제3장 정규 언어

- 정규언어 : token을 기술하는데 사용
- 정규언어의 기술 방법 : 정규문법, 정규표현, finite automata

3.1 정규 문법과 정규 언어

- 정규 문법

  - 우선형 문법 : $A \rightarrow tB,\ A \rightarrow t$
  - 좌선형 문법 : $A \rightarrow Bt,\ A \rightarrow t$

  $\Longrightarrow\ A,B \in V_N,\ t \in V_T^*$

※ t=ε이면 $A \rightarrow B$ (단일 생성 규칙), $A \rightarrow \varepsilon$ (ε-생성규칙)

예1) G_1은 우선형 문법이고, G_2은 좌선형 문법이다.

$G_1 : S \rightarrow 000S \mid 000$

$G_2 : S \rightarrow S11 \mid 11$

※ 아래 문법은 정규 문법이 아님 --> 좌선형과 우선형이 혼합

$G : S \rightarrow aR \mid c$
$R \rightarrow Sb$

$L(G) = \{ a^n cb^n \mid n \geq 0 \}$

[정의 3.1] 정규 문법의 정의

(1) $A \rightarrow aB,\ A \rightarrow a$, 이 때 $a \in V_T,\ A,B \in V_N$이다.

(2) $S \rightarrow \varepsilon$이면, S가 다른 생성규칙의 오른쪽에 나타나지 않아야 한다.

※ 우선형 문법 $A \rightarrow tB,\ t=a_1a_2...a_n,\ a_i \in V_T$을 [정의3.1] 형태로 변환

$$A \rightarrow a_1A_1 \quad A_1 \rightarrow a_2A_2 \quad ..... \quad A_{n-1} \rightarrow a_nB$$

예2) <예1>의 G_1을 정규문법 형태로 변환

$$G_1 : S \rightarrow 000S \Rightarrow S \rightarrow 0S_1,\ S_1 \rightarrow 0S_2,\ S_2 \rightarrow 0S$$
$$S \rightarrow 000 \Rightarrow S \rightarrow 0S_3,\ S_3 \rightarrow 0S_4,\ S_4 \rightarrow 0$$

예3) $L = \{a^nb^m \mid n,m \geq 1\}$ 은 정규 언어이다.

$$G : S \rightarrow aS \mid aA \qquad A \rightarrow bA \mid b$$

3.2 정규표현 (Regular Expression)

[정의 3.2] 정규표현의 정의

1. 정규표현의 기본 요소는 ϕ, ε, 그리고 terminal symbol이다.
   (1) ϕ : 공집합을 나타내는 정규표현
   (2) ε : 집합 {ε}을 나타내는 정규표현
   (3) a($\in V_T$) : 집합 {a}을 나타내는 정규표현
2. e_1, e_2가 각각 정규 언어 L_1, L_2를 표현하는 정규표현이라면,
   (1) $(e_1)+(e_2)$는 $L_1 \cup L_2$를 나타내는 정규표현 (union)
   (2) $(e_1)\cdot(e_2)$는 $L_1 \cdot L_2$를 나타내는 정규표현 (concatenation)
   (3) $(e_1)^*$는 $L_1^* = \{\varepsilon\} \cup L_1^1 \cup L_1^2 \cup \cdots \cup L_1^n \cup \cdots$에 대한 정규표현 (closure)
3. 1, 2 이외의 어떤 것도 정규표현이 될 수 없다.
   단, $(e_1)\cdot(e_2) = (e_1)(e_2)$, $(e_1)+(e_2) = (e_1) \mid (e_2)$, $e_1^+ = e_1 \cdot e_1^*$

※ 정규표현의 우선 순위 : 괄호 > * > · > +

예4) 정규표현의 예

(1) 정규표현 ab^* ⇒ 언어 { ab^n / $n \geq 0$ }
(2) 정규표현 $(0+1)^*$ ⇒ 언어 $\{0, 1\}^*$
(3) 정규표현 $(a+b)^*abb$ ⇒ a,b 반복 후에 abb로 끝나는 string 집합

- identifier의 정규표현 : letter(letter+digit)*

[정의 3.3] 정규표현의 등가 ⇒ 두 정규표현이 같은 언어의 집합을 표현할 때

즉, 정규표현 α,β가 각각 언어 L_α와 L_β를 표현한다고 할 때
$L_\alpha = L_\beta$이면 α = β이다.

예6) 정규표현 $a(ba)^*$가 나타내는 스트링의 형태는 $ababa...ba$ 이고
$(ab)^*a$도 같은 스트링 형태($abab...a$)를 표현하므로 $a(ba)^* = (ab)^*a$ 이다.

※ 정규표현은 아래와 같은 수학적인 성질을 만족한다.

(1) $\alpha + \beta = \beta + \alpha$ (2) $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$
(3) $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ (4) $\alpha(\beta+\gamma) = \alpha\beta+\alpha\gamma$
(5) $(\beta+\gamma)\alpha = \beta\alpha+\gamma\alpha$ (6) $\alpha + \alpha = \alpha$
(7) $\alpha + \phi = \alpha$ (8) $\alpha\phi = \phi = \phi\alpha$
(9) $\varepsilon\alpha = \alpha = \alpha\varepsilon$ (10) $\alpha^* = \varepsilon + \alpha\alpha^*$
(11) $\alpha^* = (\varepsilon + \alpha)^*$ (12) $(\alpha^*)^* = \alpha^*$
(13) $\alpha^* + \alpha = \alpha^*$ (14) $\alpha^* + \alpha^+ = \alpha^*$
(15) $(\alpha + \beta)^* = (\alpha^*\beta^*)^*$

[정의 3.5] 정규식의 정의 : 계수(coefficient)가 정규표현인 식

예7) $\alpha_i, \beta_i (i=1,2,3)$이 정규표현일 때 다음은 정규식이다.

$X=\alpha_1 X+\alpha_2 Y+\alpha_3, \quad Y=\beta_1 X+\beta_2 Y+\beta_3$

※ $X=\alpha X+\beta$**의 유일한 해는** $X=\alpha^*\beta$**이다.** $\Longrightarrow$ $X=\alpha^*\beta$를 준식에 대입.

예8) 아래 정규문법에 대한 정규표현을 구하시오.

$S \rightarrow aS$
$S \rightarrow bR$
$S \rightarrow \varepsilon$
$R \rightarrow aS$

$S = aS + bR + \varepsilon$ --- *(1)*
$R = aS$ --- *(2)*

$S = aS + b(aS) + \varepsilon$: 식(2)를 (1)에 대입
$= aS + baS + \varepsilon$
$= (a+ba)S + \varepsilon$: $X=\alpha X+\beta$ 의 유일한 해는 $X=\alpha^*\beta$

$S = (a+ba)^*\varepsilon = (a+ba)^*$

$L(G) = (a+ba)^*$

예9) 아래 정규문법에 대한 정규표현을 구하시오.

$$S \rightarrow aA \mid bB \mid b$$
$$A \rightarrow bA \mid \varepsilon$$
$$B \rightarrow bS$$

$S = aA + bB + b$ --- (1)
$A = bA + \varepsilon$ --- (2) ==> $A = b^*$
$B = bS$ --- (3)

$S = a(b^*) + b(bS) + b$
$= bbS + (ab^* + b)$

$L(G) = (bb)^*(ab^*+b)$

예10) 다음 정규식이 나타내는 정규표현은?

$X = \alpha_1 X + \alpha_2 Y + \alpha_3$, $Y = \beta_1 X + \beta_2 Y + \beta_3$

3.3 유한 오토마타 (Finite Automata: FA)

- 언어 인식기(language recognizer)
  입력 string이 그 언어의 문장이면 yes, 아니면 no
- 유한 자동(finite automata) : 언어 인식기 중에서 가장 간단한 형태

[정의 3.6] 유한 오토마타 M의 정의

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

Q : 상태(state)들의 유한 집합
Σ : input symbol들의 유한 집합
δ : 사상 함수(mapping function)
$Q \times \Sigma \rightarrow 2^Q$(power set of Q), $\delta(q,a) = \{p_1, p_2, ..., p_n\}$
※ ***DFA*** *vs.* ***NFA***
q_0 : start or initial state ($q_0 \in Q$)
F : final state들의 집합 ($F \subseteq Q$)

3.3.1 DFA *(Deterministic Finite Automata)*

[정의 3.7] DFA 정의 ⇒ 정의 3.6에서 δ가 전이함수*(transition function)*
$\delta : Q \times \Sigma \rightarrow Q$, $\delta(q,a) = p$

예11) 유한 오토마타 $M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$
$\delta(q_0, a) = q_1$ $\delta(q_0, b) = q_2$
$\delta(q_1, a) = q_2$ $\delta(q_1, b) = q_0$
$\delta(q_2, a) = q_0$ $\delta(q_2, b) = q_1$

※ 전이함수를 행렬로 표시한 것을 상태전이표(state-transition table)라 함.

예12) <예11>의 전이함수를 상태전이표로 구성

δ	a	b
q_0	q_1	q_2
q_1	q_2	q_0
q_2	q_0	q_1

※ 전이함수의 확장 (terminal symbol에서 terminal string으로)

$Q \times \Sigma \rightarrow Q \quad \Rightarrow \quad Q \times \Sigma^* \rightarrow Q$

$\delta(q, \varepsilon) = q, \quad \delta(q, xa) = \delta(\delta(q, x), a), \quad$ 단, $a \in \Sigma$이고 $x \in \Sigma^*$

예13) 상태 q_0에서 input string aba가 나타났을 때

$\delta(q_0, aba) = \delta(\delta(q_0, ab), a) = \delta(\delta(\delta(q_0, a), b), a)$

※ <예11>의 상태전이표 적용

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

※ 유한 오토마타에 의한 **accept** or **reject**

input string x에 대해 $\delta(q_0, x) = p$이고 $p \in F$이면 accept, 아니면 reject

[정의 3.8] DFA M에 의하여 인식되는 언어 : $L(M)$

$$L(M) = \{x \mid \delta(q_0, x) \in F\}$$

예14) $M = (\{p,q,r\}, \{0,1\}, \delta, p, \{r\})$는 string 1001, 0110을 인식하는가?

δ	0	1
p	q	p
q	r	p
r	r	r

(1) $\delta(p, 1001) = \delta(p, 001) = \delta(q, 01) = \delta(r, 1) = r \in F$이므로 accept

(2) $\delta(p, 0110) = \delta(q, 110) = \delta(p, 10) = \delta(p, 0) = q \notin F$이므로 reject

[정의 3.9] 상태전이도 (state transition diagram)

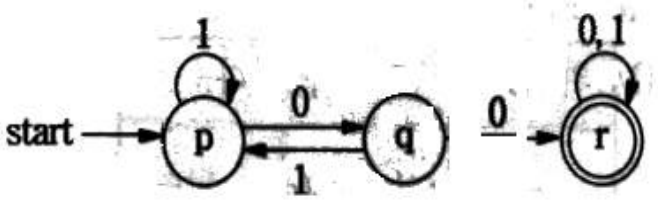
node : automata의 각 state를 표시함.

arc : $\delta(q, a)=p$ 에 대하여 state q에서 p로 가고
label이 a인 directed arc

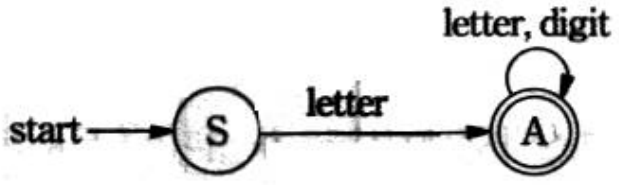
final state : double circle로 표시

start state : start 지시선으로 표시

예15) 예14에 대한 유한 자동을 상태전이도로 표현



예16) 명칭(identifier)에 대한 상태전이도



예17) 부동소수점(floating-point number)에 대한 상태전이도

※ DFA의 특징

1. no ε-transition

2. 한 상태에서 input symbol에 대해 0 or 1개의 next state

[정의 3.10] 유한 자동 $M = (Q, \Sigma, \delta, q_0, F)$이 모든 $q \in Q$, $a \in \Sigma$에 대하여 $\delta(q, a)$가 오직 1개의 next state를 가질 때, M은 completely specified.

```
Algorithm recognize;
begin
        currentState := q0;
        get(nextSymbol);
        while input string not exhausted do
                begin currentState := δ(currentState, nextSymbol);
                        get(nextSymbol);
                end;
        if currentState in F then write('input recognized')
        else write('input not recognized')
end.
```

3.3.2 NFA (Nondeterministic Finite Automata)

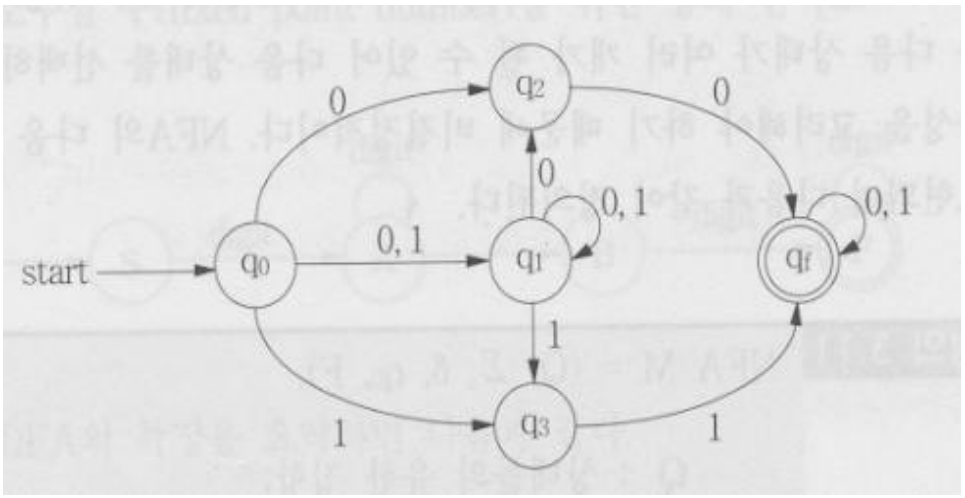
[정의 3.11] NFA 정의 ⟹ "[정의 3.6] FA 정의"와 동일

NFA $M = (Q, \Sigma, \delta, q_0, F)$

Q : 상태(state)들의 유한 집합
Σ : input symbol들의 유한 집합
δ : 사상 함수(mapping function)
$Q \times \Sigma \rightarrow 2^Q$(power set of Q), $\delta(q,a) = \{p_1, p_2, ..., p_n\}$
q_0 : start or initial state ($q_0 \in Q$)
F : final state들의 집합 ($F \subseteq Q$)

예18) NFA $M = (\{q_0, q_1, q_2, q_3, q_f\}, \{0, 1\}, \delta, q_0, \{q_f\})$

δ	0	1
q_0	$\{q_1, q_2\}$	$\{q_1, q_3\}$
q_1	$\{q_1, q_2\}$	$\{q_1, q_3\}$
q_2	$\{ q_f \}$	ϕ
q_3	ϕ	$\{ q_f \}$
q_f	$\{ q_f \}$	$\{ q_f \}$



※ 전이함수의 확장

[단계 1] 하나의 symbol을 string으로 확장

$$Q\times\Sigma \rightarrow 2^{Q} \quad \Rightarrow \quad Q\times\Sigma^{*} \rightarrow 2^{Q}$$
$$\delta(q, \varepsilon) = \{q\}, \quad \delta(q, xa) = \cup_{p\in\delta(p, a)} \delta(q,a)$$

예19) 예18에서 주어진 NFA에 따라 q_0에서 스트링 1001을 다 본 상태

$$\begin{aligned}\delta(q_0, 1001) &= \delta(q_1, 001) \cup \delta(q_3, 001)\\ &= \delta(q_1, 01) \cup \delta(q_2, 01) \cup \phi\\ &= \delta(q_1, 1) \cup \delta(q_2, 1) \cup \delta(q_f, 1)\\ &= \{q_1, q_3, q_f\}\end{aligned}$$

[단계 2] 하나의 state를 여러 state로 확장

$$Q\times\Sigma \rightarrow 2^{Q} \quad \Rightarrow \quad 2^{Q}\times\Sigma^{*} \rightarrow 2^{Q}$$
$$\delta(\{p_1,p_2,\ldots,p_k\},x) = \cup_{i=1}^{k} \delta(p_i,a)$$

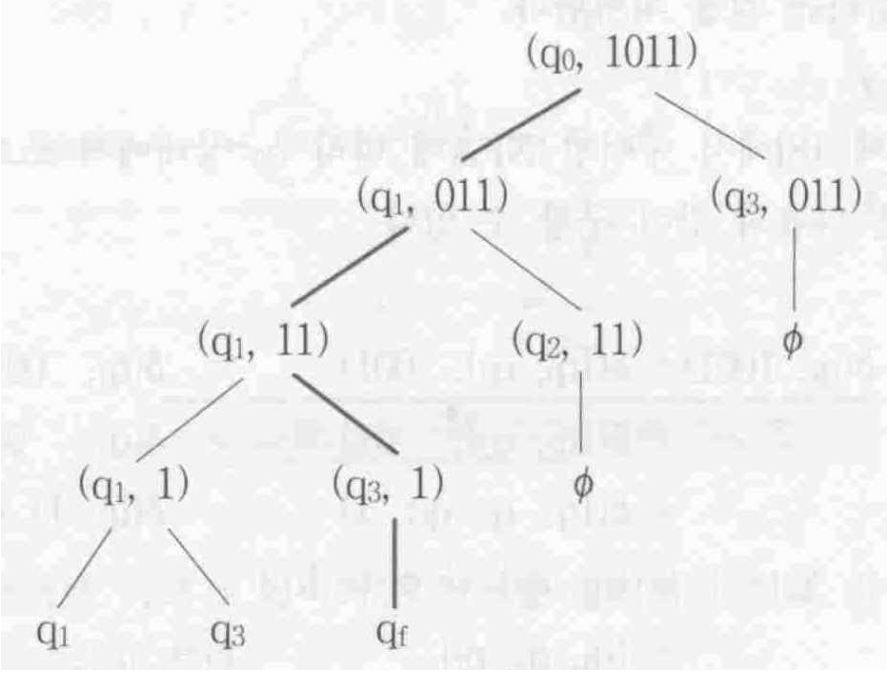
예20) $\delta(q_0, 1001) = \delta(\{q_1, q_3\}, 001) = \delta(\{q_1, q_2\}, 01) = \delta(\{q_1, q_2, q_f\}, 1)$
$= \{q_1, q_3, q_f\}$

※ final state가 하나라도 있으면 accept

예21) 예18의 NFA M은 string 1011을 인식하는가? yes

$$\delta(\{q_0\}, 1011) = \delta(\{q_1, q_3\}, 011) = \delta(\{q_1, q_2\}, 11)$$
$$= \delta(\{q_1, q_3\}, 1) = \{q_1, q_3, q_f\}$$

※ 위 과정을 tree형태로 나타내면 아래 그림과 같다.
⇒ terminal node에 final state가 하나라도 있으면 accept



※ state수가 m이고 input string의 길이가 n이면,
tree의 node수는 m^n ⇒ exponential time
(computationally intractable)

3.3.3 NFA를 DFA로 변환

모든 NFA는 같은 언어를 인식하는 DFA로 변환할 수 있다.

[정리 3.2] NFA $M=(Q, \Sigma, \delta, q_0, F)$을 DFA $M'=(Q', \Sigma, \delta', q_0', F')$으로 변환

(1) $Q' = 2^Q$ (Q의 power set)
Q'의 한 상태는 $[q_1, q_2, ..., q_i]$의 형태로 표시한다.
(2) $q_0' = [q_0]$
(3) $F' = \{q \in Q' \mid q$는 F의 상태들 중에 적어도 하나를 포함$\}$
(4) $\delta(\{q_1, q_2, ..., q_i\}, a) = \{p_1, p_2, ..., p_j\}$이면,
$\delta'([q_1, q_2, ..., q_i], a) = [p_1, p_2, ..., p_j]$이다.

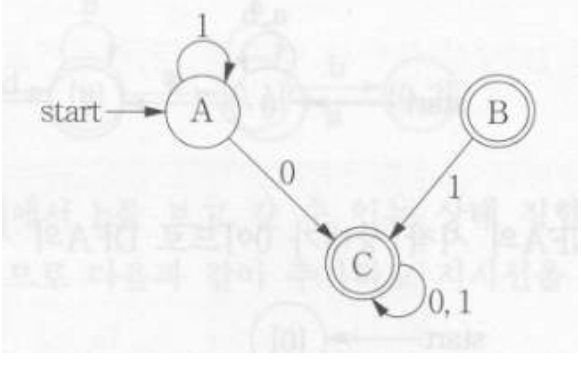
예22) NFA $M=(\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$을 DFA로 변환하시오.

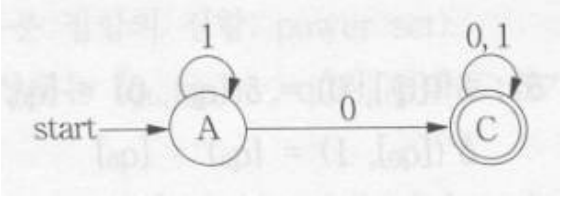
δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	Φ	$\{q_0, q_1\}$

(1) 상태집합 $Q' = \{[q_0], [q_1], [q_0, q_1]\}$
(2) start state $q_0' = [q_0]$
(3) final states $F' = \{[q_1], [q_0, q_1]\}$
(4) 전이함수 δ'

δ'	0	1
$[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_1]$	Φ	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

※ inaccessible state 발생 ⇒ 상태 B 제거 가능
$[q_0] = A, [q_1] = B, [q_0, q_1] = C$로 했을 때 상태전이도

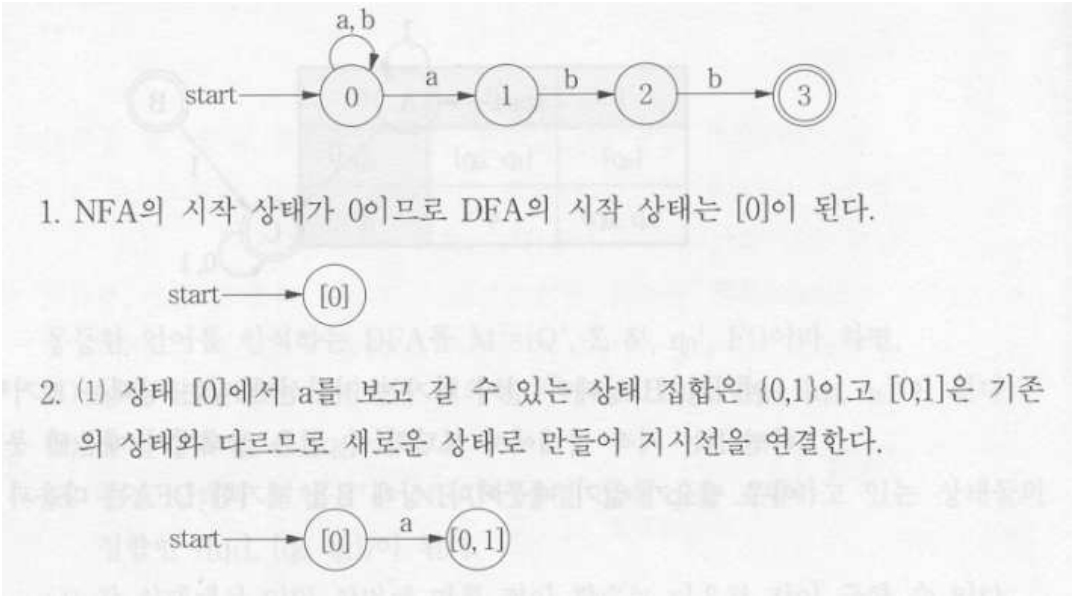




※ accessible states만 생성하는 DFA로 변환 방법

1. NFA의 start state q_0을 DFA의 start state $[q_0]$로 놓는다.
   초기에 $Q' = \{[q_0]\}$이 된다.
2. Q'의 각 상태에서 input symbol을 보고 갈 수 있는 next state를 구한다.
3. next state가 Q'에 없으면 Q'에 추가하고 input string에 directed arc 연결
   next state가 Q'에 있으면 input string에 대한 directed arc만 연결
4. 과정 2,3을 반복하여 new state가 추가되지 않을 때까지 계속한다.
   new state가 NFA의 final state를 포함하고 있으면 DFA의 final state

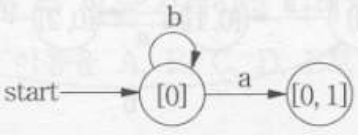
예23) 다음 NFA로부터 동등한 언어를 인식하는 DFA를 구하시오.



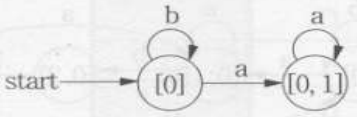
1. NFA의 시작 상태가 0이므로 DFA의 시작 상태는 [0]이 된다.

2. (1) 상태 [0]에서 a를 보고 갈 수 있는 상태 집합은 {0,1}이고 [0,1]은 기존의 상태와 다르므로 새로운 상태로 만들어 지시선을 연결한다.

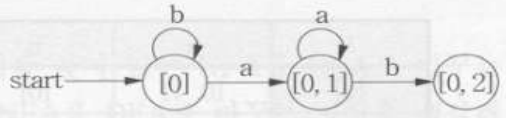
(2) 상태 [0]에서 b를 보고 갈 수 있는 상태 집합은 {0}이고, [0]은 기존의 상태와 같으므로 지시선만 그린다.



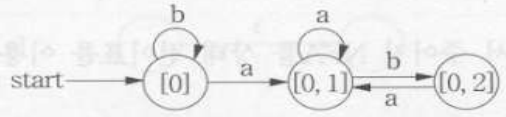
3. (1) 상태 [0,1]에서 a를 보고 갈 수 있는 상태 집합은 {0, 1}이고, [0, 1]은 기존에 존재하는 상태이므로 지시선만 그린다.



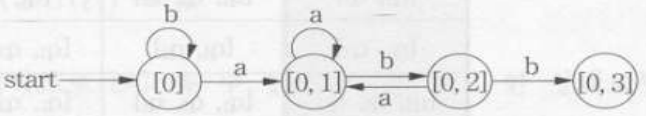
(2) 상태 [0, 1]에서 b를 보고 갈 수 있는 상태 집합은 {0, 2}이고, [0, 2]는 새로운 상태이므로 추가하고 지시선을 연결한다.



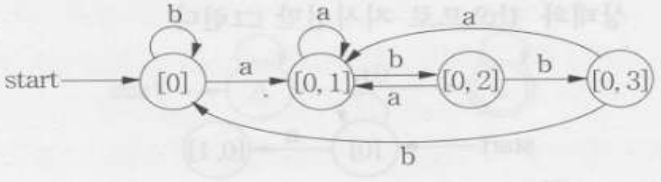
4. (1) 상태 [0, 2]에서 a를 보고 갈 수 있는 상태 집합은 {0, 1}이고, [0, 1]은 기존에 존재하는 상태이므로 지시선만 그린다.



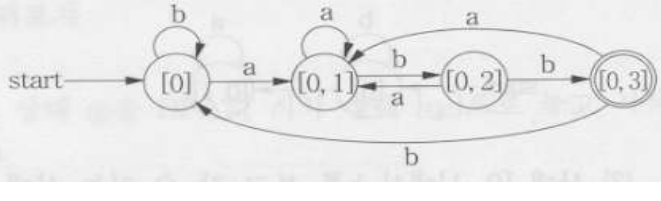
(2) 상태 [0, 2]에서 b를 보고 갈 수 있는 상태 집합은 {0, 3}이고 [0, 3]은 새로운 상태이므로 다음과 같이 추가하고 지시선을 연결한다.



5. 상태 [0, 3]에서 a를 보고 갈 수 있는 상태 집합은 {0,1}이고, b를 보고 갈 수 있는 상태는 {0}이므로 새로 만들지 않고 지시선만 만든다.



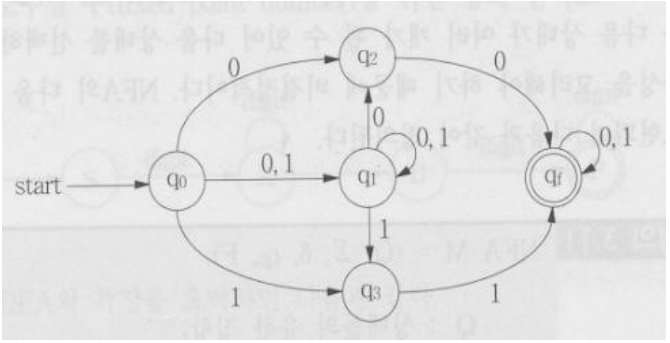
6. 더 이상 새로운 상태가 추가되지 않기 때문에 종료된다. 그리고 NFA의 종결 상태 3을 포함하는 상태 [0, 3]은 DFA의 종결 상태로 표시한다.



이상과 같은 일련의 과정을 상태 전이표를 이용하여 구현하면 다음과 같다.

δ	a	b
[0]	[0, 1]	[0]
[0, 1]	[0, 1]	[0, 2]
[0, 2]	[0, 1]	[0, 3]
[0, 3]	[0, 1]	[0]

예24) 예18의 NFA를 상태전이표를 이용하여 DFA로 변환



아래 오른쪽 상태전이표는 각 상태 이름을 A, B, C, D, E로 바꾼 것임.

δ	0	1
[q0]	[q1, q2]	[q1, q3]
[q1, q2]	[q1, q2, qf]	[q1, q3]
[q1, q3]	[q1, q2]	[q1, q3, qf]
[q1, q2, qf]	[q1, q2, qf]	[q1, q3, qf]
[q1, q3, qf]	[q1, q2, qf]	[q1, q3, qf]

δ	0	1
A	B	C
B	D	C
C	B	E
D	D	E
E	D	E

3.3.4 ε-NFA를 DFA로 변환

ε-NFA : ε-transition을 갖는 NFA

[정의 3.12] ε-NFA $M=(Q, \Sigma, \delta, q_0, F)$

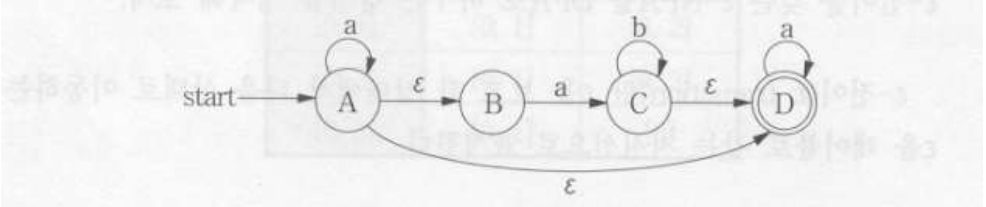
ε-transition을 제외한 모든 정의는 NFA와 같다.

$\delta : Q\times(\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$

[정의 3.13] ε-CLOSURE의 정의

1. 임의의 state s에 대해서 ε-CLOSURE(s)는 s와 s로부터 ε-transition에 의하여 도달할 수 있는 모든 state들의 집합이다.
   $\varepsilon\text{-CLOSURE}(s) = \{s\} \cup \{ q \mid \delta(p,\varepsilon)=q,\ p \in \varepsilon - CLOSURE(s)\}$
2. T가 하나 이상의 state인 경우에 ε-CLOSURE(T)는 집합 T의 각 state에 대해 1과 같은 방법으로 CLOSURE 집합을 구하여 합집합을 구한다.
   $\varepsilon\text{-CLOSURE}(T) = \cup_{q\in T}\ \varepsilon - CLOSURE(q)$

예25) 다음 ε-NFA에 대하여 CLOSURE를 구하시오.



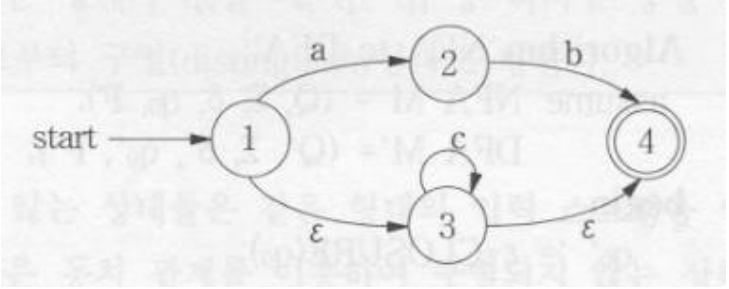
(1) ε-CLOSURE(A) = {A, B, D}

(2) ε-CLOSURE({A, C}) = ε-CLOSURE(A) ∪ ε-CLOSURE(C)
= {A, B, C, D}

※ ε-NFA를 DFA로 변환하는 방법 (ε-CLOSURE 이용)

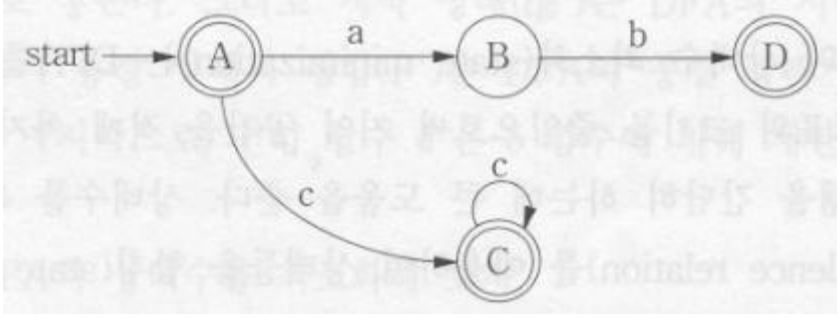
1. start state에 대한 ε-CLOSURE를 구하고 이것을 DFA의 start state로 함
2. ε-CLOSURE를 이용하여 next state를 구한다.
3. 추가된 state들에 대해 새로운 state가 생성되지 않을 때까지 2.를 반복한다.

예26) 다음 ε-NFA를 상태전이표를 이용하여 DFA로 변환하시오. (p.97)



δ	a	b	c
ε-CLOSURE(1)={1, 3, 4} ≡[1, 3, 4]	ε-CLOSURE(2)={2} ≡[2]	ϕ	ε-CLOSURE(3)={3, 4} ≡[3, 4]
[2]	ϕ	ε-CLOSURE(4)={4} ≡[4]	ϕ
[3, 4]	ϕ	ϕ	ε-CLOSURE(3)={3,4} ≡[3,4]
[4]	ϕ	ϕ	ϕ

A = [1,3,4], B = [2], C = [3,4], D = [4] 로 치환한 오토마타



※ ε-NFA를 DFA로 변환하는 알고리즘 : *LEX 알고리즘의 핵심적인 부분*

LEX : 정규 문법을 DFA로 구성하는 Tool --> 어휘분석기 자동 생성

```
Algorithm NFA_to_DFA;
 assume NFA M = (Q, Σ, δ, q0, F);
        DFA M'= (Q', Σ, δ', q0', F');
begin
   q0' := ε-CLOSURE(q0);
   Q' := {q0'};  δ' := { };
   repeat
      for each q ∈ Q' do
        for each a ∈ Σ do
          q' := ε-CLOSURE(δ(q, a));
          if q' ∉ Q' then Q' := Q' ∪ {q'} fi
          δ' := δ' ∪ {δ'(q, a) = q'};
        end for
      end for
   until Q' does not change
   F' := { };
   for each q ∈ Q' do
     if q ∩ F ≠ Φ then F' := F' ∪ {q}; fi
end NFA_to_DFA.
```

3.3.5 DFA의 상태수 최소화 (state minimization)

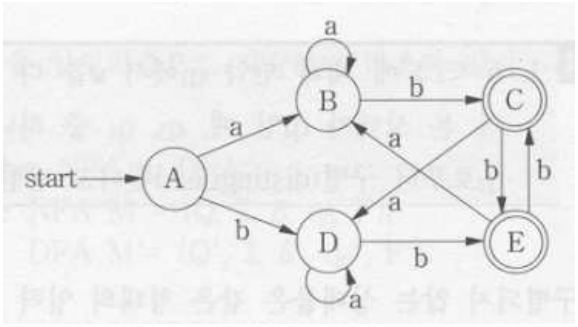
방법: 동치관계(equivalence relation)를 이용하여 상태합침(state merge)

[정의 3.14] 동치관계(equivalence relation) 정의
$\omega \in \Sigma^*$에 대해 q_1에서 ω를 다 본 상태가 q_3이고 q_2에서 ω를 다 본 상태가 q_4일 때, q_3, q_4 중에서 하나만 final state에 속하면 q_1은 q_2로부터 구별(distinguish)된다고 말한다.

DFA의 상태수 최소화 방법

1. 초기의 동치관계를 구성한다. 즉, 전체 상태를 final state와 nonfinal state의 두 동치류(equivalence class)로 분할(partition)한다.
2. 같은 input symbol에 대해 서로 다른 동치류로 가는 arc가 존재하면, 또 다른 분할을 하여 새로운 동치류를 만든다.
   (그 input symbol에 의해 구별된다고 함)
3. 2.를 반복하여 더 이상 분할이 일어나지 않을 때,
   DFA $M' = (Q', \Sigma, \delta', q_0', F')$을 다음과 같이 구성한다.
   (1) Q' : 동치류의 집합. Q'의 한 state를 $[q]$로 표시하며, 그 의미는 상태 q를 포함하는 동치류를 나타낸다.
   (2) $q_0' = [q_0]$
   (3) $[p]$, $[q]$를 임의의 동치류라고 할 때, $\delta(p,a)=q$이면 $\delta'(p,a)=[q]$이다.
   (4) $F' = \{ [q] \mid q \in F \}$

예27) 다음 DFA의 상태수를 최소화하시오.



1. final state인지 아닌지에 따라 동치 class {*A, B, D*}와 {*C, E*}로 분할한다.

	1 : {A, B, D}	2 : {C, E}
a	1 1 1	1 1
b	1 2 2	2 2

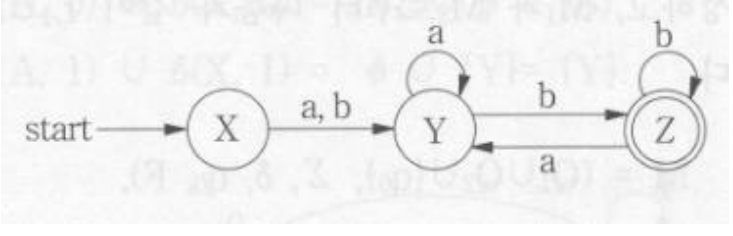
2. 각 동치 class가 각 input symbol에 대해 구별되는가를 조사하여 더 이상 분할이 일어나지 않을 때까지 반복한다.
   동치 class 1:{*A, B, D*}는 input symbol *b*에 대하여 {*A*}와 {*B, D*}로 분할.

	1 : {A}	2 : {B, D}	3: {C, E}
a	2	2 2	2 2
b	2	3 3	3 3

3. 더 이상 분할이 되지 않으므로 *X=[A], Y=[B,D], Z=[C,E]*로 놓고 DFA *M′*을 구성. 이를 상태전이도로 표현하면 다음과 같다.

   M′ = ({X,Y,Z}, {a,b}, δ′, X, {Z})

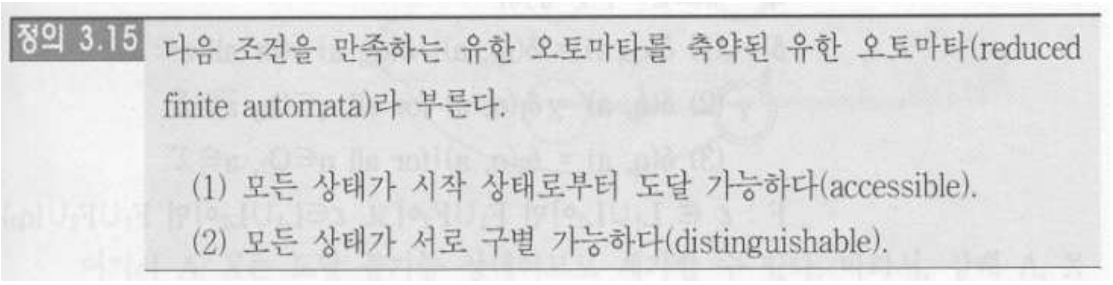
δ′	a	b
X	Y	Y
Y	Y	Z
Z	Y	Z



[정의 3.15] 축약된 유한 자동 (reduced finite automata)

(1) 모든 상태가 start state로부터 도달 가능하다(accessible).

(2) 모든 상태가 서로 구별 가능하다(distinguishable).


정의 3.15 다음 조건을 만족하는 유한 오토마타를 축약된 유한 오토마타(reduced finite automata)라 부른다.

(1) 모든 상태가 시작 상태로부터 도달 가능하다(accessible).
(2) 모든 상태가 서로 구별 가능하다(distinguishable).

※ 축약된 유한 자동으로 변환 방법

[단계 1] 들어오는 arc가 없이 나가는 arc만 갖는 inaccessible state를 모두 제거

[단계 2] 동치관계를 이용하여 구별되는 않는 상태들을 하나로 merge한 후에 새로운 automata를 구성.

3.3.6 유한 자동의 닫힘 성질

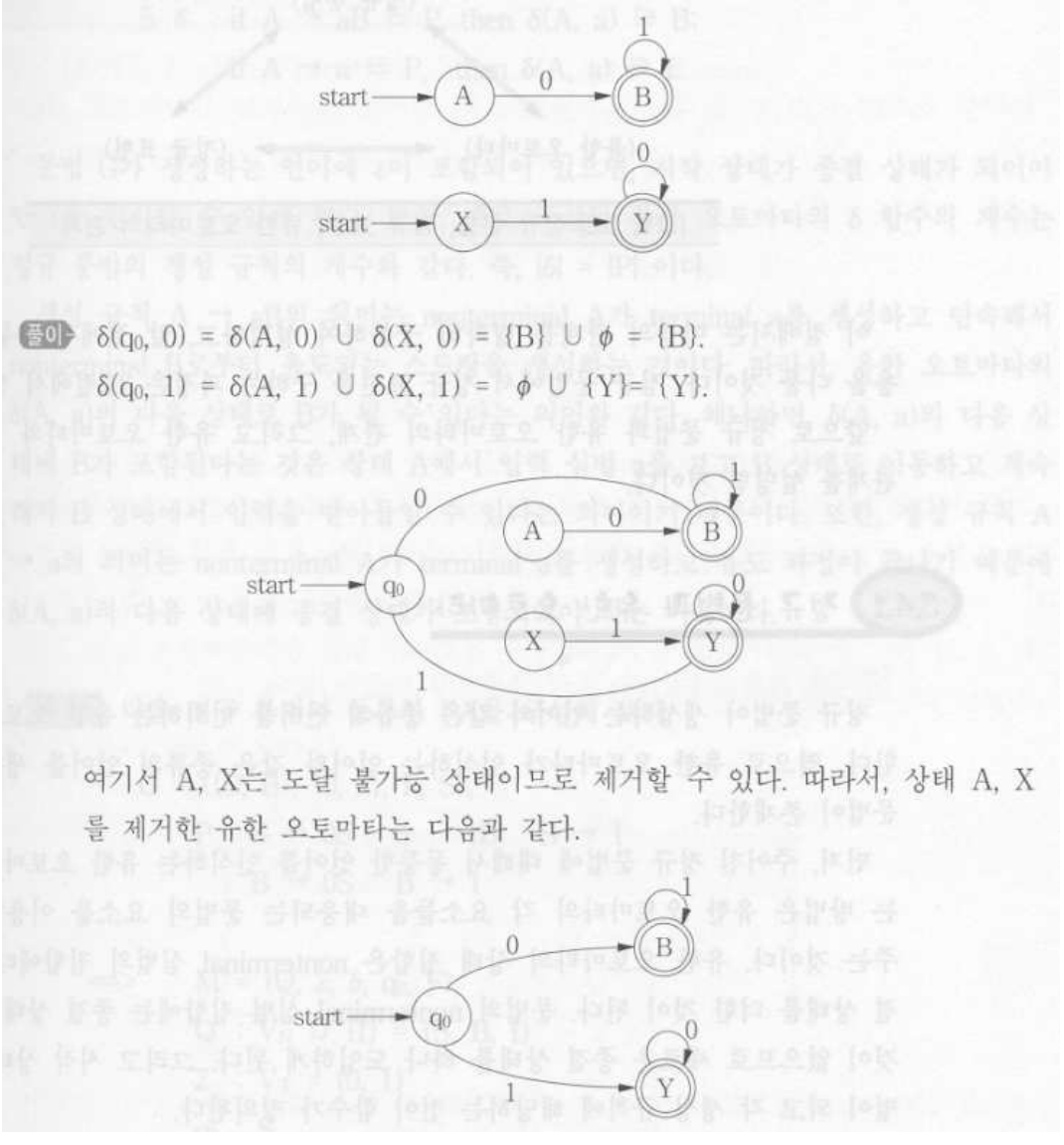
FAL : Finite Automata Language

[정리 3.3] L_1, L_2가 FAL이면 다음도 역시 FAL이다. ⟶ 증명은 교재 참조

1. $L_1 \cup L_2$
2. $L_1 L_2$
3. L_1^*

※ 유한 자동 언어는 합집합, 접속(concatenation), closure에 대하여 닫혀 있다.

예28) 다음과 같은 두 F.A.를 merge하시오.



3.4 정규 언어의 속성

정규 문법 ⇔ 정규표현 ⇔ 유한 자동

3.4.1 정규 문법과 유한 자동

※ 정규 문법을 인식하는 유한 자동의 구성 방법

Given G = (V_N, V_T, P, S), construct M = ($Q, \Sigma, \delta, q_0, F$)

1. Q : $V_N \cup \{f\}$, f는 새로 만들어진 final state
2. Σ : V_T
3. q_0 : S
4. F : $\varepsilon \notin L(G)$, then $\{f\}$ else $\{S, f\}$
5. δ : if $A \rightarrow aB \in P$, then $\delta(A, a) \ni B$
   if $A \rightarrow a \in P$, then $\delta(A, a) \ni f$

예29) 정규문법 G=($\{S, B\}$, $\{0, 1\}$, P, S)는 아래 FA로 변환된다.

P : $S \rightarrow 0S$ $S \rightarrow 1B$ $S \rightarrow 0$ $S \rightarrow 1$
$B \rightarrow 0S$ $B \rightarrow 0$

⇒ $M = (Q, \Sigma, \delta, q_0, F)$
Q : $V_N \cup \{f\} = \{S, B, f\}$
Σ : $V_T = \{0, 1\}$
q_0 : S
F : $\{f\}$

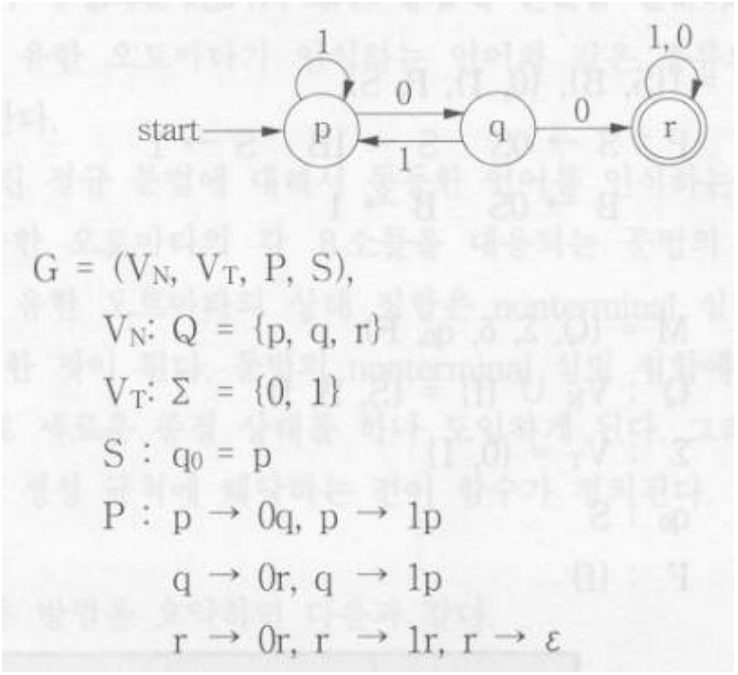
δ	0	1
S	$\{S, f\}$	$\{B, f\}$
B	$\{S, f\}$	ϕ
f	ϕ	ϕ

※ 유한 자동으로부터 정규 문법을 구성하는 방법

Given $M = (Q, \Sigma, \delta, q_0, F)$, construct $G = (V_N, V_T, P, S)$

1. V_N : Q
2. V_T : Σ
3. S : q_0
4. P : if $\delta(A, a) = r$ then $q \rightarrow ar$
   if $q \in F$ then $q \rightarrow \varepsilon$

예30) 다음 DFA를 정규문법으로 변환하시오.



G = (V_N, V_T, P, S),
V_N: Q = {p, q, r}
V_T: Σ = {0, 1}
S : q_0 = p
P : p → 0q, p → 1p
q → 0r, q → 1p
r → 0r, r → 1r, r → ε

3.4.2 유한 자동과 정규 수식

※ 유한자동으로부터 정규표현을 구하는 과정

(1) 유한자동으로부터 정규문법으로 구한다.
(2) 정규문법으로부터 정규표현을 구한다.

연습 : <예30>의 유한 자동으로부터 정규표현을 얻는 과정.

δ	0	1
p	*q*	*p*
q	*r*	*p*
r	*r*	*r*

1) 상태전이표로부터 정규문법 구한다.

$p \rightarrow 0q$ $\quad p \rightarrow 1p$
$q \rightarrow 0r$ $\quad q \rightarrow 1p$
$r \rightarrow 0r$ $\quad r \rightarrow 1r$ $\quad r \rightarrow \varepsilon$

2) 정규문법으로부터 정규표현식

$p = 0q + 1p$
$q = 0r + 1p$
$r = 0r + 1r + \varepsilon$

3) 정규표현을 구한다.

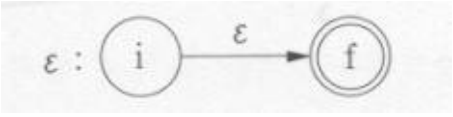
$L(M) = (01+1)^*00(0+1)^*$

※ 정규표현으로부터 유한자동을 구하는 과정

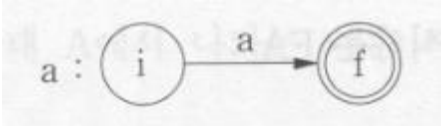
(1) 정규표현의 정의로부터 NFA(ε-NFA)를 구성한다.
(2) simplification 방법을 적용하여 크기를 줄인다.
(3) ε-NFA를 DFA로 변환한다.

1. 정규표현 ε, *a*에 대하여

(1) ε을 인식하는 NFA

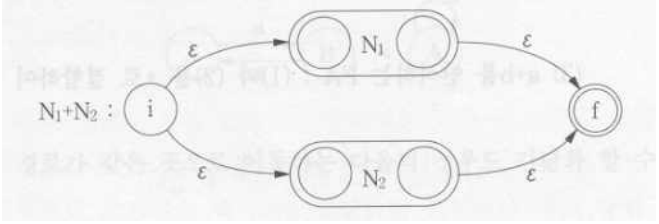


(2) 심벌 a를 인식하는 FA



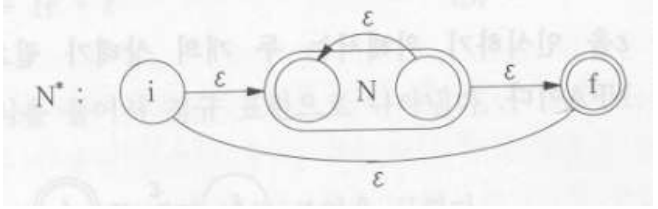
2. 정규표현 N_1+N_2, $N_1 \bullet N_2$, N^*에 대하여

(1) N_1+N_2 구성하는 방법



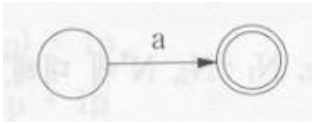
(2) $N_1 \bullet N_2$ 구성하는 방법

(3) N^* 구성하는 방법

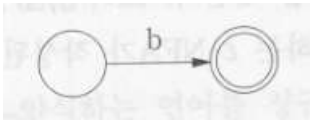


예31) 정규표현 $(a+b)^*$를 인식하는 유한자동을 구성하시오.

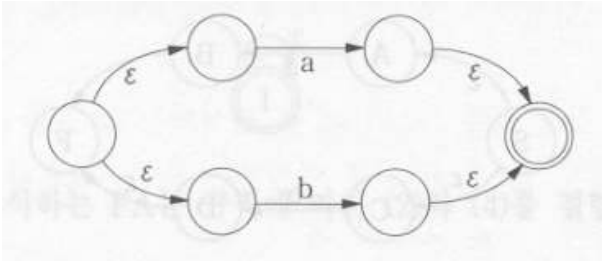
(1) a를 인식하는 FA



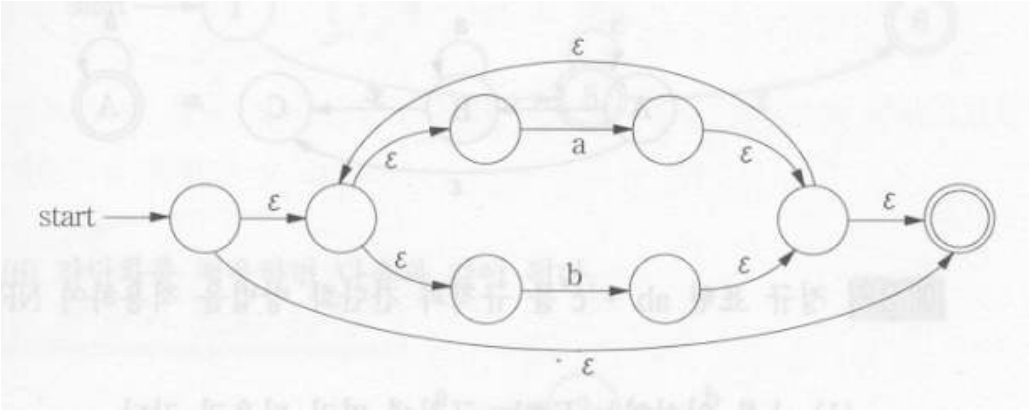
(2) b를 인식하는 FA



(3) a+b를 인식하는 FA

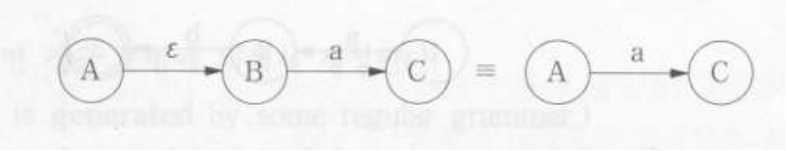


(4) $(a+b)^*$를 인식하는 FA

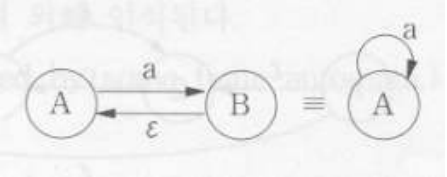


※ ε-NFA를 간단화하는 방법

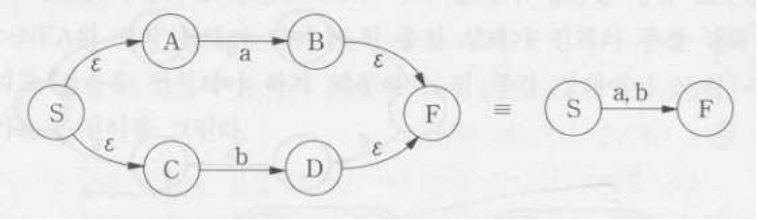
1. 상태 A에서 나가는 다른 지시선이 없을 때 A, B는 같은 상태로 취급



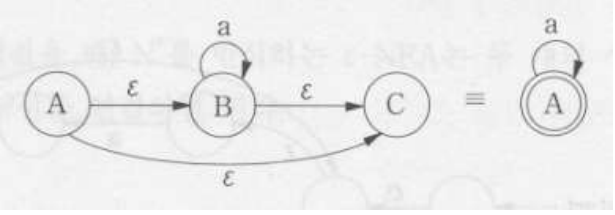
2. 다음과 같은 형태를 하나의 상태로 축약



3. 두 개의 경로가 같은 곳으로 이동하는 경우

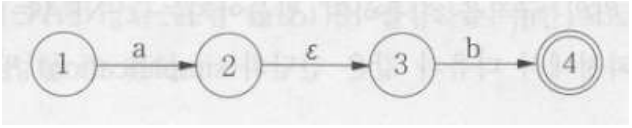


4. a^*를 인식하는 NFA

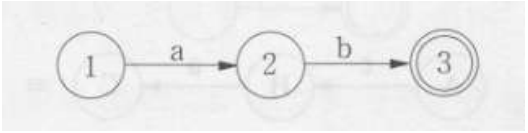


예32) 정규표현 $ab+c^*$를 NFA로 변환하시오.

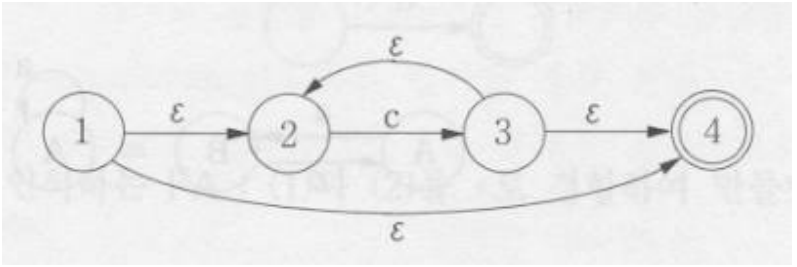
(1) ab를 인식하는 FA

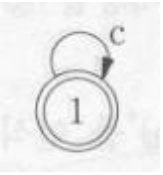


(2) 간단화 방법을 적용

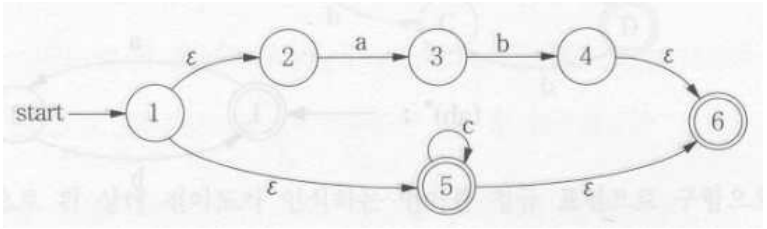


(3) c^*를 인식하는 NFA --> 간단화 방법 적용

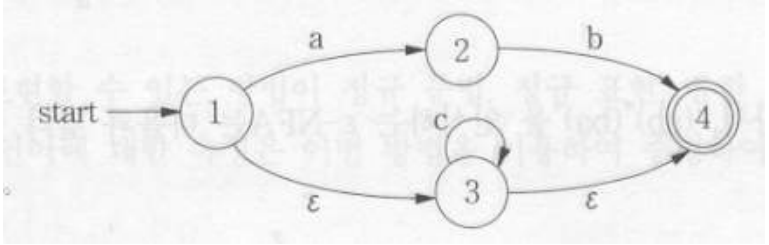




(4) $ab+c^*$를 인식하는 NFA

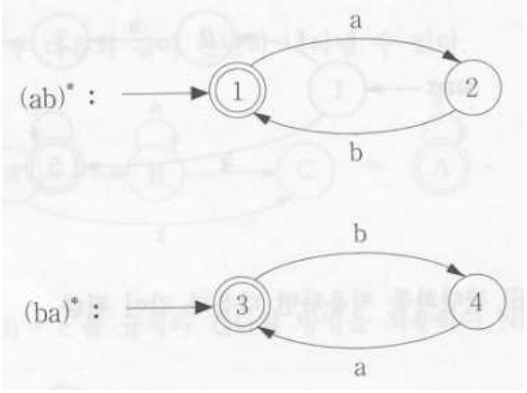


(5) 간단화 방법을 적용

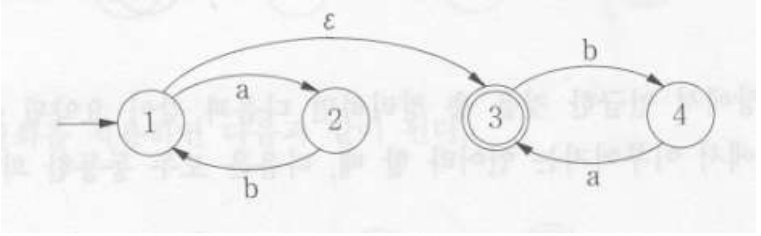


예33) 정규표현 $(ab)^*(ba)^*$를 인식하는 DFA를 고안하시오.

(1) $(ab)^*$, $(ba)^*$를 인식하는 NFA



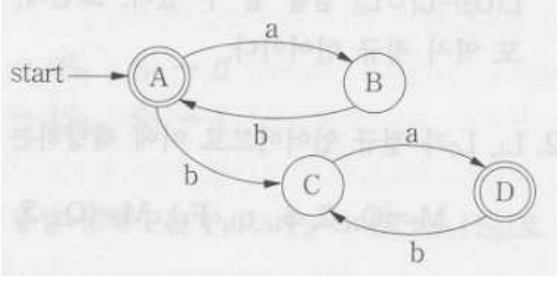
(2) $(ab)^*(ba)^*$를 인식하는 NFA



(3) DFA로 변환하여 상태전이표 구성

δ	a	b
A=[1, 3]	[2]	[4]
B=[2]	ϕ	[1,3]
C=[4]	[3]	ϕ
D=[3]	ϕ	[4]

(4) 상태전이도 구성



3.4.3 정규 언어의 닫힘 성질

[정리 3.4] L_1, L_2가 정규언어이면 다음 언어도 역시 정규언어이다.

1. $L_1 \cup L_2$
2. $L_1 \bullet L_2$
3. L_1^*

※ 정규언어는 합집합, 접속(concatenation), closure에 대하여 닫혀 있다.

예34) 다음 두 문법이 생성하는 언어의 합집합을 생성하는 문법을 구성하시오.

P_1 : $S_1 \rightarrow 0S_1 \quad S_1 \rightarrow 0$
P_2 : $S_2 \rightarrow 0S_2 \quad S_2 \rightarrow 1$

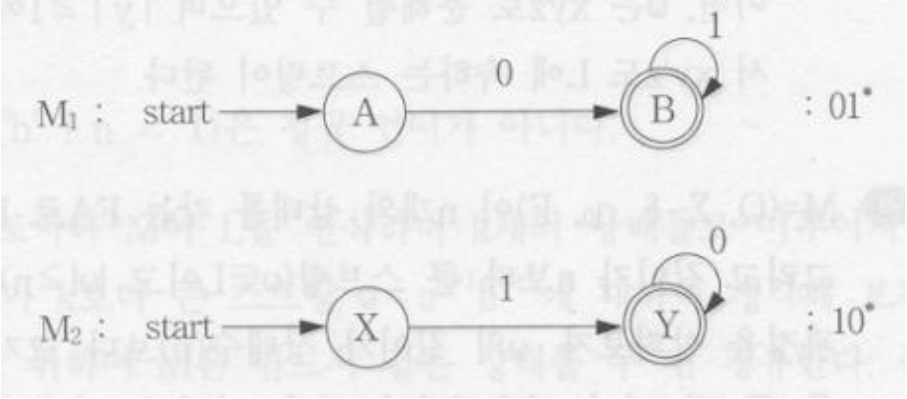
(1) 합집합 생성

P : $S \rightarrow S_1 \quad S \rightarrow S_2$
$S_1 \rightarrow 0S_1 \quad S_1 \rightarrow 0$
$S_2 \rightarrow 0S_2 \quad S_2 \rightarrow 1$

(2) 단일생성 규칙 제거

P : $S \rightarrow 0S_1 \mid 0 \mid 0S_2 \mid 1$
$S_1 \rightarrow 0S_1 \mid 0$
$S_2 \rightarrow 0S_2 \mid 1$

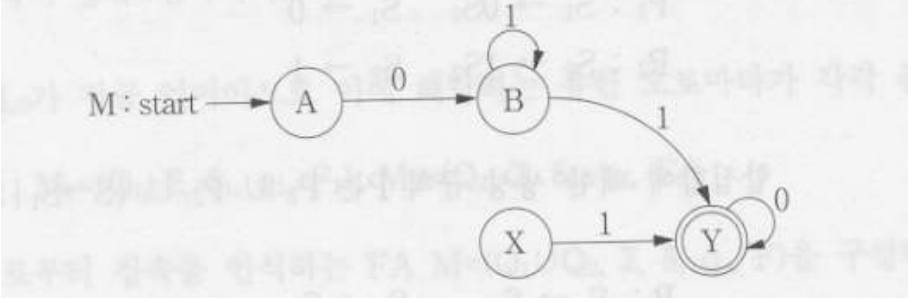
예35) 다음 두 FA를 접속한 FA를 구성하시오.



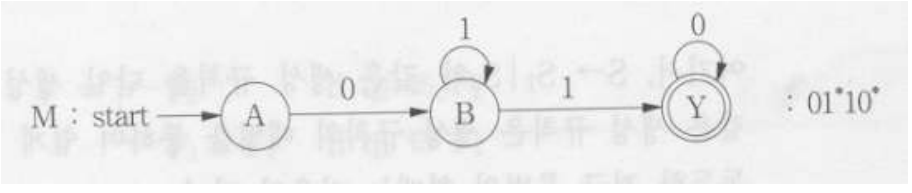
(1) M_1의 final state B에 대해 다음을 구해서 지시선 추가

$\delta(B, 0) = \delta_1(B, 0) \cup \delta_2(X, 0) = \phi.$

$\delta(B, 1) = \delta_1(B, 1) \cup \delta_2(X, 1) = \{B\} \cup \{Y\} = \{B, Y\}.$



(2) X가 inaccessible state이므로 제거



3.4.4 정규 언어에 대한 Pumping Lemma

Pumping Lemma -- 정규언어의 속성으로 어떤 언어가 정규언어가 아님을 증명하는데 유용한 보조 정리.

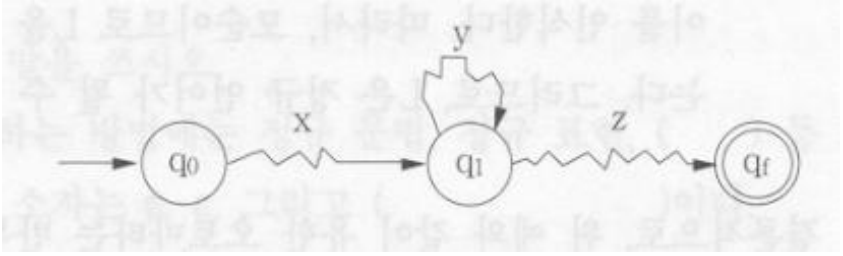
[정리 3.5] 정규 언어의 속성

> L이 정규언어라 하자. ω가 L에 속하는 string이고 어떤 p에 대하여 $|\omega| \geq p$일 때, ω는 xyz로 쓸 수 있으며, 모든 $i \geq 0$에 대해서 xy^iz는 L에 속한다. 여기서, $|y|$는 p보다 작다.
>
> (ω는 충분히 큰 string이고 y는 ω의 substring으로 반복되는 부분임)

(증명) 유한자동 언어 L을 인식하는 유한자동 M이 n개의 state를 가질 때 길이가 n보다 큰 string ω를 인식하려면 반드시 두 번 이상 경유하는 state가 있어야 한다. 이 때 두 번 이상 경유하는 state를 q_1이라 하면,

$$\delta(q_0, xyz) = \delta(q_1, yz) = \delta(q_1, z) = q_f \in F$$

를 만족하는 $\omega = xyz$가 존재한다. 즉, q_1에서 하나 이상의 state를 거쳐 y를 처리하고 나면 다시 q_1으로 되돌아가는 substring y가 존재한다. 따라서 유한자동 M에 의하여 xy^iz를 인식할 수 있으며, xy^iz도 역시 유한자동 언어 L에 속한다.



※ 길이가 충분히 큰 string이 정규 언어에 속하려면 반드시 xyz의 형태로 표시되며, xy^iz도 역시 그 정규 언어에 속한다.

예36) $L = \{a^n b^n \mid n \geq 1\}$은 정규언어가 아님을 증명하시오.

(증명) L을 인식하는 유한자동 M이 k개의 state로 구성된다고 가정할 때
k보다 큰 string $\omega = a^{k+1}b^{k+1}$을 인식하기 위하여 M은 반드시 같은
state를 두 번 이상 경유하게 된다.
이 때 두 번 경유되는 state를 p라 하면,

$$\delta(q_0, \omega_1) = p,\ \delta(p,\ \omega_2) = p,\ \delta(p, \omega_3) = q_f \in F$$

를 만족하는 $\omega = \omega_1 \omega_2 \omega_3$가 존재한다.
따라서 Pumping Lemma에 의해 M은 $\omega_1 \omega_2 \omega_2 \omega_3$을 인식하게 된다.
만약 반복되는 substring ω_2가 모두 a로 이루어져 있다면
$\omega_1 \omega_2 \omega_2 \omega_3$는 b보다 a의 개수가 더 많아지고,
ω_2가 모두 b로 이루어져 있다면, $\omega_1 \omega_2 \omega_2 \omega_3$는 a보다 b의 개수가
더 많아진다.
ω_2가 모두 a,b로 이루어져 있다면, $\omega_1 \omega_2 \omega_2 \omega_3$는 $a^m b^n a^p b^l$형태가 된다.
따라서 $\omega_1 \omega_2 \omega_2 \omega_3$는 L에 속하지 않게 되는데 M이 이를 인식하는 것은
모순이므로 L을 인식하는 어떠한 유한자동도 존재하지 않으며
L은 정규언어가 될 수 없다.

<숙제> 3장 연습문제 풀이