계산이론

2022년 1학기

이은주

3장 정규 언어와 정규 문법
정규 표현
정규 표현과 정규 언어의 관계
정규문법
정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

문법(Grammar)

- [정의] 문법 G = (N , ∑ , P , S), G = (N , ∑ , S, P)
  - N : nonterminal 기호의 집합(variable이라고도 한다)
  - ∑ : terminal 기호의 집합
  - P : $\alpha \rightarrow \beta$ 형태의 **생성규칙**(production)의 집합

    (단, $\alpha \in (N \cup \sum)^{*} N (N \cup \sum)^{*}$ and $\beta \in (N \cup \sum)^{*}$)
  - S : **시작 기호**(start symbol) ∈ N

- $\alpha \rightarrow \beta 1$ ⇨ $\alpha \rightarrow \beta 1 \mid \beta 2$

  $\alpha \rightarrow \beta 2$

Chomsky Hierarchy(촘스키의 분류)

- [정의] 문법 $G = (N, \Sigma, S, P)$ 의 분류
  - $(\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$ and $\beta \in (N \cup \Sigma)^*$)

- **Type 3** : **정규(regular) 문법** (언어 예 : a^n)
  a) **우선형**(right-linear) 문법 : 생성규칙: $\mathbf{A} \rightarrow xB$ 또는 $\mathbf{A} \rightarrow x$ $(A, B \in N \; and \; x \in \Sigma^*)$
  b) **좌선형**(left-linear) 문법 : 생성규칙: $\mathbf{A} \rightarrow Bx$ 또는 $\mathbf{A} \rightarrow x$

- **Type 2** : **문맥자유(context-free) 문법** (언어 예 : $a^n b^n$)
  생성규칙: $\mathbf{A} \rightarrow \beta (A \in N)$

- **Type 1** : **문맥연관(의존)(context-sensitive) 문법** (언어 예 : $a^n b^n c^n$)
  생성규칙: $\alpha A \beta \rightarrow \alpha \gamma \beta$
  (항상 $\alpha \rightarrow \beta$ 에서 $|\alpha| \leq |\beta|$, non-contracting(줄어들지 않는) 문법, S → λ 은 허용)

- **Type 0** : **무제한(unrestricted) 문법**
  생성규칙: $\boldsymbol{\alpha} \rightarrow \beta$ $(\boldsymbol{\alpha} \neq \lambda)$
  (모든 종류의 형식 문법 포함)

- 정규 언어를 표현하는 두 가지 방법
  - 정규 표현(regular expression)
  - 정규 문법(regular grammar)

- 임의의 언어가 정규 언어가 되기 위해서
  - 유한 인식기 존재
    - 모든 정규 언어의 기술 : dfa, nfa
    - 특정 문자열이 주어진 언어에 속하는지를 결정할 때 : 논리적인 판단 과정에 사용
    - 대부분의 경우 : 정규 언어를 기술하는 보다 정교한 방법이 필요함

정규 표현

- 정규 표현(regular expression)
  - 표기법 : 알파벳 심벌들의 문자열, 괄호, 연산자 +, ·, * 등 사용

    예1) 언어 {a} : 정규 표현 a

    예2) 언어 {a, b, c} : 정규 표현 a+b+c
  - 기호 · : 접합(concatenation)의 의미
  - 기호 * : 스타-폐포(star-closure)의 의미
  - 정규 표현 (a+b · c)*
    - {a} ∪ {bc}의 스타-폐포라는 뜻
    - 언어{λ, a, bc, aa, abc, bca, bcbc, aaa, aabc, ...}

정규 표현의 정의

- [정의 3.1] (84page) Σ를 주어진 알파벳이라 할 때 정규 표현의 정의:

  (1) ϕ, λ와 a $\in$ Σ : 모두 정규 표현

  - 기본 정규 표현(primitive regular expression)

  (2) r_1과 r_2가 정규 표현 $\rightarrow r_1 + r_2, r_1 \cdot r_2, {r_1}^*, (r_1)$등은 모두 정규 표현

  (3) 특정 문자열의 정규 표현

  - 기본 정규 표현에서 시작하여 위 규칙 (2)를 유한 횟수만큼 반복함으로써 해당 문자열이 유도될 수 있어야 함

정규 표현

- 예제 3.1(84page) Σ = {a, b, c}에 대하여, 다음 문자열은 [정의3.1] 규칙에 따라 구성되기 때문에 정규 표현

$$(a + b \cdot c)^* \cdot (c + \emptyset)$$

예) $r_1 = c$라 하고 $r_2 = \emptyset$이라 하면

- $c + \emptyset$과 $(c + \emptyset)$은 정규 표현
- 이러한 과정을 반복 : 위의 식 전부가 생성될 수 있음

예) $(a + b +)$

- 기본 정규 표현으로부터 이를 형성하는 방법이 없으므로 정규 표현이 아님

정규 표현과 관련된 언어

- [정의 3.2] (85page) 정규 표현 r에 의해 묘사되는 언어 L(r) 정의
  1. ∅ : 공집합을 나타내는 정규 표현
  2. λ : {λ}를 나타내는 정규 표현
  3. 모든 a ∈ Σ에 대해 a : {a}를 나타내는 정규 표현

  r_1과 r_2가 정규 표현일 경우
  4. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
  5. $L(r_1 \cdot r_2) = L(r_1)L(r_2)$
  6. $L\big((r_1)\big) = L(r_1)$
  7. $L({r_1}^*) = \big(L(r_1)\big)^*$

정규 표현과 관련된 언어

- 예제 3.2 (85page) 언어 $L\left(a^* \cdot (a+b)\right)$를 집합 형태로 표현하면?

$$
\begin{aligned}
L\left(a^* \cdot (a+b)\right) &= L(a^*)L(a+b) \\
&= \left(L(a)\right)^* \left(L(a) \cup L(b)\right) \\
&= \{\lambda, a, aa, aaa, \cdots\}\{a, b\} \\
&= \{a, aa, aaa, \cdots, b, ab, aab, \cdots\}
\end{aligned}
$$

정규 표현과 관련된 언어

- [정의 3.2] (85page) 정규 표현 r에 의해 묘사되는 언어 L(r) 정의
  1. ∅ : 공집합을 나타내는 정규 표현
  2. λ : {λ}를 나타내는 정규 표현
  3. 모든 a ∈ Σ에 대해 a : {a}를 나타내는 정규 표현

  r_1과 r_2가 정규 표현일 경우

  4. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
  5. $L(r_1 \cdot r_2) = L(r_1)L(r_2)$
  6. $L((r_1)) = L(r_1)$
  7. $L(r_1^*) = (L(r_1))^*$

정규 표현과 관련된 언어

- 모호성(ambiguity) 예 : 정규 표현 $a \cdot b + c$에서

$r_1 = a \cdot b,\ \ r_2 = c$:

$$\mathrm{L}(a \cdot b + c) = \{ab, c\}$$

$r_1 = a,\ \ r_2 = b + c$:

$$\mathrm{L}(a \cdot b + c) = \{ab, ac\}$$

  - 해결
    - 연산자 우선순위 부여 기법을 사용
      - 스타-폐포 : 접합보다 우선
      - 접합 : 합집합 연산보다 우선
      - 접합 연산자는 생략 가능 : $r_1 r_2 = r_1 \cdot r_2$

정규 표현

- [정의] **정규 표현** 및 그 정규표현이 나타내는 **정규 집합**

  (1) 정규표현 **λ** ⇨ 정규집합 {λ}

  (정규표현 λ : 정규집합 {λ}을 나타냄)

  (2) 정규표현 **a** ⇨ 정규집합 {a}

  (3) 정규표현 p, q 가 각각 정규집합 P, Q 를 나타내면

  (a) 정규표현 **(p+q)** ⇨ 정규집합 P∪Q

  (b) 정규표현 **(pq)** ⇨ 정규집합 PQ

  (c) 정규표현 **(p)*** ⇨ 정규집합 P^*

정규 표현과 관련된 언어

- 예제 3.3 (86page) Σ = {a, b}에 대해, 다음의 정규 표현이 묘사하는 언어?

$$r = (a + b)^*(a + bb)$$

$$L(r) = \{a, bb, aa, abb, ba, bbb, \cdots\}$$

  - $(a + b)$: 문자 a와 b로 이루어지는 모든 문자열
  - $(a + bb)$: a 또는 bb
  - $L(r)$: a 또는 bb로 끝나는 알파벳 $\{a, b\}$에 대한 모든 문자열들의 집합

정규 표현과 관련된 언어

- 예제 3.4 (86page) Σ = {a, b}에 대해, 다음의 정규 표현이 묘사하는 언어?

$$r = (aa)^*(bb)^*b$$

$$L(r) = \{b, aab, bbb, aabbb, \cdots\}$$

  - 짝수 개의 a와 이어서 홀수 개의 b를 갖는 모든 문자열들로 이루어지는 언어

$$L(r) = \{a^{2n}b^{2m+1} : n \geq 0, m \geq 0\}$$

정규 표현과 관련된 언어

- 예제 3.5 (87page) Σ = {0, 1}에 대해, 다음 언어에 대한 정규 표현은?

$$L(r) = \{w \in \Sigma^* : w\text{는 적어도 한 쌍의 연속된 0을 가지고 있다}\}$$

  - $L(r)$에 속하는 모든 문자열
    - 어디엔가 00을 갖게 되며, 그 앞의 문자열이나 그 뒤에 오는 문자열은 어떤 형태를 갖든 상관없음

  - 알파벳 $\{0,1\}$에 대한 모든 문자열 : $(0+1)^*$

$$r = (0+1)^*00(0+1)^*$$

정규 표현과 관련된 언어

- 예제 3.6 (87page) 다음 언어를 묘사하는 정규 표현은?

$$L = \{w \in \{0,1\}^* : w\text{는 연속된 0들의 쌍을 갖지 않는다}\}$$

  - 이 언어에 속하는 문자열에 0이 나타날 경우
    - 뒤에는 반드시 1
    - 부문자열 : 앞이나 뒤에 임의의 개수의 1

$$(1^*011^*)^*$$

  - 0으로 끝나는 문자열이나 완전히 1로만 구성되는 문자열 고려

$$r = (1^*011^*)^*(0 + \lambda) + 1^*(0 + \lambda)$$

  - 언어 L이 1과 01의 반복으로 이루어짐을 고려하면

$$r = (1 + 01)^*(0 + \lambda)$$

정규 표현

[ex]

① 정규표현 **01 :** 집합 {01}

② $\mathbf{0^*}$ ⇨ { λ, 0, 00, 000, … }

③ $\mathbf{(a+b)^*}$ ⇨ { λ, a, b, aa, ab, ba, bb, aaa, … }

④ $\mathbf{(0+1)^*011}$

⑤ $\mathbf{(a+b)(a+b+0+1)^*}$

⑥ $\mathbf{(a+bc)^*(c+\lambda)}$

⑦ $\mathbf{a^*}$

$\mathbf{(aa)^*}$

$\mathbf{(bb)^*b}$

⑧ $\mathbf{(aa)^*(bb)^*b}$

⑨ $\mathbf{(0+1)^*00(0+1)^*}$

정규 표현

[ex]

⑩ $(1+10)^*$ ⇨ { 1로 시작하고 연속된 두 개의 0을 가지지 않는 스트링 }

⑪ $(a+b)(a+b) = (a+b)^2$ ⇨ $\{ w \mid |w| = 2, w \in \{a,b\}^* \}$

$((a+b)^2)^*$ ⇨ $\{ w \mid |w| \bmod 2 = 0, w \in \{a,b\}^* \}$

$((a+b)^3)^*(a+b)$ ⇨ $\{ w \mid |w| \bmod 3 = 1, w \in \{a,b\}^* \}$

⑫ { 1로 시작하면서 하나의 0 만을 포함하는 스트링 } 11*01*

⑬ $\{ a^n b \mid n \geq 0 \}$ a*b

⑭ $\{ awa \mid w \in \{a,b\}^* \}$ a(a+b)*a

⑮ $\{ a^n b^m, n \geq 0, m \geq 0 \}$ a*b*

⑯ $\{ (ab)^n , n \geq 0 \}$ (ab)*

⑰ $\{ aw1aaw2a \mid w1,w2 \in \{a,b\}^* \}$ a(a+b)*aa(a+b)*a

정규 표현과 관련된 언어

- 임의의 주어진 언어에 대하여 : 무한히 많은 정규 표현

- 정규 표현들 사이의 동치 관계(equivalence relation)
  - 두 개의 정규 표현이 같은 언어를 묘사하는 경우

정규 표현

- 정규 표현의 항등 관계

  a) $r\lambda = \lambda r = r$

  b) $a \cdot a^* = a^* \cdot a$

  $(r^*)^* = r^*$

  c) $(rs)^* r = r(sr)^*$

  d) $(r^* + s^*)^* = (r + s)^*$

  $(r^* s^*)^* = (r + s)^*$

  $\boldsymbol{r^*(sr^*)^* = (r^* s)^* r^* = (r + s)^*}$

정규 표현

[정의] **어떤 정규 집합을 정의하는 정규 표현은 유일하지 않다.**
서로 다른 정규 표현이 동일한 정규 집합을 정의하는 경우 : **동치**

[ex] $\{a,b\}^*$ 중 두 개 이상의 b를 포함하는 스트링의 집합을 나타내는 정규표현들

- $a^*ba^*b(a+b)^*$
- $(a+b)^*ba^*ba^*$
- $(a+b)^*b(a+b)^*b(a+b)^*$

정규 표현과 정규 언어의 관계

- 정규 언어와 정규 표현
  - 두 개념은 본질적으로 같음
  - 어떤 정규 언어에 대해서 이에 대응하는 정규 표현 존재
  - 어떤 정규 표현에 대해서 이에 대응하는 정규 언어 존재

정규 표현에 대한 정규 언어

- r이 정규 표현일 경우 $L(r)$은 정규 언어
  - 정의 (53page) : dfa에 의해 인식되는 언어는 정규 언어
    - dfa와 nfa의 동치성을 고려하면 nfa에 의해 인식되는 언어도 정규 언어

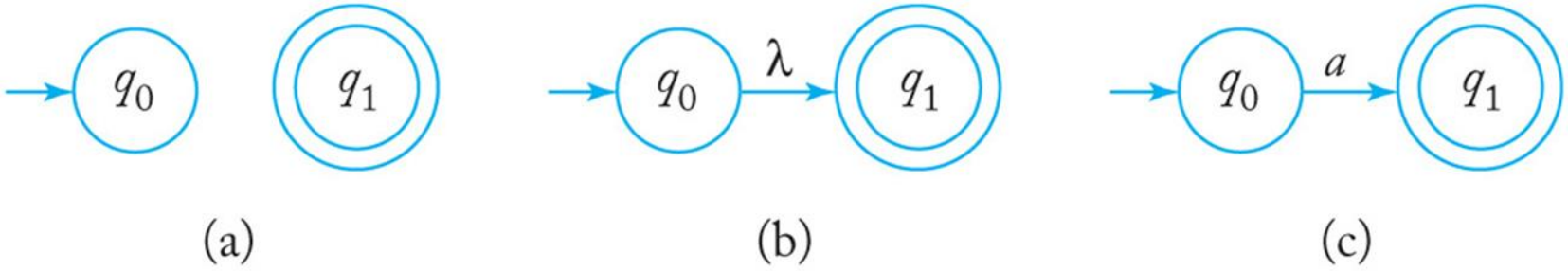
- 주어진 정규 표현 r에 대해 $L(r)$을 인식하는 nfa를 구성할 수 있음
  - nfa의 구성 : $L(r)$에 대한 재귀적 정의에 따름
  - 정의 3.2(85page) 의 (1), (2)와 (3)에 대한 간단한 오토마타 구성
  - (4), (5)와 (7)을 구현하기 위하여 이들이 결합되는지 보임

정규 표현에 대한 정규 언어

- [정리 3.1] (90page) r이 정규표현일 때
  언어 $L(r)$을 인식하는 nfa가 존재하며, 결과적으로 $L(r)$은 정규 언어

증명

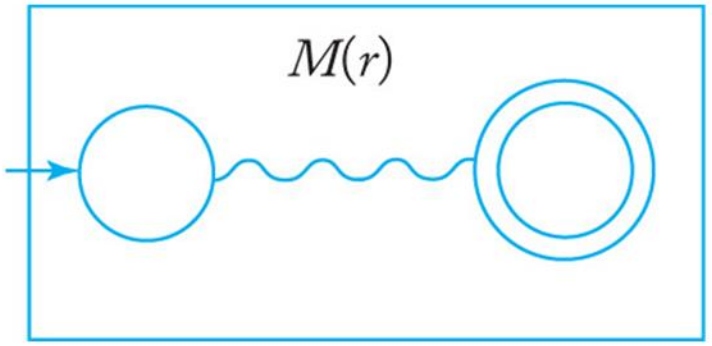
1. 우선 간단한 정규 표현인 ∅, λ와 a ∈ Σ에 대한 언어를 인식하는 오토마타를 구성



정규 표현에 대한 정규 언어

- [정리 3.1] (90page) 증명 계속

  2. 정규 표현 r_1과 r_2에 의해 묘사 되는 언어를 인식하는 오토마타 $M(r_1)$과 $M(r_2)$가 주어져 있다고 가정



$L(r)$을 인식하는 nfa의

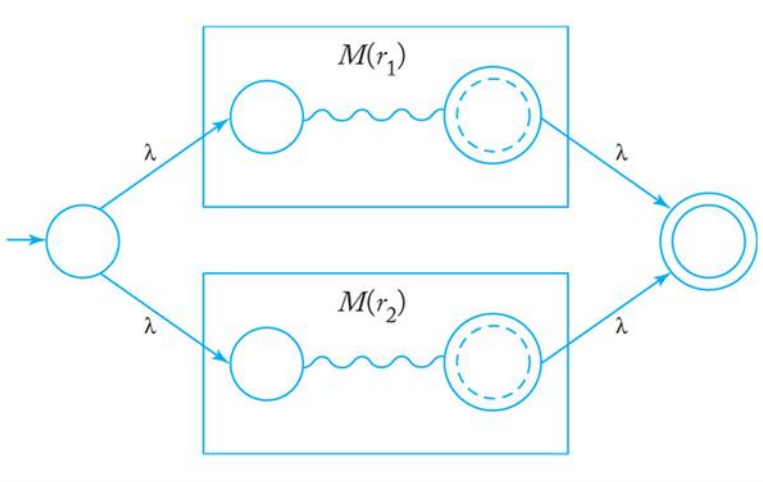
도식적인 표현 형태

정규 표현에 대한 정규 언어

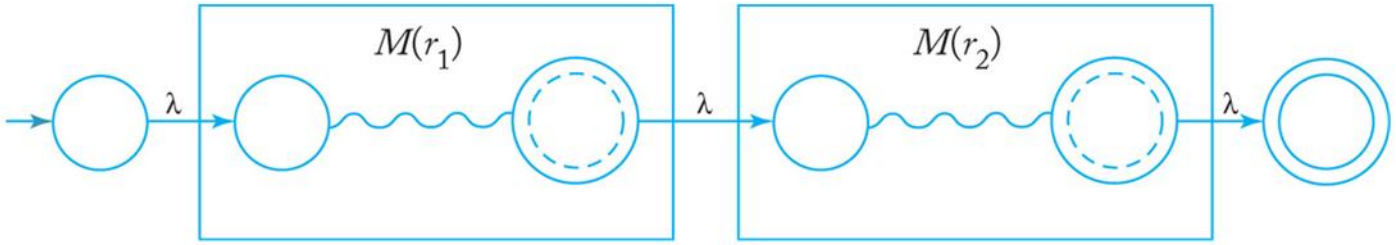
- [정리 3.1] (90page) 증명 계속

  3. 오토마타 $M(r_1)$과 $M(r_2)$를 사용하여 정규 표현 $r_1 + r_2,\ r_1 r_2,\ r_1{}^*$ 등에 대한 오토마타 구성

$L(r_1 + r_2)$에 대한 오토마타

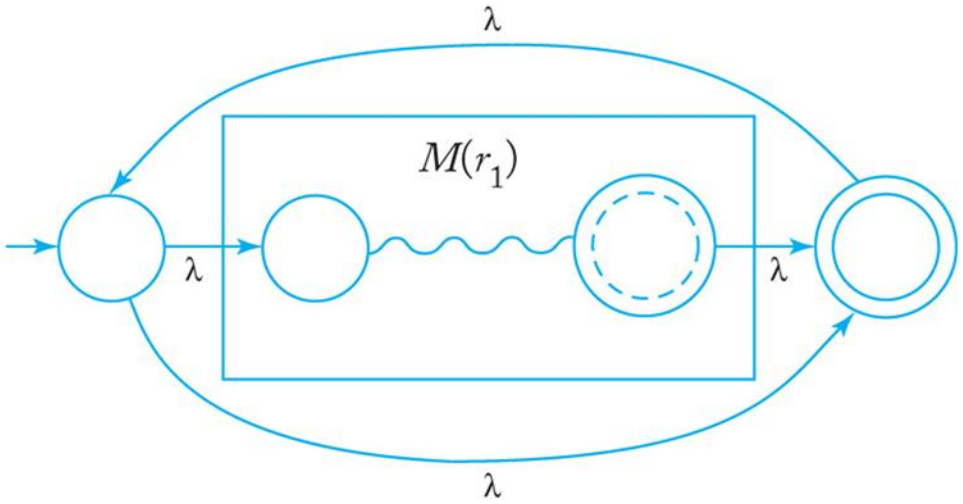


$L(\ r_1 r_2)$에 대한 오토마타



정규 표현에 대한 정규 언어

- [정리 3.1] (90page) 증명 계속
  $L(r_1{}^*)$에 대한 오토마타



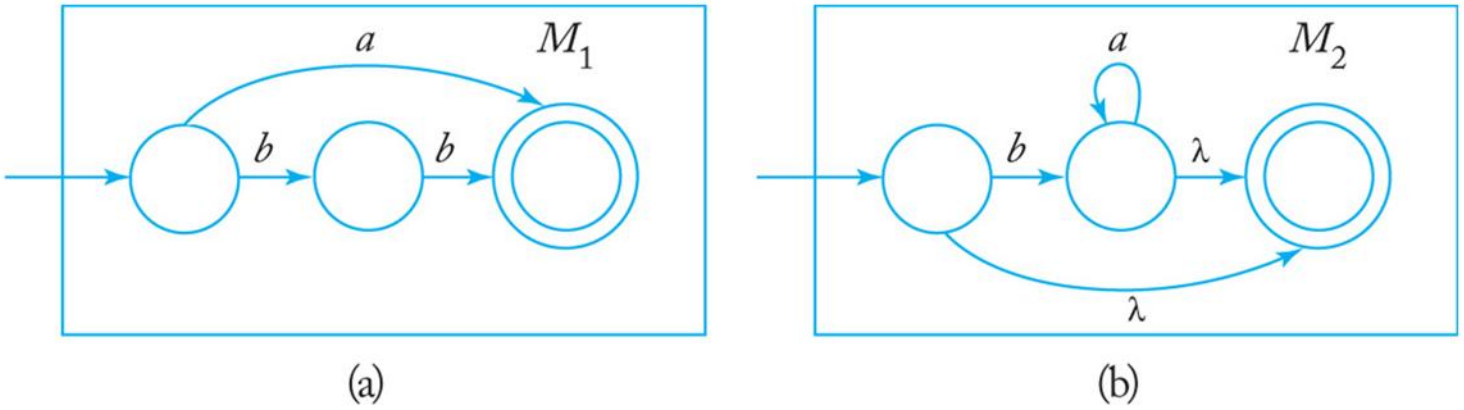
- 더 엄밀하게 논증
  - 합성된 오토마타의 상태와 전이를 구성하는 공식적인 방법 제시
- 주어진 정규 표현에 의해 묘사되는 언어를 인식하는 오토마타 증명
  - 연산자의 수를 기반으로한 귀납적 증명법

정규 표현에 대한 정규 언어

- 예제 3.7 (92page) 다음 정규 표현 r에 대해 $L(r)$을 인식하는 nfa를 구성하면?

$$r = (a + bb)^*(ba^* + \lambda)$$

1. 정규 표현 $(a + bb)$와 $(ba^* + \lambda)$에 대한 오토마타를 주어진 규칙대로 직접 구성

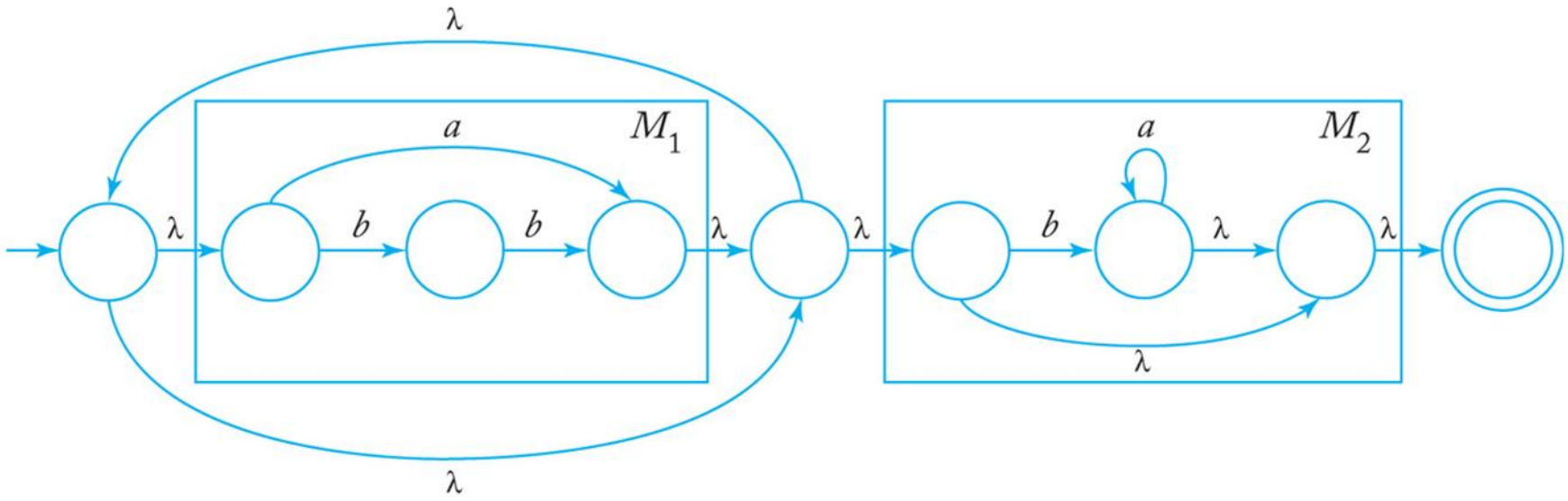


2. 정리 3.1에서 언급한 방법으로 합성

정규 표현에 대한 정규 언어

- 예제 3.7 (92page) 계속

$L\big((a+bb)^*(ba^*+\lambda)\big)$를 인식하는 오토마타



정규 언어에 대한 정규 표현

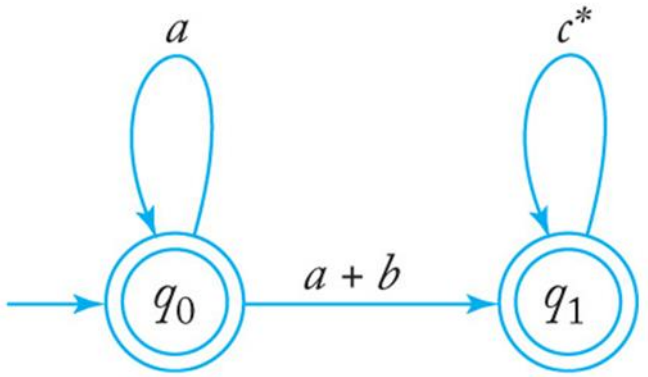
- 모든 정규 언어에 대해 대응하는 정규 표현 존재
  - 정규 언어에 대응하는 nfa
    - nfa의 초기 상태 q_0로 부터 승인 상태까지의 모든 보행에 대한 라벨들을 생성시킬 수 있는 정규 표현을 찾는 일 : 복잡
      - 전이 그래프에 사이클들이 존재, 임의의 순서로 몇 번이나 순회할 지 모름
      - 기록하는(bookkeeping) 문제 발생
      - 일반 전이 그래프(generalized transition graph, GTG)인 우회 기법 사용하여 해결

정규 언어에 대한 정규 표현

- 일반 전이 그래프(generalized transition graph, GTG)
  - 간선의 라벨에 정규 표현을 부여하는 전이 그래프
  - 초기 상태로부터 승인 상태까지의 임의의 보행에 대한 라벨
    - 여러 정규 표현들의 접합
    - 정규 표현
  - 정규 표현들이 묘사하는 문자열들은 해당 일반 전이 그래프에 의하여 인식되는 언어의 부분집합
  - 이와 같이 생성되는 모든 부분집합들의 합집합이 해당 언어가 됨

정규 언어에 대한 정규 표현

- 예제 3.8 (93page)
  일반 전이 그래프



  - 인식되는 언어 : L(a*+a*(a+b)c*)
  - 그래프에서 라벨 a인 간선 (q_0, q_0)은 임의의 개수만큼의 a를 생성시킬 수 있는 사이클 : L(a*)를 표현
    - 이 간선의 라벨을 a*로 변경하여도 이에 의해 인식되는 언어에는 아무런 영향을 미치지 않음

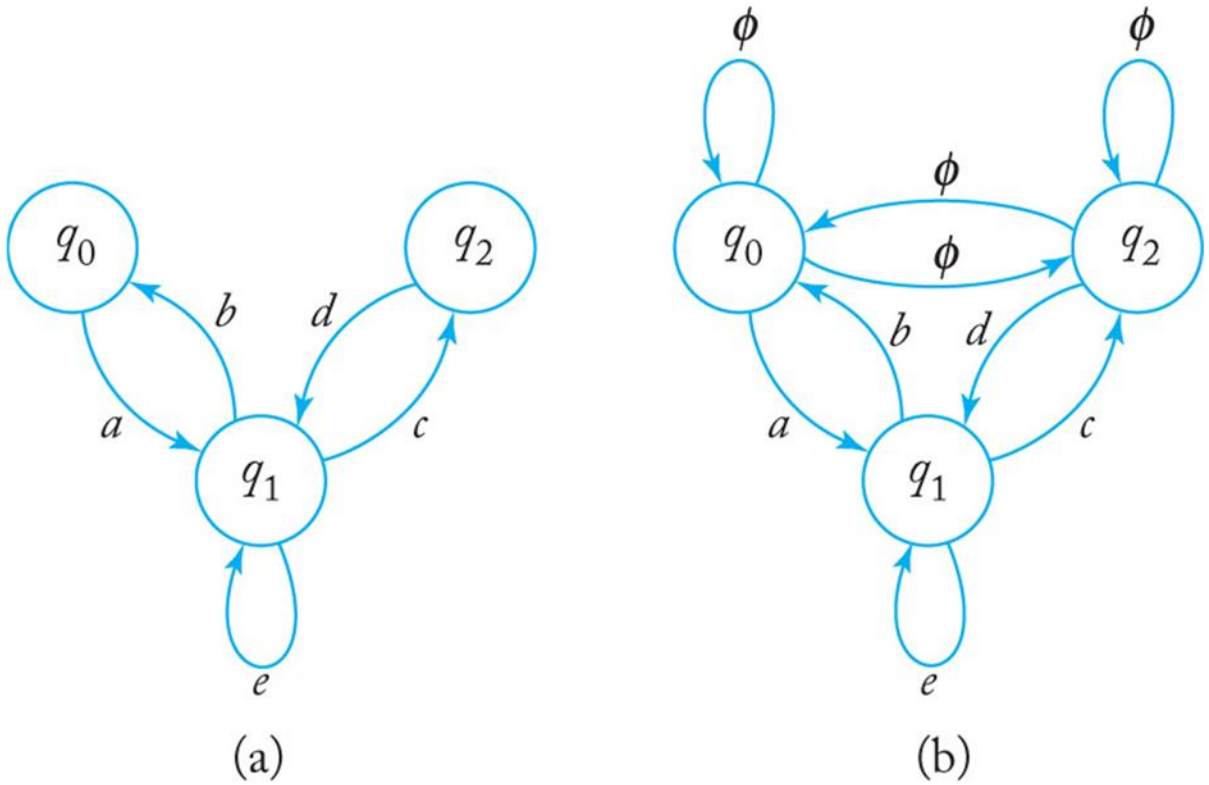
- 단일 문자 a를 라벨로 갖는 간선 : 정규 표현 a를 라벨로 갖는 것으로 해석할 수 있음
- 여러 문자 a, b, . . . 를 라벨로 갖는 간선 : 정규 표현 a+b+ . . . 을 라벨로 갖는 것으로 해석

정규 언어에 대한 정규 표현

- 완전 GTG(complete GTG)
  - 모든 간선을 포함하는 그래프
  - nfa로부터 변환된 GTG에 몇몇 간선이 존재하지 않을 경우, 그 간선들을 추가하고 ϕ를 라벨로 함
  - |V|개 정점을 갖는 완전 GTG : 정확히 $|V|^2$개의 간선

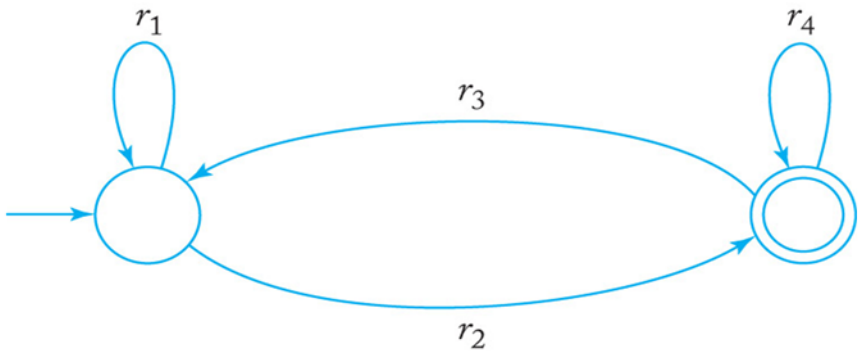
정규 언어에 대한 정규 표현

- 예제 3.9 (95page)
  - 그림(a)의 GTG : 완전 GTG 아님, 그림(b)의 GTC : 완전 GTC



정규 표현에 대한 정규 언어

- 상태의 수가 두 개인 완전 GTG



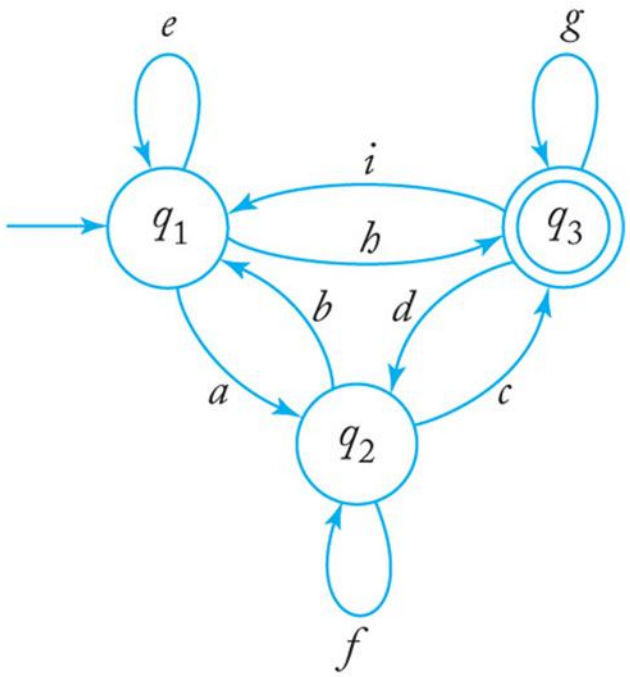
  - 주어진 그래프에 대한 올바른 정규 표현

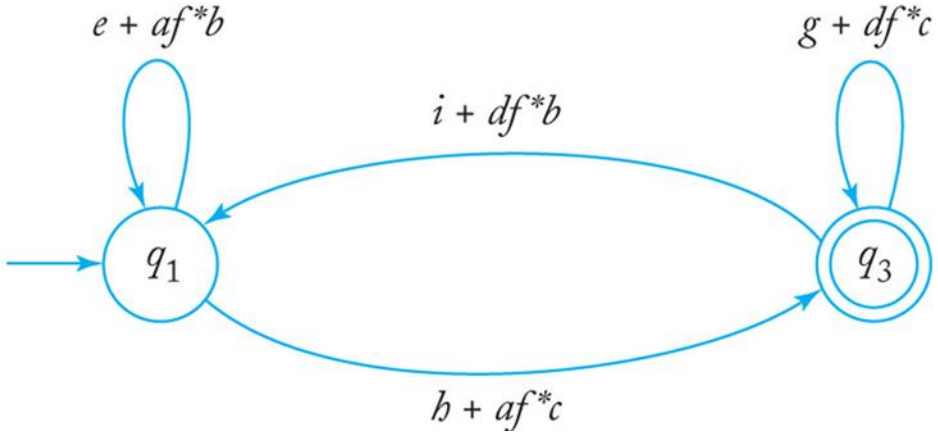
$$r = r_1{}^* r_2 (r_4 + r_3 r_1{}^* r_2)^*$$

- GTG가 2개보다 더 많은 상태를 가질 경우 : 매 단계마다 상태 하나씩을 줄임
  - 동치인 그래프를 만들어 낼 수 있음

정규 표현에 대한 정규 언어

- 예제 3.10 (95page) 완전 GTG에서 상태 줄인 동치 그래프 만들기
- q_2 제거 : 우선 몇 개의 간선을 추가
  - q_1에서 q_1으로 가는 간선 : 라벨 $e + af^*b$
  - q_1에서 q_3으로 가는 간선 : 라벨 $\mathrm{h} + af^*c$
  - q_3에서 q_1으로 가는 간선 : 라벨 $\mathrm{i} + df^*b$
  - q_3에서 q_3으로 가는 간선 : 라벨 $\mathrm{g} + df^*c$
- 상태 q_2와 이에 연결된 모든 간선들을 제거





정규 표현에 대한 정규 언어

- 완전 GTG에서 상태 줄인 동치 그래프 만들기
  1) 임의의 GTG에 대하여, 두 개의 상태가 남을 때 까지 매 단계마다 한 상태씩 제거
  2) 결과의 정규 표현을 얻기 위해 식 $r = r_1{}^* r_2 (r_4 + r_3 r_1{}^* r_2)^*$을 적용

정규 표현에 대한 정규 언어

- procedure : nfa-to-rex
  1. 상태가 $q_0, q_1, \cdots, q_n$이고 승인 상태(초기 상태와는 다른)가 하나인 nfa로 부터 시작
  2. nfa를 완전 GTG로 변환
     - r_{ij} : q_i에서 q_j로의 간선의 라벨
  3. GTG가 오직 두 개의 상태 q_i(초기 상태)와 q_j(승인 상태)를 가짐
     - 연관된 정규 표현 : 식 $r = {r_{ii}}^* r_{ij}\left(r_{jj} + r_{ji}{r_{ii}}^* r_{ij}\right)^*$
  4. GTG가 3개의 상태 q_i(초기 상태), q_j(승인 상태) 와 q_k(제3의 상태)를 가짐
     - 다음과 같은 라벨을 갖는 새 간선들을 추가

$$r_{pq} + r_{pk}{r^*}_{kk} r_{kq}\,, \qquad p = i, j, \qquad q = i, j$$

     - 정점 q_k와 그에 연결된 간선들을 제거

정규 표현에 대한 정규 언어

- procedure : nfa-to-rex 계속

5. 만일 GTG가 오직 4개 이상의 상태를 가지고 있다면, 제거할 상태 q_k를 선택
   - 모든 상태들의 쌍 (q_i, q_j)에, $i \neq k, j \neq k$, 규칙 4를 적용
   - 각 단계에서, 가능한 경우 아래와 같은 단순화 하는 규칙을 적용

$$r + \emptyset = r$$

$$r\emptyset = \emptyset$$

$$\emptyset^* = \lambda$$

   - 정점 q_k와 그것과 연결된 간선들을 제거
6. 올바른 정규 표현을 얻을 때까지 단계 3에서 5까지 반복

우선형 문법과 좌선형 문법

- 정규 언어를 묘사하는 세 번째 방법
  - 간단한 문법을 사용하는 방법

[정의 3.3] 문법 $G = (V, T, S, P)$에서 모든 생성규칙들이 다음의 형태를 갖는 경우 : 우선형(right-linear) 문법

$$A \rightarrow xB$$

$$A \rightarrow x \quad (A, B \in V \text{이고 } x \in T^*)$$

또한 한 문법의 생성규칙들이 모두 다음의 형태를 갖는 경우 이 문법을 좌선형(leftlinear)문법이라 한다.

$$A \rightarrow Bx \text{ 또는 } A \rightarrow x$$

- 정규 문법(regular grammar) : 우선형 문법이거나 좌선형 문법

정규문법(Regular Grammar)

[정의] 문법 G가 우선형 문법(RLG, Right Linear Grammar)이거나 좌선형 문법(LLG, Left Linear Grammar)일 때 G : **정규문법**

- 예제 3.13 (103 page)

  $G_1 = (\{S\}, \{a, b\}, S, P_1)$

  $S \rightarrow abS \mid a$

(RLG ⇨ 정규문법)

- 문법 G_1에 대한 과정

$S \Longrightarrow a$

$S \Longrightarrow abS \Longrightarrow aba$

$S \Longrightarrow abS \Longrightarrow ababS \Longrightarrow ababa$

- $L(G_1)$은 정규표현 $r = (ab)^*a$ 으로 묘사

정규문법(Regular Grammar)

- 예제 3.13 (103 page) 계속

$G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$

(LLG ⇨ 정규문법)

$S \rightarrow S_1 ab$

$S_1 \rightarrow S_1 ab \mid S_2$

$S_2 \rightarrow a$

- 문법 G_2에 대한 하나의 유도 과정

$S \Longrightarrow S_1 ab \Longrightarrow S_2 ab \Longrightarrow aab$

$S \Longrightarrow S_1 ab \Longrightarrow S_1 abab \Longrightarrow S_2 abab \Longrightarrow aabab$

- $L(G_2)$: 정규표현 $r = aab(ab)^*$ 에 의해 묘사되는 언어

정규문법(Regular Grammar)

- 예제 3.14 (104 page)

  $G = (\{S, A, B\}, \{a, b\}, S, P)$

  $P : S \rightarrow A$

  $A \rightarrow aB \mid \lambda$

  $B \rightarrow Ab$

RLG도 LLG도 아님 ⇨ 정규문법 아님

- 선형문법(linear grammer)
  - 각 생성규칙의 우변에 하나 이하의 변수, 변수의 위치 제한 없음
- 모든 정규 문법 : 선형 문법, 모든 선형 문법 : 정규 문법이 되지는 않음

- 정규 문법 : 정규 언어에 대한 또 다른 표현 방법

우선형 문법의 정규 언어 생성

- 우선형 문법에 의해 생성되는 언어 : 정규 언어

- 우선형 문법의 유도 과정을 모방하는 nfa 구성
  - 우선형 문법의 문장형태(sentential form)
    - 하나의 변수만 존재, 변수가 가장 오른쪽에 있는 특별한 형태
  - 유도 과정에서 임의의 순간에 생성규칙 $D \rightarrow dE$를 사용, 다음의 유도를 진행하는 단계에 있다고 가정

$$db \cdots cD \implies ab \cdots cdE$$

    - 이 문법에 해당하는 nfa : 이 단계를 심벌 d를 읽을 경우 상태 D에서 상태 E로 이동하는 것으로 모방할 수 있음

우선형 문법의 정규 언어 생성

- [정리 3.3] (105 page) $G = (V, T, S, P)$가 우선형 문법이면, $L(G)$는 정규 언어

증명) $V = \{V_0, V_1, \dots\}, S = V_0$

$P : V_0 \Rightarrow v_1 V_i$

$\Rightarrow v_1 v_2 V_j$

$\overset{*}{\Rightarrow} v_1 v_2 \dots v_k V_n$

$\Longrightarrow v_1 v_2 \dots v_k v_l = w$

오토마타 초기 상태 : V_0

각 변수 V_i에 대응하는 비승인 상태를 생성하여 라벨 V_i 지정

$V_i \longrightarrow a_1 a_2 \cdots a_m V_j$ 에 대하여 오토마타에 상태 V_i로 부터 V_j로의 전이 추가

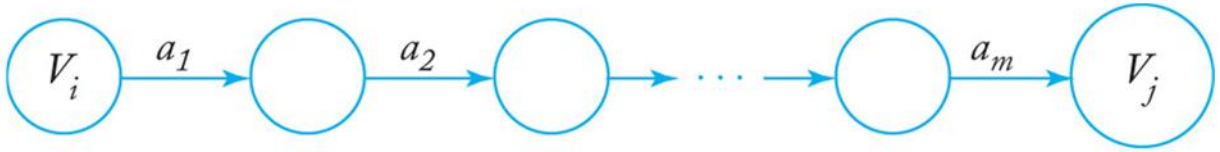
우선형 문법의 정규 언어 생성

- [정리 3.3] (105 page) 계속

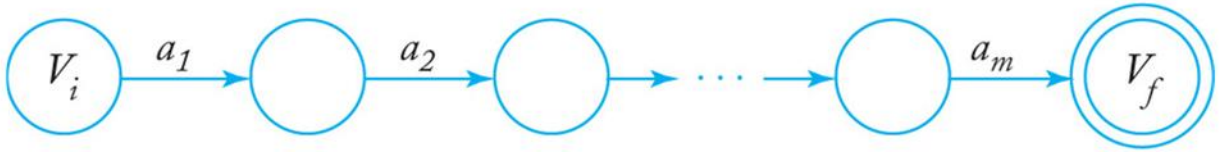
$V_i \longrightarrow a_1 a_2 \cdots a_m$ 에 대하여 다음의 전이 추가

$$\delta^*(V_i, a_1 a_2 \ldots a_m) = V_f$$

상태 V_f는 승인 상태



Represents $V_i \longrightarrow a_1 a_2 \ldots a_m V_j$



Represents $V_i \longrightarrow a_1 a_2 \ldots a_m$

우선형 문법의 정규 언어 생성

- [정리 3.3] (105 page) 계속

$w \in L(G)$이고 $v_1 v_2 \ldots v_k v_l = w$ 을 만족할 때

$V_f \in \delta^*(V_0, w)$

$\therefore w$는 M에 의해 승인됨

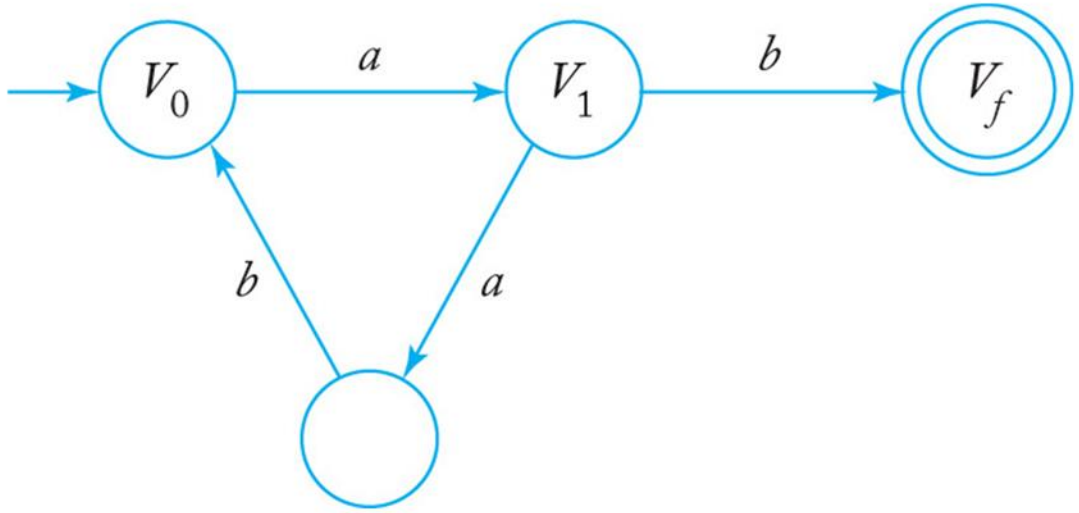
- w가 M에 의해 승인된다고 가정
  - w를 승인하기위해 오토마타는 상태 $V_0, V_i, \ldots$ 등을 거쳐 상태 V_f에 도달
  - $w = v_1 v_2 \ldots v_k v_l$
  - 유도과정 $V_0 \Rightarrow v_1 V_i \Longrightarrow v_1 v_2 V_j \overset{*}{\Rightarrow} v_1 v_2 \ldots v_k V_k \Longrightarrow v_1\ v_2 \ldots v_k v_l$

    $\therefore w$는 $L(G)$에 속함

우선형 문법의 정규 언어 생성

- 예제 3.15 (106 page) 문법에 의해 생성되는 언어 인식 : 유한 오토마타

$$V_0 \longrightarrow aV_1$$
$$V_1 \longrightarrow abV_0 \mid b$$



$$L\big((aab)^*ab\big)$$

정규 언어에 대한 우선형 문법

- 모든 정규 언어 : 우선형 문법에 의해 생성될 수 있음

증명)

- 주어진 언어의 dfa 구성
- 정리 3.3에서 보인 내용을 역으로 구성
- 구성된 dfa의 상태들 : 문법에서 변수
- 전이를 발생시키는 심벌들 : 단말 심벌들

- [정리 3.4] (107 page) L이 알파벳 Σ에 대한 정규 언어일 때,

L = L(G)를 만족하는 우선형 문법 G = (V, Σ, S, P)가 항상 존재한다.

정규 언어에 대한 우선형 문법

- 예제 3.16 (108 page) $L(aab^*a)$에 대한 우선형 문법을 구성

$\delta(q_0, a) = \{q_1\}$	$q_0 \longrightarrow aq_1$
$\delta(q_1, a) = \{q_2\}$	$q_1 \longrightarrow aq_2$
$\delta(q_2, b) = \{q_2\}$	$q_2 \longrightarrow bq_2$
$\delta(q_2, a) = \{q_f\}$	$q_2 \longrightarrow aq_f$
$q_f \in F$	$q_f \longrightarrow \lambda$

  - 문자열 $aaba$의 유도 과정

$$q_0 \Longrightarrow aq_1 \Longrightarrow aaq_2 \Longrightarrow aabq_2 \Longrightarrow aabaq_f \Longrightarrow aaba$$

정규 언어와 정규 문법간의 동치성

- [정리 3.5] (109 page) 언어 L이 정규언어이고 그럴 때에만

$L = L(G)$를 만족하는 좌선형 문법 G가 존재한다.

증명) 주어진 좌선형 문법 생성규칙 : $\mathrm{A} \to Bv$ 또는 $\mathrm{A} \to v$

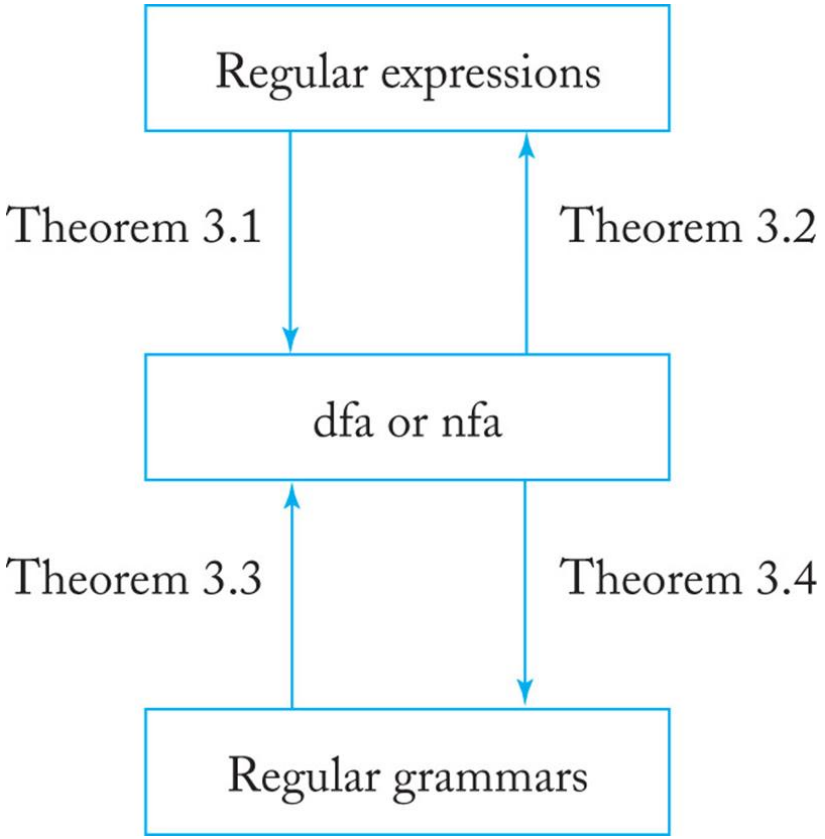
- 우선형문법 $\hat{G}$ 구성 : $\mathrm{A} \to v^R B$ 또는 $\mathrm{A} \to v^R$

$$L(G) = \left(L\left(\hat{G}\right)\right)^R$$

- [정리 3.6] (110 page) 언어 L이 정규언어이고 그럴 때에만 $L = L(G)$를 만족하는 정규 문법 G가 존재한다.

정규 언어와 정규 문법간의 동치성

- 정규 언어를 묘사하는 방법
  - dfa, nfa, 정규 표현, 정규 문법 등



정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

RE ⇨G_3

[알고리즘] *정규표현* ⇨ *정규문법*

① λ $S \rightarrow \lambda$

② $x \cdot y$ $A \rightarrow xB$ 또는 $A \rightarrow xy$
$B \rightarrow y$

③ x^*y $A \rightarrow xA \mid y$

④ yx^* $A \rightarrow Ax \mid y$

⑤ $x + y$ $A \rightarrow x \mid y$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex] *정규표현* ⇨ *정규문법*

① $a \cdot b$ ⇨ $S \to aA$, $A \to b$ or $S \to ab$

② $ac \cdot (a + b)$ ⇨ $S \to acA$, $A \to a + b$

$\therefore\ S \to acA$, $A \to a \mid b$ or $S \to aca \mid acb$

③ a^*b ⇨

$S \to aS \mid b$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex] *정규표현* ⇨ *정규문법*

④ a^* ⇨ $S \to aS \mid \lambda$ or $S \to Sa \mid \lambda$

⑤ $(a+b)^*$ ⇨ $S \to (a+b)S \mid \lambda$

$\therefore\ S \to aS \mid bS \mid \lambda$

⑥ $\mathrm{a}(a+b)^*$ ⇨ $S \to S(a+b) \mid a$

$\therefore\ S \to Sa \mid Sb \mid a$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex] 정규표현 $(ab)^+(a+d)$

$(ab)^+(a+d) = (ab)^*(ab)(a+d) = (ab)^*(aba+abd)$

$\therefore\ S \rightarrow abS \mid (aba+abd)$

$S \rightarrow abS \mid aba \mid abd$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex]

① 예제3.16 (108 page) 정규표현 aab^*a

$aab^*a = aa \cdot b^*a$

$S \rightarrow aa\mathrm{A}$

$A \rightarrow b^*a$

⇨

$S \rightarrow aaA$

$A \rightarrow bA \mid a$

①‘ $aab^*a = aab^* \cdot a$

$S \rightarrow Aa$

$A \rightarrow aab^*$

⇨

$S \rightarrow Aa$

$A \rightarrow Ab \mid aa$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex]

② $L = \{a^n | \, n \geq 0\} \cup \{b^n a | \, n \geq 1\}$, 정규표현 $a^* + b^+ a$

$S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow a^*$

$S_2 \rightarrow b^+ a$

⇨

$S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow aS_1 \mid \lambda$

$S_2 \rightarrow bS_2 \mid ba$

- $a^* + bc^* d$

$S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow a^*$

$S_2 \rightarrow bc^* d$

⇨

$S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow aS_1 \mid \lambda$

$S_2 \rightarrow bS_3 d$

$S_3 \rightarrow cS_3 \mid \lambda$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

- ***G_3 ⇨ RE*** [알고리즘]

G_3		정규표현식		정규표현
$S \to Sx \mid y$	⇨	$S = y + Sx$	⇨	$S = yx^*$
$S \to xS \mid y$		$S = xS + y$		$S = x^*y$

- $G: S \to xS \mid y$

$$L(G) = \{y, xy, xxy, \cdots, x^n y, \cdots\}$$

∴ 정규표현 $= x^*y$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex] $S \rightarrow S_1ab$ ⇨ $S = S_1ab \cdots\cdots ①$

$S_1 \rightarrow S_1ab \mid S_2$ ⇨ $S_1 = S_1ab + S_2 \cdots\cdots ②$

$S_2 \rightarrow a$ ⇨ $S_2 = a \cdots\cdots ③$

③ → ② $S_1 = S_1ab + a = a + S_1ab = a(ab)^* \cdots\cdots ④$

④ → ① $S = a(ab)^*ab = a(ab)^+$

[ex] $S \rightarrow aS \mid bR \mid \lambda$ ⇨ $S = aS + bR + \lambda$

$R \rightarrow aS$ ⇨ $R = aS$

$\therefore S = aS + baS + \lambda$

$= (a + ba)S + \lambda$

$= (a + ba)^*\lambda$

$= (a + ba)^*$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex] $S \rightarrow aA \mid bB \mid b$ ⇨ $S = aA + bB + b$ …… ①

$A \rightarrow bA \mid \lambda$ ⇨ $A = bA + \lambda$ …… ②

$B \rightarrow bS$ ⇨ $B = bS$ …… ③

from ② $A = b^*\lambda = b^*$ …… ④

③,④ → ① $S = ab^* + bbS + b$

$= bbS + (b + ab^*)$

$\therefore \; S = (bb)^*(ab^* + b)$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

- ***FA ⇨ G_3***

[알고리즘]

FA

Ⓐ →ᵃ Ⓑ

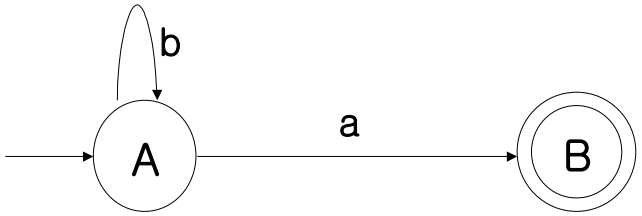
모든 최종상태 Ⓕ

G_3

A → aB

F → λ

[ex]

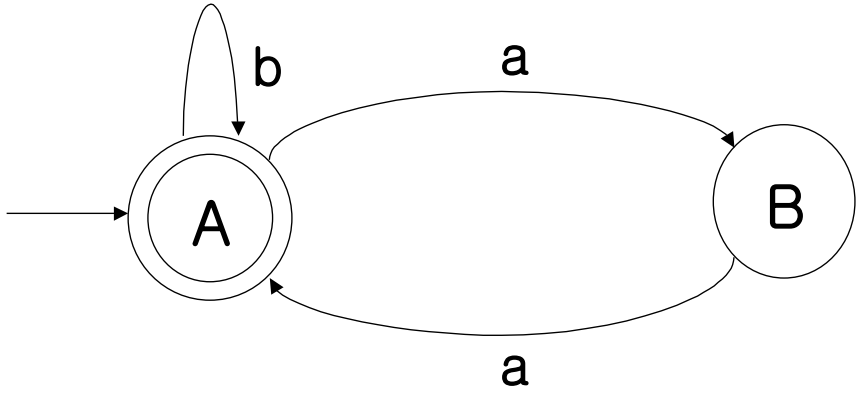


A → aB | bA

B → λ

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex]



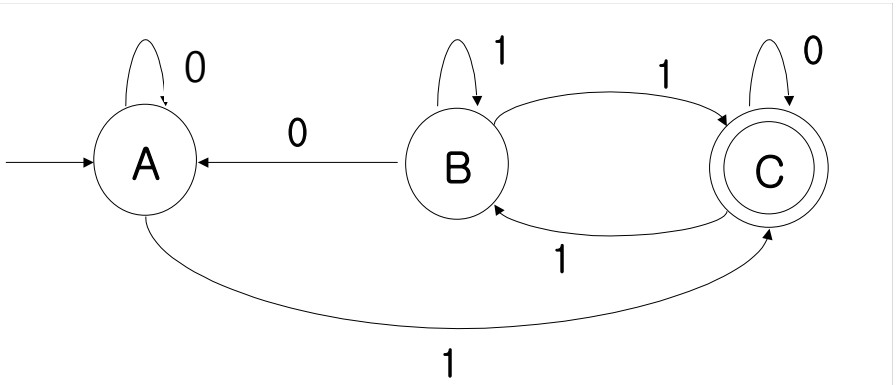
$A \rightarrow aB \mid bA$

$B \rightarrow aA$

$A \rightarrow \lambda$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex]



A → 0A | 1C

B → 0A | 1B | 1C

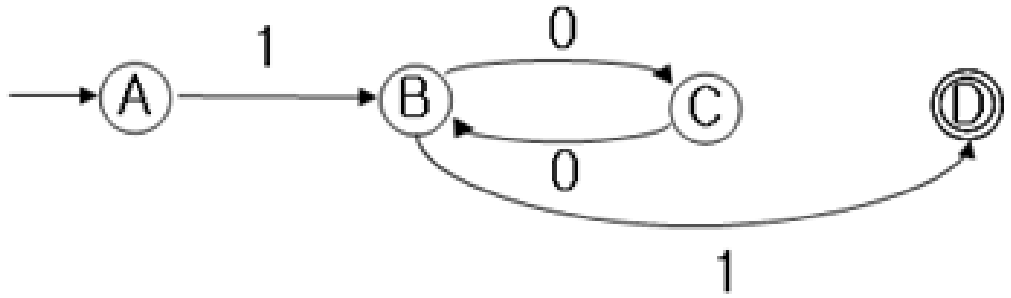
C → 0C | 1B

C → λ

C → 0C | 1B | λ

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex]



A → 1B

B → 0C | 1D

C → 0B

D → λ

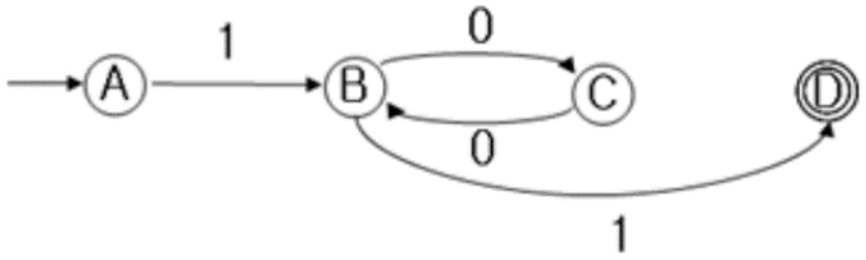
정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

- **G_3 ⇨ *FA*** [알고리즘]

G_3		*FA*
A → aB		Ⓐ →ᵃ Ⓑ
A → Ba	⇒	Ⓐ →ᵃ Ⓒ, C ∈ F
A → aC		
S → λ		S ∈ F

[ex]

A → 1B		A → B1
B → 0C | 1	또는	B → C0 | 1
C → 0B		C → B0

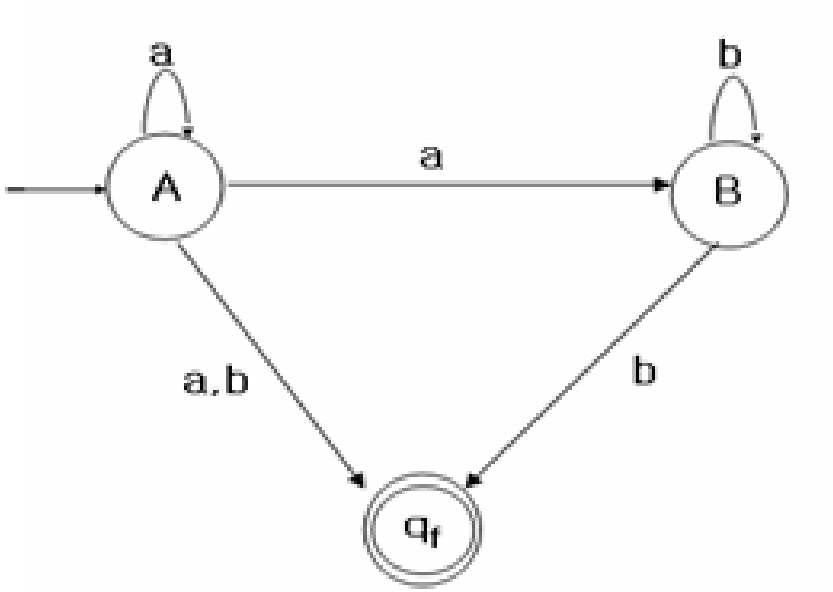


정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex]

A → aA | aB | a | b

B → bB | b



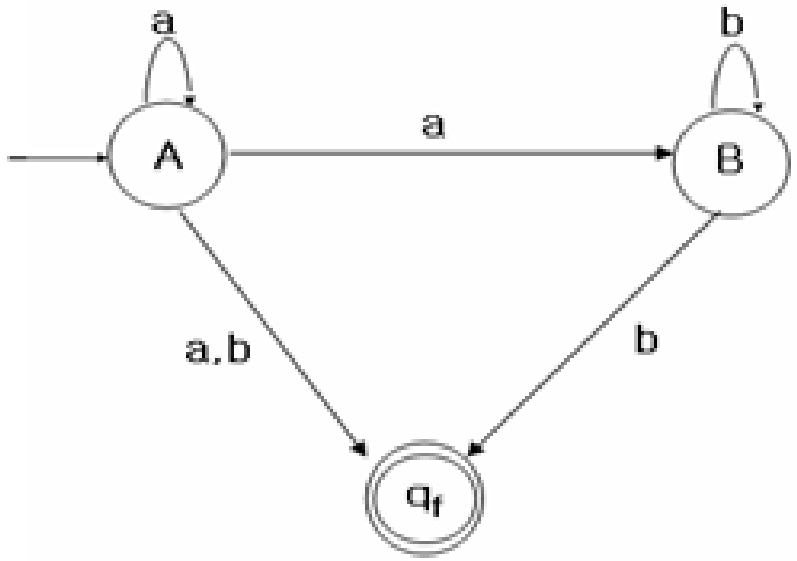
$A = a^* (ab^+ + a + b) = a^* (ab^*+b)$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex]

A → aA | aB | a | b

B → bB | b



$A = aA + aB + a + b$

$B = bB + b$

$A = aA + (ab^{+} + a + b)$

$B = b^{+}$

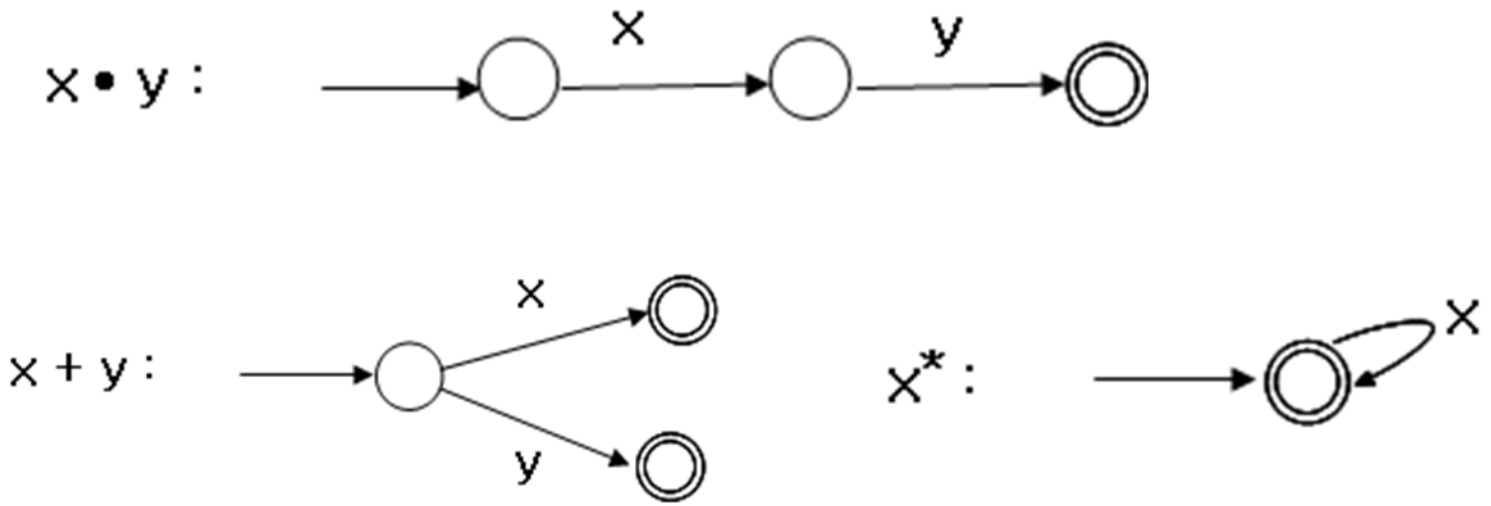
$A = a^{*}(ab^{+} + a + b) = a^{*}(ab^{*} + b)$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

- ***RE ⇨ FA***

[알고리즘] ***정규표현 ⇨ NFA ⇨ DFA ⇨ 최소화된 DFA***

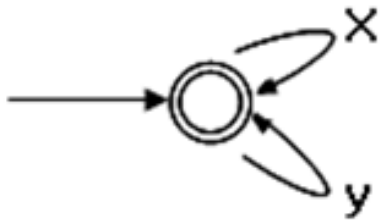
[정리] r이 정규표현일 때 정규언어 L(r)을 인식하는 NFA가 존재한다.



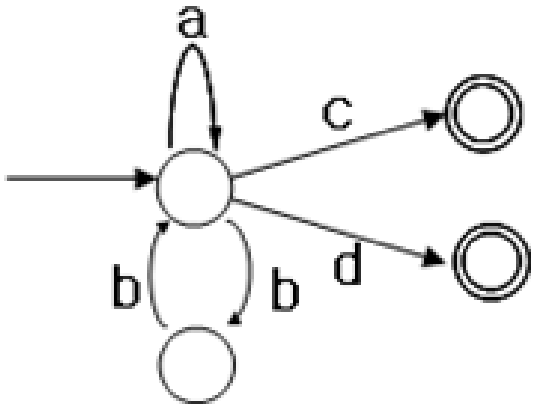
정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex] 정규표현으로 부터 NFA 만들기

$(x + y)^*$:

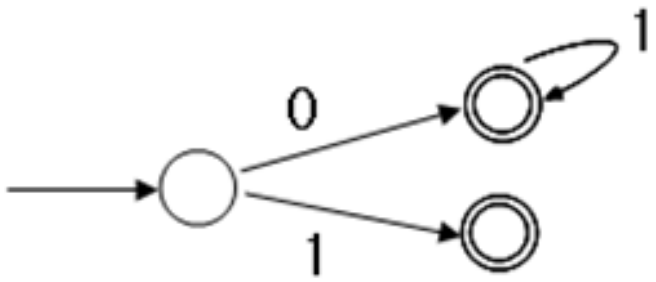


$(a+bb)^*(c+d)$:

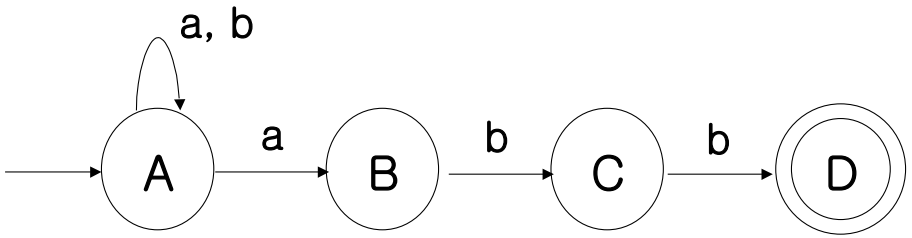


정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

(2) 01* + 1



(3) $(a+b)^*abb$



정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

- ***FA ⇨ RE***
  - ✔ 직접 변환은 매우 어려움.

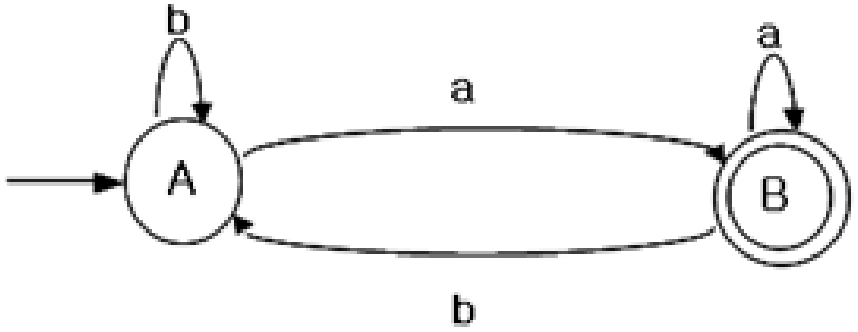
[알고리즘]

FA ⇨ G3 ⇨ 정규표현

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex]

(1)



G_3 : $A \rightarrow aB \mid bA$

$B \rightarrow aB \mid bA \mid \lambda$

정규표현식 : $A = bA + aB$

$B = bA + aB + \lambda = A + \lambda$

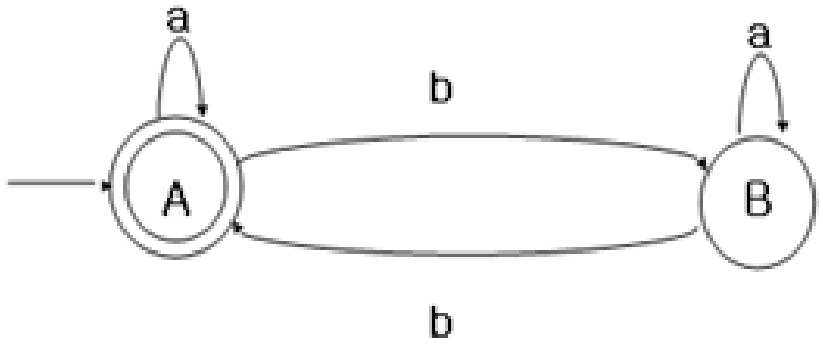
$\therefore A = bA + a(A + \lambda)$

$= (a+b)A + a$

$= (a+b)^*a$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

(2)



$G_3 : A \rightarrow aA \mid bB \mid \lambda$

$B \rightarrow aB \mid bA$

정규표현식 : $A = aA + bB + \lambda$

$B = aB + bA = a^*bA$

$\therefore A = aA + bB + \lambda$

$= aA + ba^*bA + \lambda$

$= (a+ba^*b)A + \lambda$

$= (a+ba^*b)^*$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[ex]

$G_3 : A \rightarrow 0A \mid 1B$ ⇨ $A = oA + 1B = o^*1B$ …… ①

$B \rightarrow 0A \mid 1C \mid \lambda$ ⇨ $B = oA + 1C + \lambda$ …… ②

$C \rightarrow 0C \mid 1B$ ⇨ $C = oC + 1B = o^*1B$ …… ③

① = ③ ∴ A = C

② $B = 0A + 1A + \lambda = (1+0)A + \lambda$ …… ④

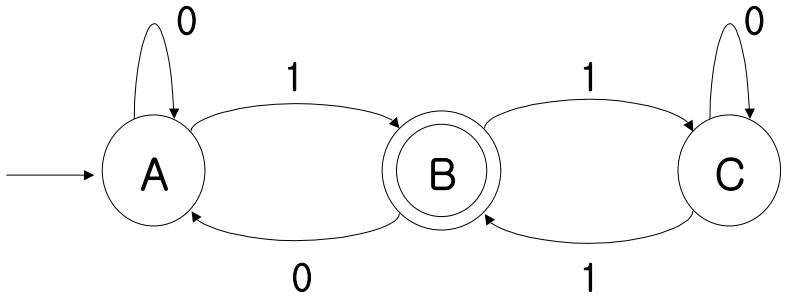
④ → ① $A = 0^*1\ (\ (1+0)A + \lambda\ )$

$= 0^*1\ (1+0)A + 0^*1$

$= (\ 0^*1\ (1+0)\ )^*\ 0^*1$

$= (\ 0^*\ (11+10)\ )^*\ 0^*\ 1$

$= (\ 0+11+10\ )^*\ 1$



$(r^*s)^*r^* = r^*(sr^*)^* = (r+s)^*$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

[다른 풀이법] DFA를 최소 상태의 DFA로 바꾸면

최소 DFA	0	1
초기 A	A	B
최종 B	A	A

정규표현식 $A = 0A + 1B$

$B = 0A + 1A + \lambda$

$\therefore A = 0A + 1(0A+1A+\lambda)$

$= 0A + 10A + 11A + 1$

$= (0+10+11)A + 1$

$= (0+10+11)^* 1$

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

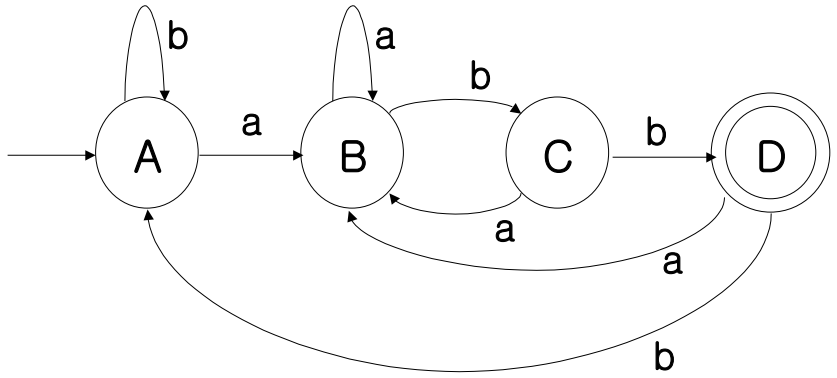
[ex]

$G_3 : A \rightarrow aB \mid bA$

$B \rightarrow aB \mid bC$

$C \rightarrow aB \mid bD$

$D \rightarrow aB \mid bA \mid \lambda$



정규표현식 : $A = bA + aB = b^*aB$ …… ①

$B = aB + bC = a^*bC$ …… ②

$C = aB + bD$ …… ③

$D = aB + bA + \lambda$ …… ④

정규문법(G_3), 정규표현(RE), 유한 오토마타(FA) 간의 변환

④ → ③ $C = aB + bD = aB + b(aB + bA + \lambda)$

$= aB + baB + bbA + b$

$= aB + baB + bbb^*aB + b$

$= (a + ba + bbb^*a)B + b = b^*aB + b$ …… ⑤

⑤ → ② $B = a^*b(b^*aB + b) = a^*bb^*aB + a^*bb$

$= (a^*bb^*a)^* a^*bb$ …… ⑥

⑥ → ① $A = b^*aB$

$= \underline{b^*a\ (a^*b.\ b^*a)^*}.\ a^*bb$

$= (b^*a.\ a^*b)^*\ b^*a.\ a^*bb$

$= \underline{(b^*a^*.\ ab)^*\ b^*a^*}\ abb$

$= b^*a^*\ (ab.\ b^*a^*)^*\ abb$

$= b^*.\ \underline{a^*\ (abb^*.\ a^*)^*}\ abb$

$= b^*\ (\underline{a+abb^*})^*\ abb$

$= \underline{b^*\ (ab^*)^*}\ abb$

$= (b+a)^*\ abb$

r(sr)* = (rs)*r

a.a* = a*.a

(rs)*r = r(sr)*

r*(sr*)* = (r*s)*r* = (r+s)*

r*(sr*)* = (r*s)*r* = (r+s)*

$A = bA + aB = b^*aB$ …… ①

$B = aB + bC = a^*bC$ …… ②

$C = aB + bD$ …… ③

$D = aB + bA + \lambda$ …… ④

정규언어와 정규문법

- [정리] 다음 문장들은 같은 의미다.
  - L은 정규집합(regular set)이다.
  - L은 좌선형 또는 우선형 언어(right/left-linear language(RLL/LLL))이다.
  - L은 유한 오토마타 언어(finite automata language)이다.
  - L은 NFA 언어(NFA language)이다.
  - L은 정규표현(regular expression)에 의해 나타낼 수 있다.