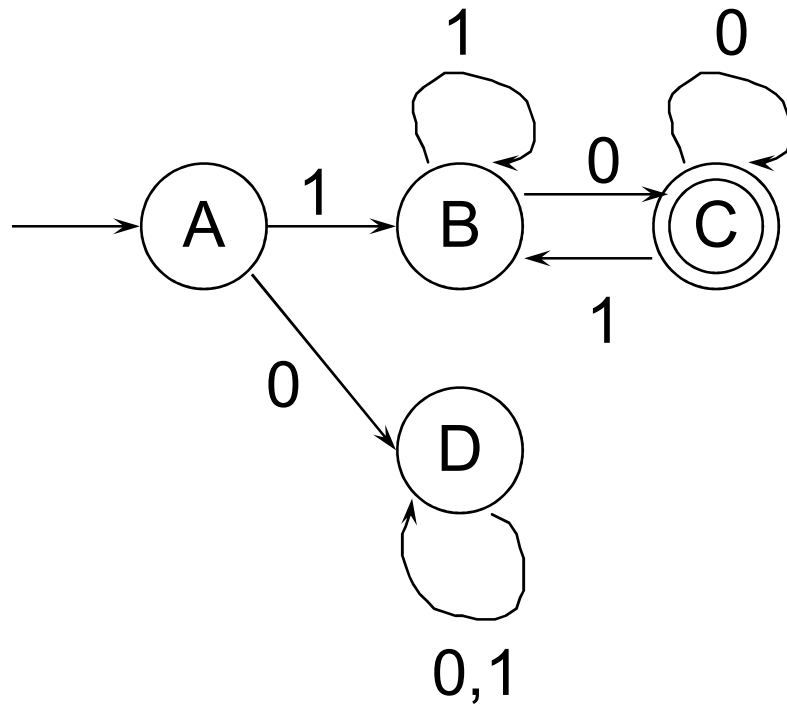
Q2. Which strings will be accepted?

001 0101 1101  
~~00~~ ~~110~~ ~~011~~

DFA 특징 : 종결이 '1' 이어야됨



## Practice #4



state \ input	0	1
A	D	B
B	C	B
<u>C*</u>	C	B
D	D	D

DFA

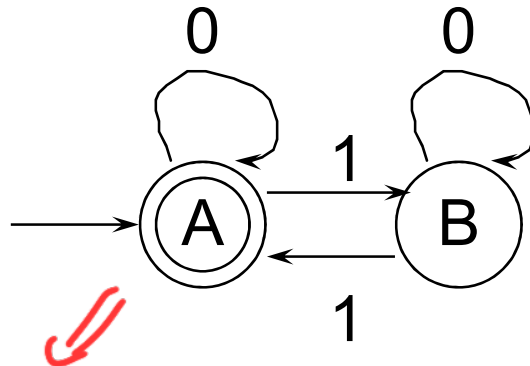
Which strings will be accepted?

1 1 0 0  
 1 0 0 0 0 1 ✗  
 0 0 0 1 ✗



반드시 1로 시작  
 반드시 0으로 끝

## Practice 5: Even Parity Checker



state \ input	0	1
A*	A	B
B	B	A

DFA

$(0^*10^*1)^*$

Which strings will be accepted?

1 0 0 1 *0*

1 0 1 0 0 1 *x*

0 0 1 1 *0*

Q1. Is the DFA can recognize an *empty-string* ?

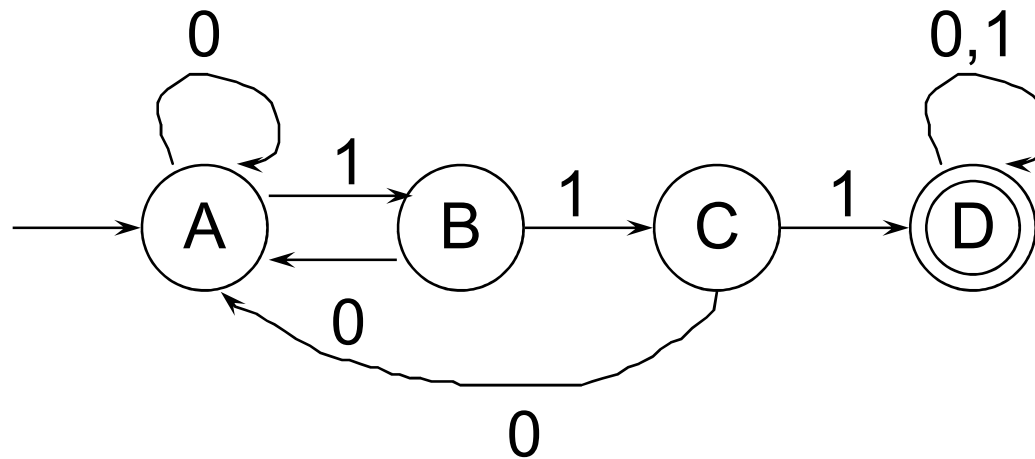
Q2. Draw a DFA of an *odd-parity* checker.

1의 개수가 짝 수개이면 인식



# Quiz

Q: 아래 DFA는 어떤 특징을 갖는 문자열을 인식하는가?

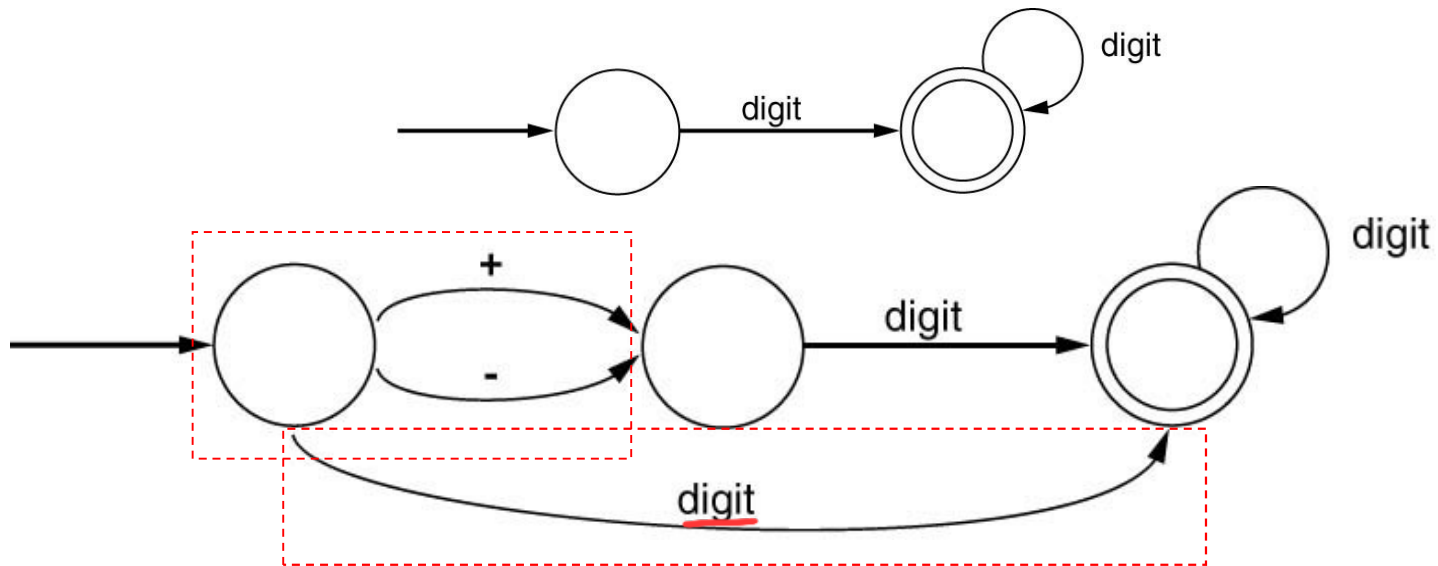


1 이 연속 3 개

(111)!

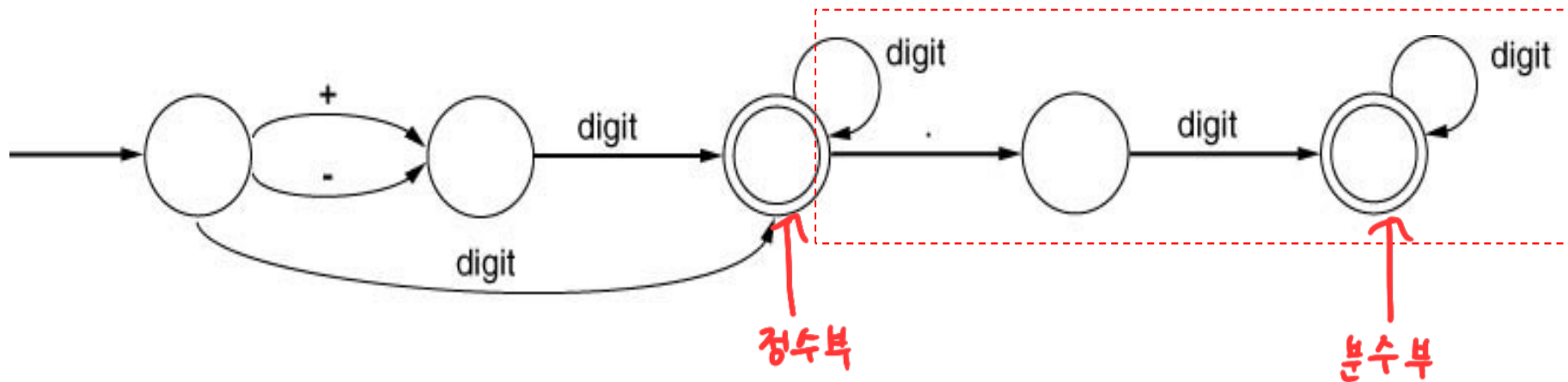
# DFA for recognizing a signedNat

*digit* = [0-9]  
*nat* = *digit*<sup>+</sup>  
*signedNat* = (+|-)? *nat*



# DFA for recognizing a number

*digit* = [0-9]  
*nat* = *digit*<sup>+</sup>  
*signedNat* = (+|-)? *nat*  
*number* = *signedNat* ("." *nat*)?



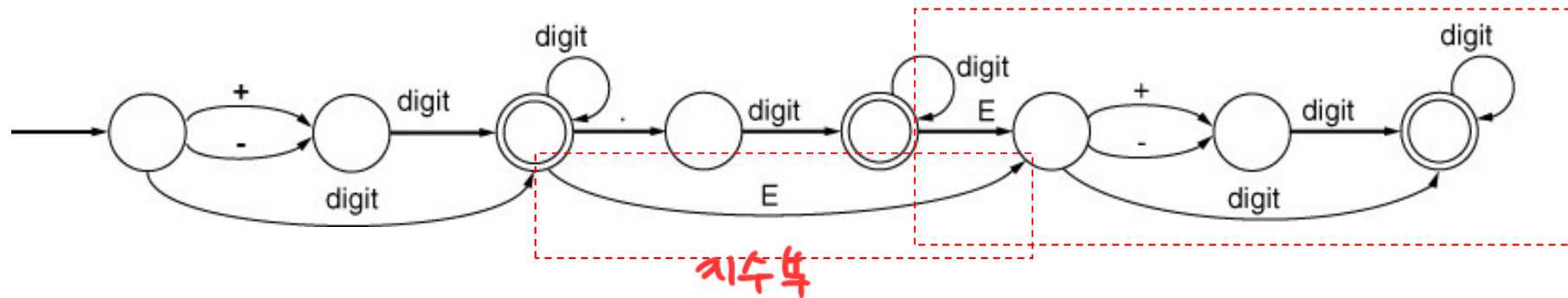
# DFA for a floating-point number

*digit* = [0-9]

*nat* = *digit*<sup>+</sup>

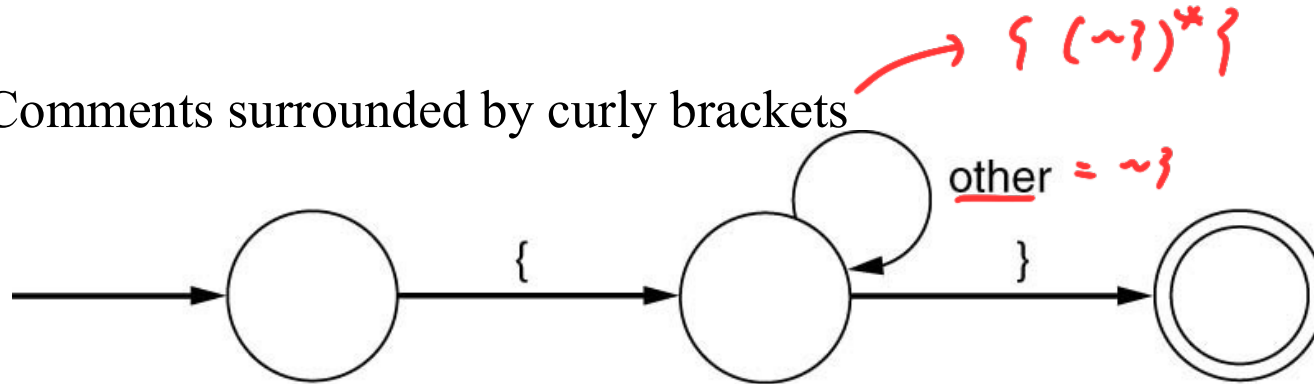
*signedNat* = (+|-)? *nat*

*number* = *signedNat* ("." *nat*)? (*E signedNat*)?

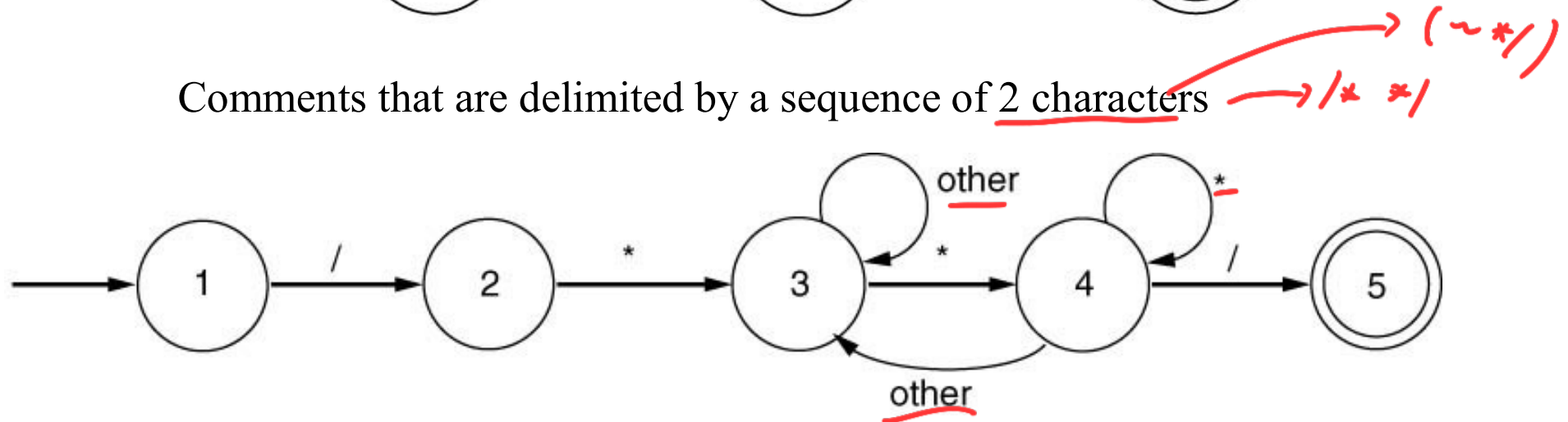


# DFAs for recognizing comments

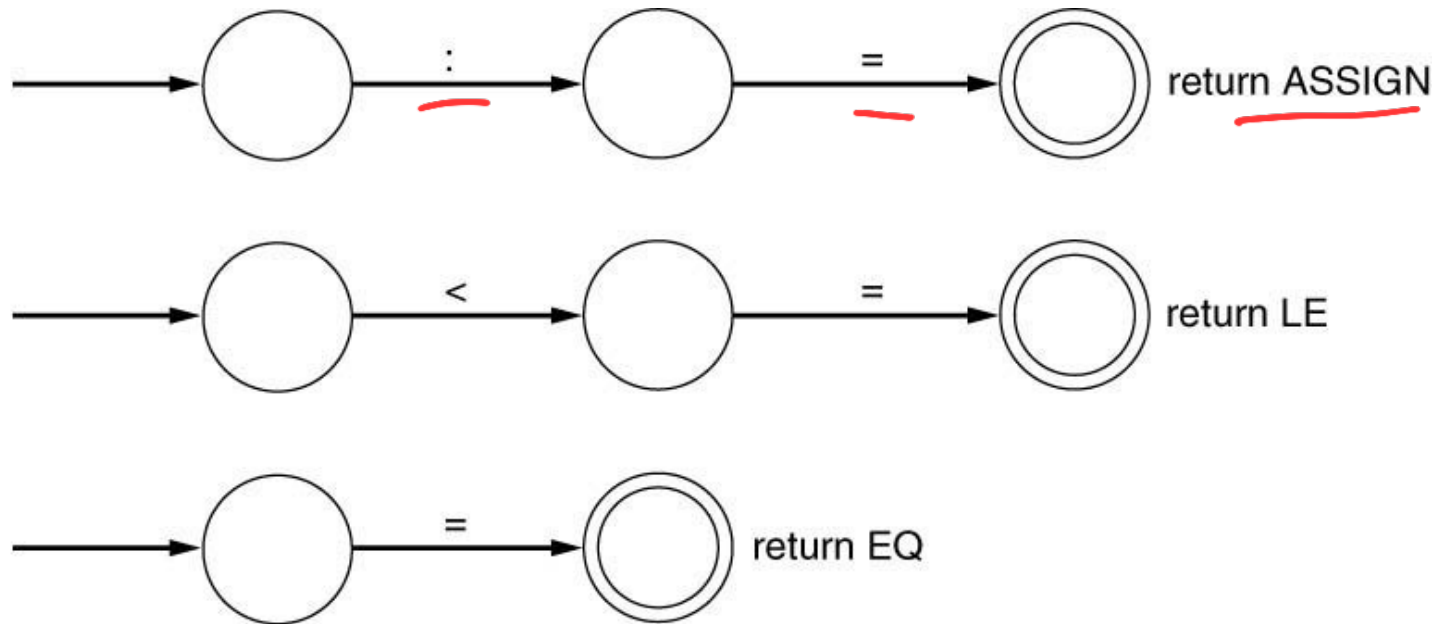
Comments surrounded by curly brackets



Comments that are delimited by a sequence of 2 characters

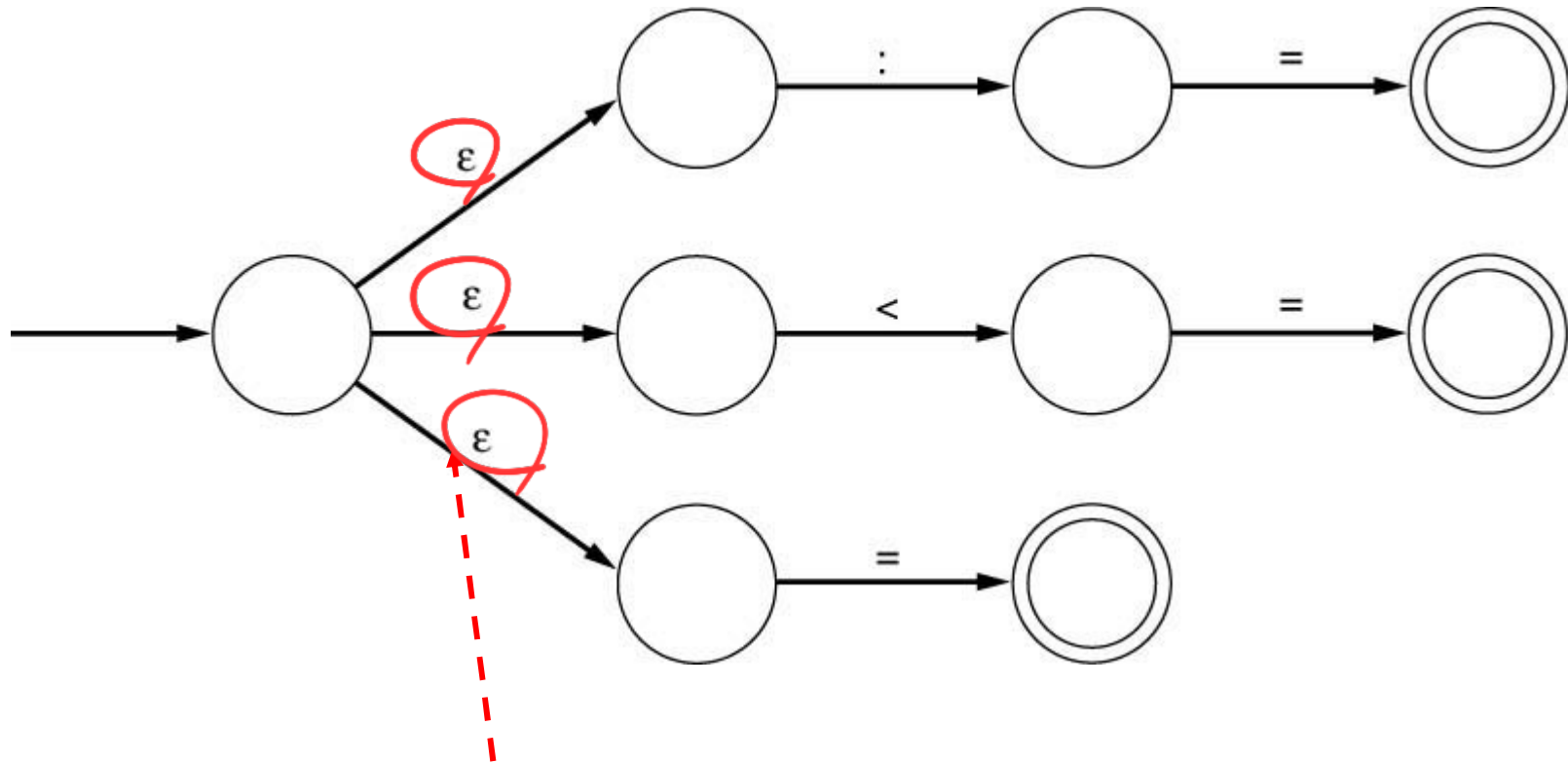


## PL token에 대한 NFA 구성 예 (1/2)





## PL token에 대한 NFA 구성 예 (2/2)



$\epsilon$ -transition may occur “spontaneously”

# 비결정적 유한 오토마타(NFA) (1/2)

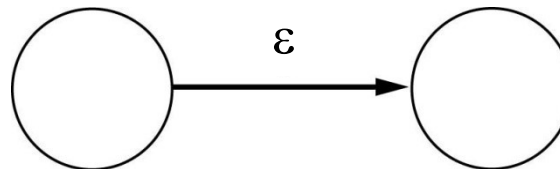
## ■ 찾고자 하는 모든 token에 대한 DFA를 한 개의 DFA로 합쳐야 함.

- DFA의 경우 입력 기호에 대해 다음 상태가 unique하게 결정되어야 하기 때문에 하나로 합치기가 어려움

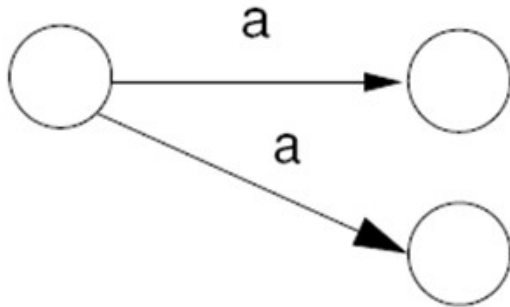
## ■ NFA : DFA 정의를 확장

- More than one transition from a state may exist
- $\epsilon$ -transition
  - A transition may occur without consulting the input string.
  - Without lookahead and without change to the input string

입력에  
무관하게

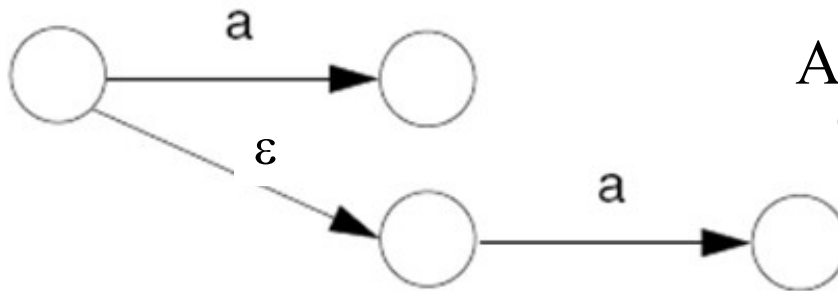


## 비결정적 유한 오토마타(NFA) (2/2)



An NFA with two *a* transitions

---

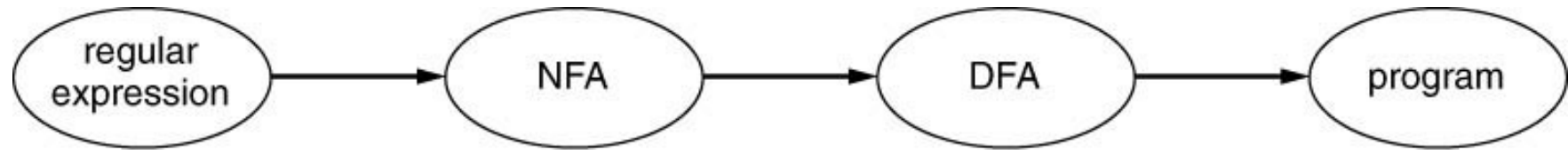


An NFA with a  $\epsilon$ - transitions

---

# Why we need NFA?

---



정규 표현(RE)은 항상 FA로 나타낼 수 있다.

RE는 NFA로 쉽게 변환할 수 있다!

DFA는 coding이 쉽다!

# NFA 정의

■ NFA  $M = ( \Sigma, S, \widetilde{T}, s_0, A )$

▪ 상태 천이 함수  $T : S \times (\Sigma \cup \{\epsilon\}) \rightarrow \underline{P(S)}$

•  $P(S)$  : the power set of  $S$

– 예:  $S = \{1, 2\}$  이면  $P(S) = \{ \{\}, \{1\}, \{2\}, \{1, 2\} \} \rightsquigarrow \{1, 2\}$  로 구성된 모든 상태의 집합

$$T(S_i, a) = S' \quad \text{단, } S' \subseteq S, a \in \Sigma \cup \{\epsilon\}$$

DFA

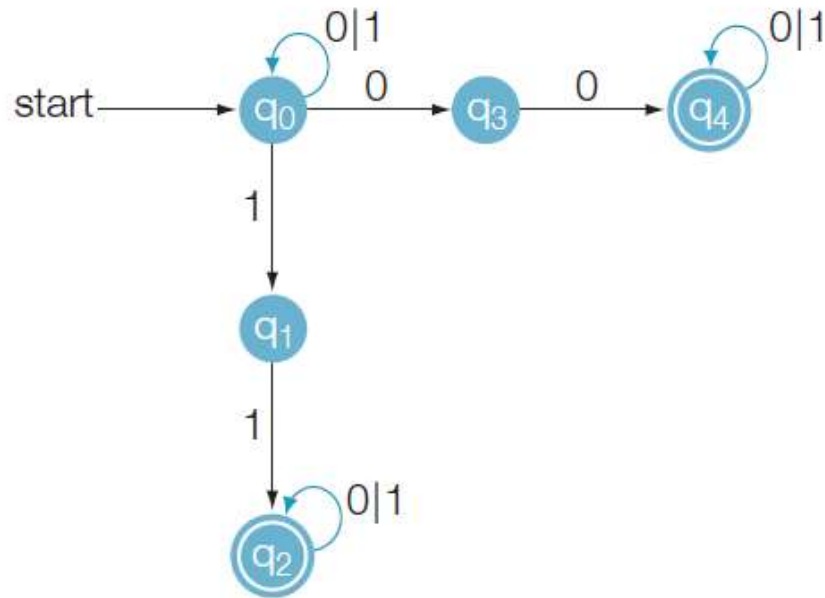
$$\underline{T : S \times \Sigma \rightarrow S}$$

■  $M$  이 인식하는 언어  $L(M)$ 은 아래 조건을 만족하는 문자열  $c_1 c_2 \dots c_n$  의 집합 ( $c_i \in \Sigma \cup \{\epsilon\}$ )

▪  $s_1 = \pi(s_0, c_1), s_2 = \pi(s_1, c_2), \dots, s_n = \pi(s_{n-1}, c_n)$ 인 상태들이 존재

▪  $\nexists, s_n \in A$

# NFA 예



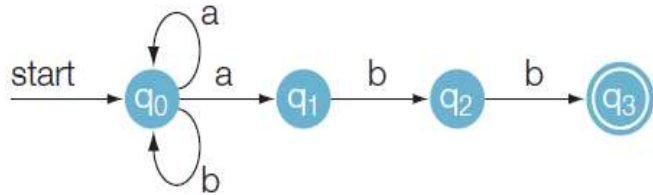
$\delta$	0	1
$q_0$	<u><math>\{q_0, q_3\}</math></u>	<u><math>\{q_0, q_1\}</math></u>
$q_1$	<u><math>\emptyset</math></u>	<u><math>\{q_2\}</math></u>
$q_2$	$\{q_2\}$	$\{q_2\}$
$q_3$	$\{q_4\}$	$\emptyset$
$q_4$	$\{q_4\}$	$\{q_4\}$

IV FA

## NFA $M$ 이 인식하는 언어 $L(M)$

- NFA  $M = (\Sigma, S, T, s_0, A)$  일 때  $M$  이 인식하는 언어  $L(M)$ 은
- 아래 조건을 만족하는 문자열  $c_1 c_2 \dots c_n$  의 집합 ( $c_i \in \Sigma \cup \{\epsilon\}$ )
  - $s_1 = \pi(s_0, c_1), s_2 = \pi(s_1, c_2), \dots, s_n = \pi(s_{n-1}, c_n)$ 인 상태들이 존재
  - 단,  $s_n \in A$

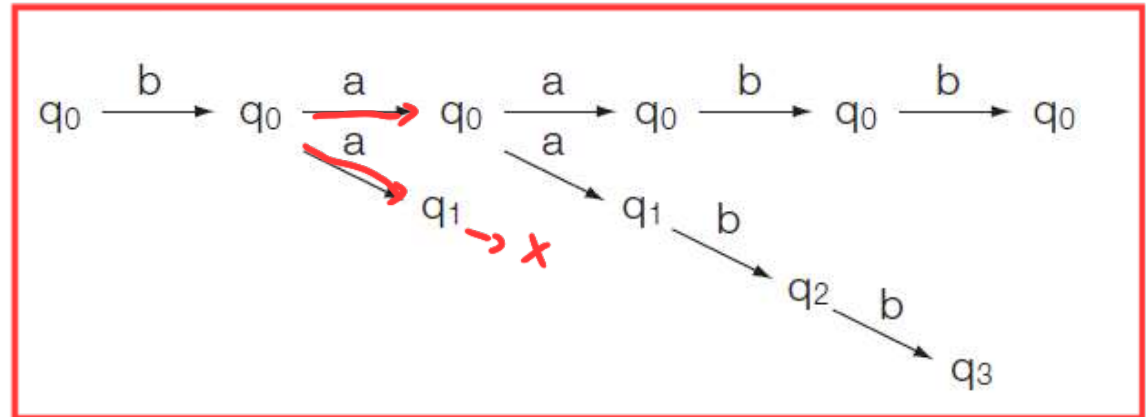
# NFA 문자열 인식 예(1)



$$\begin{aligned}
 \delta(q_0, \underline{baabb}) &= \delta(q_0, \underline{aabb}) \\
 &= \delta(q_0, \underline{abb}) \cup \delta(q_1, \underline{abb}) \\
 &= \{\delta(q_0, \underline{bb}) \cup \delta(q_1, \underline{bb})\} \cup \emptyset \\
 &= \delta(q_0, \underline{b}) \cup \delta(q_2, \underline{b}) \\
 &= \{q_0\} \cup \{q_3\} \\
 &= \{q_0, q_3\} \\
 \underline{\{q_0, q_3\}} \cap A &= \{q_3\}
 \end{aligned}$$

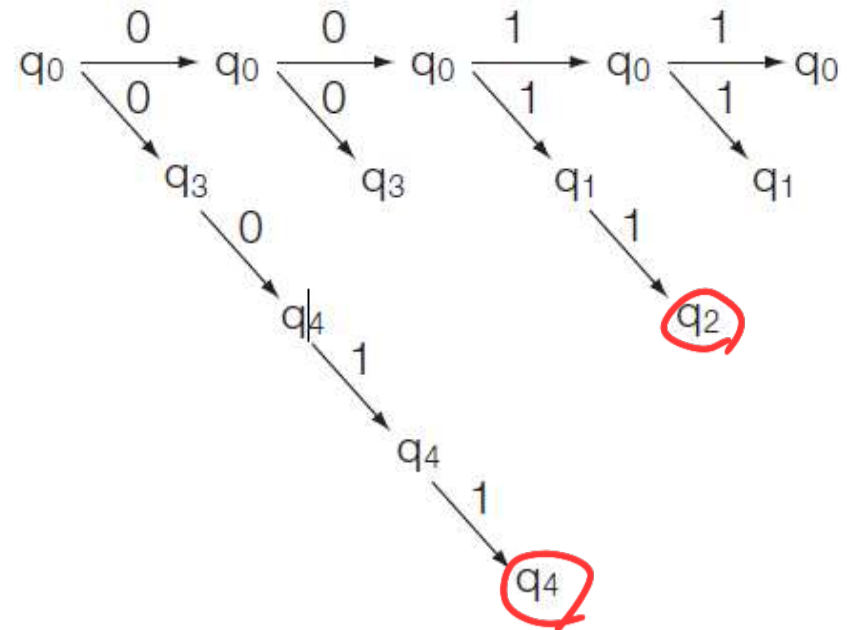
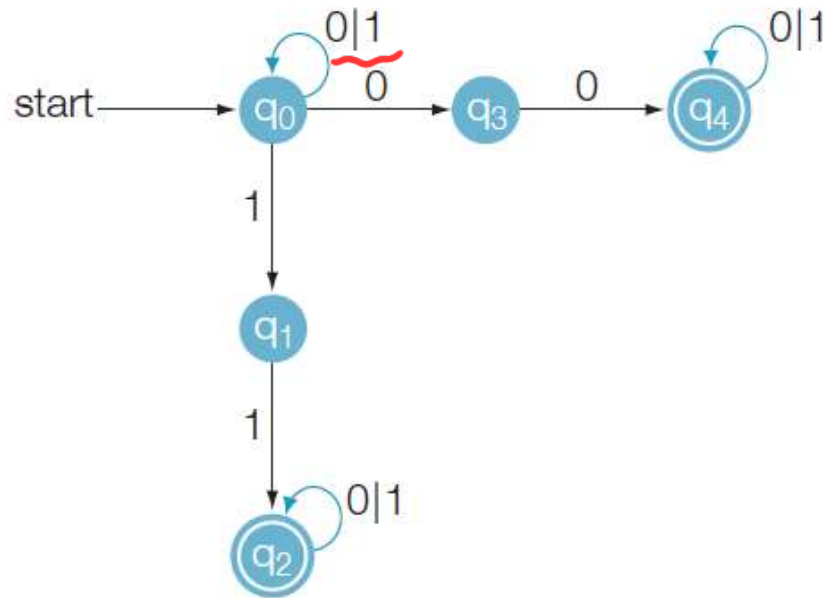
↳ **종결상태**

∴ 위 NFA는 문자열 **baabb**을 인식한다.



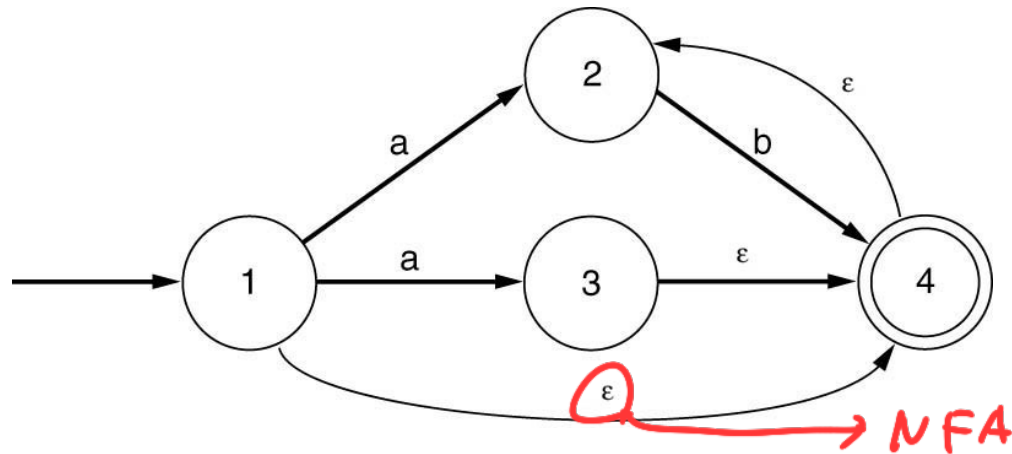


## NFA 문자열 인식 예(2)



DFA  $\rightarrow$  one path

## NFA 는 **abb**를 어떻게 인식할까?



시작상태 1  $\rightarrow$   **$\epsilon$ -transition**  $\rightarrow$  { **1, 2, 4** }

첫 번째 문자 a: {1, 2, 4} 에 속한 각 상태에 대해

1  $\rightarrow$  a  $\rightarrow$  {2, 3}  $\rightarrow$   **$\epsilon$ -transition**  $\rightarrow$  { **2, 3, 4** }

2  $\rightarrow$  a  $\rightarrow$  { }

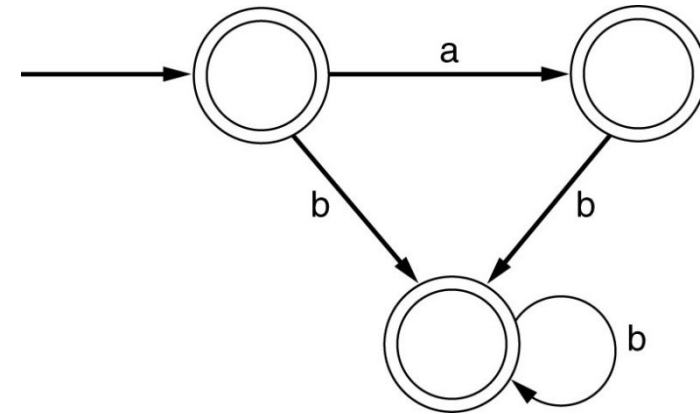
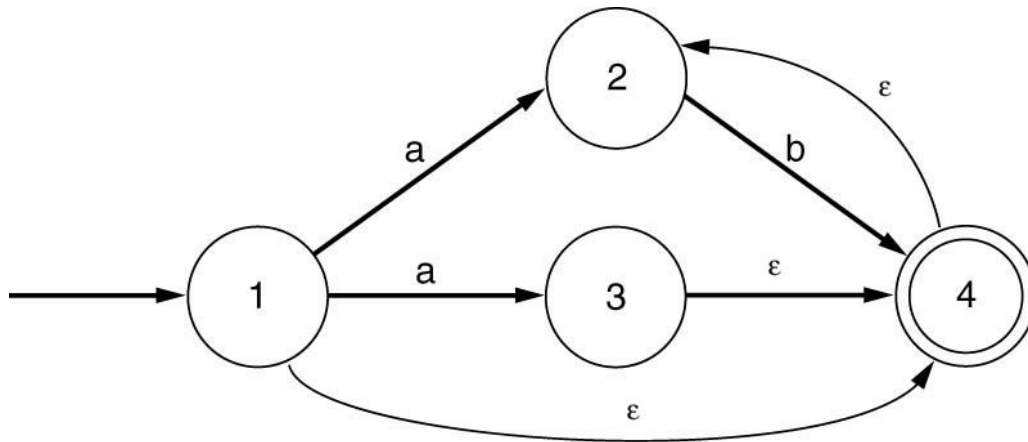
4  $\rightarrow$  a  $\rightarrow$  { }

두 번째 문자 b: {2, 3, 4} 에 속한 각 상태에 대해

.....



## NFA 를 DFA로 바꾸면?



An equivalent DFA

**NFA는 똑같은 기능을 하는 DFA로 항상 변환할 수 있다!**

# About Turing Machine

(supplementary material)

# Do you know Alan Turing?

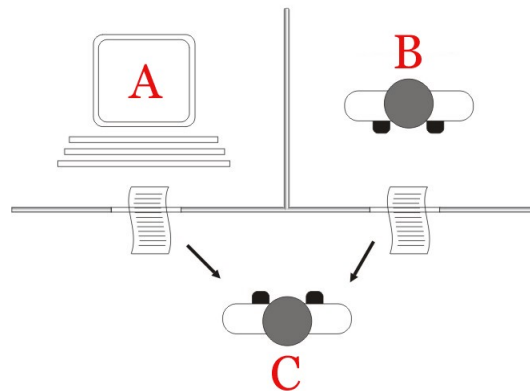
---



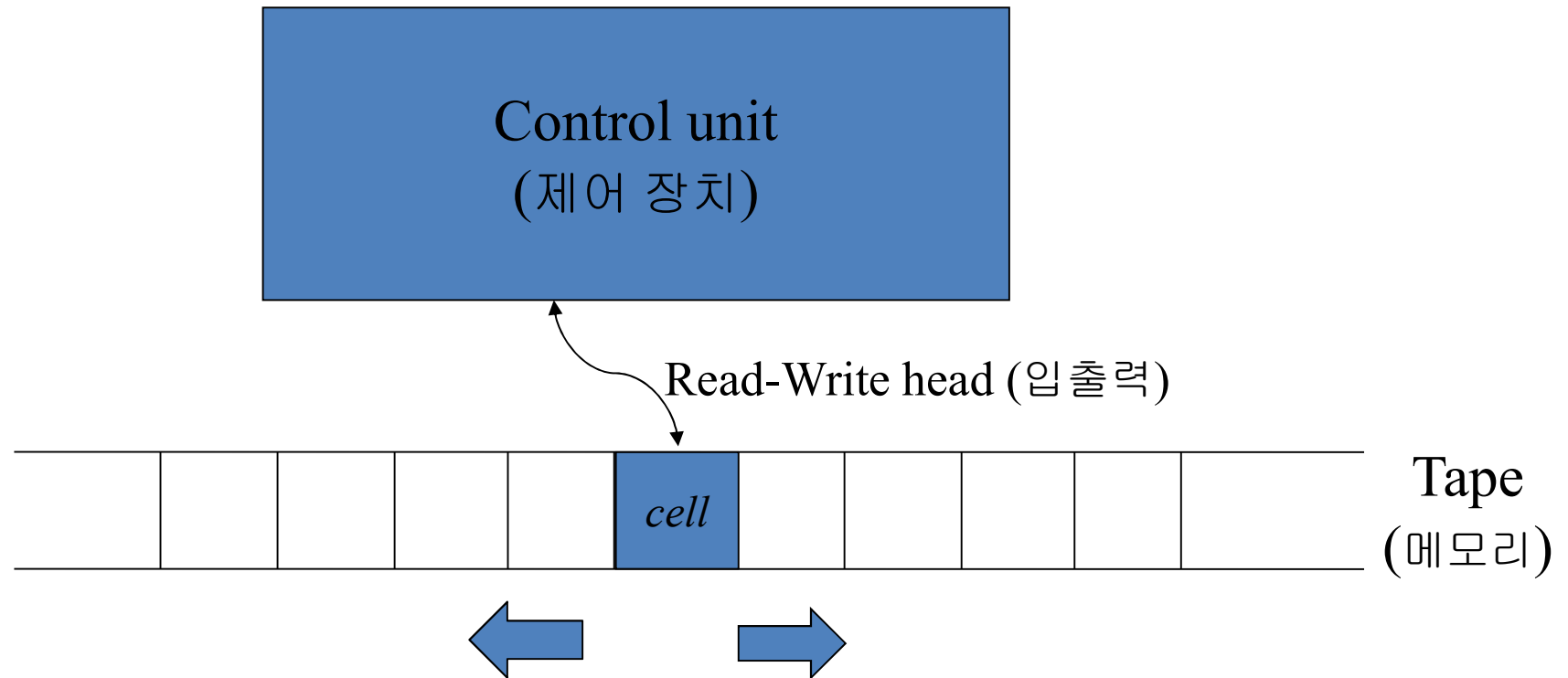
# Do you know Alan Turing?

- Mathematician, logician, cryptanalyst
- *Turing* machine
- Manchester Mark 1
- Enigma machine
- Homosexual
- Apple

- Turing test



# Turing Machine



# Turing Machine 이란?

---

## ■ 테이프(tape) : 저장 공간

### ▪ 1차원 셀 배열

- 셀(cell)
  - 한 개의 symbol을 저장하고 있거나 비어있을 수 있다.
- 배열 크기는 무한대
  - 왼쪽 또는 오른쪽으로 무한대로 확장 가능

### ▪ Tape head

- 왼쪽 또는 오른쪽으로 셀 단위로 이동
- 셀에 저장되어 있는 symbol 을 읽어오거나 다른 symbol로 변경



# Turing Machine

---

## ■ Turing Machine M

$$M = (Q, \Sigma, \Gamma, [\ ], \delta, q_0, F)$$

$\Sigma$  : 입력 알파벳 ( $\Gamma$  에서 공백 기호 제외)

$\Gamma$  : 테이프 알파벳

$[\ ] \in \Gamma$  : 공백 기호(*an empty symbol*)

$\delta$  : 천이 함수  $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

$Q$  : 상태 집합

$q_0 \in Q$  : control unit의 초기 상태(*start state*)

$F \subseteq Q$  : 종료 상태(*final state*) 집합

# Turing Machine

---

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$$

- Control unit의 현재 상태  $Q$
- tape head가 현재 가리키는 셀에 저장된 기호  $\Gamma$
  
- Control unit의 새로운 상태  $Q$
- 셀에 새롭게 저장되는 기호  $\Gamma$
- 이동 방향 : L 또는 R  $\{L,R\}$