Chap. 3
Regular Languages and Regular Grammars

Finite Automata

DFA

NFA

Regular Language

정규 언어

Regular Grammer

정규 문법

Regular expression

Agenda of Chapter 3

- Regular Expressions
- Connection Between Regular Expressions and Regular Languages
- Regular Grammars
  - Right-Linear Grammar
  - Left-Linear Grammar

Definition of regular expressions

정확히 정규표현식.

- Regular expressions
  - One way of describing regular languages
  - Use notations involving strings, (,), +, ·, *
  - Ex) $(a+b\cdot c)^*$ ➔ $(\{a\} \cup \{bc\})^*$

concatenation.

[Formal definition] regular expressions

- Σ : a given alphabet language를 만드는 기본 alphabet.
- 1. Primitive regular expressions : ∅, λ, symbols in Σ empty string {λ}. {a, b, c}.
- 2. If r_1, r_2 are regular expressions
  then r_1+r_2, $r_1\cdot r_2$, r_1^*, (r_1) are regular expressions
- A string is regular expressions iff 아래와 같은 문장
  it can be derived from the primitive regular expressions
  by a finite number of applications of the rules in 2.

regular expression

regular expression. re re 5개 다 primitive.

ab++ +a·b a^n X 안됨

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

regular expression. re. re

Languages assoc. with regular expressions(1/2)

- Language L(r) denoted by regular expressions r
  1. $\emptyset$ denotes empty set $L(\emptyset) = \{\ \}$.
  2. λ denotes $\{\lambda\}$ $L(\lambda) = \{\lambda\}$.
  3. a denotes {a} $L(a) = \{a\}$.

  If r1, r2 are regular expressions then

  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. $L((r_1)) = L(r_1)$ → ( ) 우선 순위
  7. $L(r_1^*) = (L(r_1))^*$

- Ex) $L(a^* \cdot (a+b))=$ $\{a\}^* \{a,b\}$ = $\{a^n, a^m b \mid n \geq 1, m \geq 0\}$

  $\{a\}$ $\{b\}$

  $\{a\}^*$ $\{a,b\}$.

  이 둘의 concatenation.

- Precedence rule

  $( ) > * > \cdot > +$ → 연산자 우선 순위

Languages assoc. with regular expressions(2/2)

Ex3.3] $r=(a+b)^*(a+bb)$ ~ w

$L(r) = \{ wa, wbb \mid w \in \{a,b\}^* \}$.

$\{a,b\}^* \rightarrow$ 아무거나
$\{a,bb\}$.

$(aa)^+ = (aa)^* \cdot aa$.
굳이 만들 필요가 X

Ex3.4] $r= (aa)^* (bb)^*b$

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

Ex3.5] $\Sigma=\{0, 1\}$

"'00' 최소한 한개"

$L(r)=\{w\in \Sigma^* \mid$ w has at least one pair of consequtive zeros$\}$

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

Ex3.6] Give a regular expression r for the languages

10101110

$L(r)=\{w\in\{0, 1\}^* \mid$ w has no pair of consecutive zeros$\}$

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

적어도 00은 포함.

0만 붙이게 되면, 0으로 끝나는 애가 X 즉 $(0+\lambda)$은 붙여야함

"연속한 00 이 없어야 한다."

1 01 이 2 묶어 지면 된다.

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

- Equivalence of two regular expressions (r_1 & r_2)
  - r_1 & r_2 (denote the same language)

Reg. Expressions denote reg. languages(1/3)

[THEOREM 3.1]

Let r be a regular expression.
Then there exist an nfa accepting L(r).
Consequently, L(r) is a regular language.

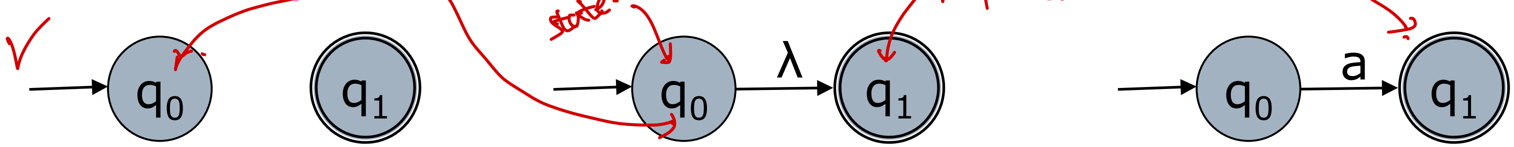
L(r)을 accept 하는 dfa | nfa 찾으면 된다!

Reg Exp → NFA → dfa.
Regular Language.
양쪽으로 증명

→ 존재한다 regular language.

Proof)

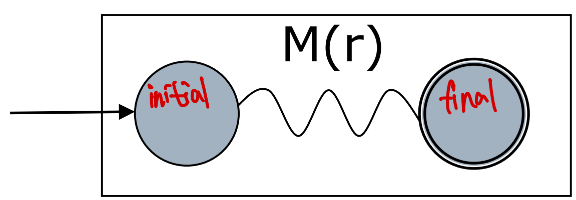
1. begin with automata accepting the languages for the simple expressions ∅, λ, a



2. Assume that we have M(r_1) and M(r_2) accepting languages denoted by r_1 and r_2.

Automata.

M(r_1+r_2).



Reg. Expressions denote reg. languages(2/3)

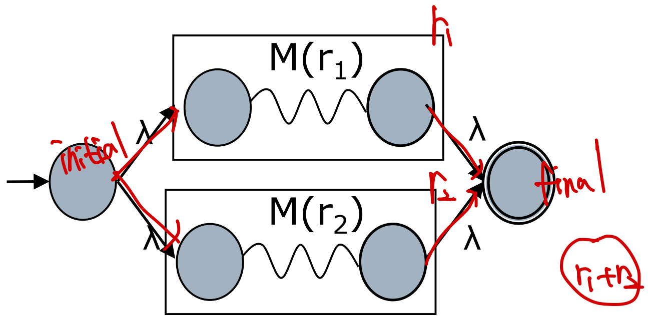
Proof continued)

$L(r_1) \cup L(r_2)$

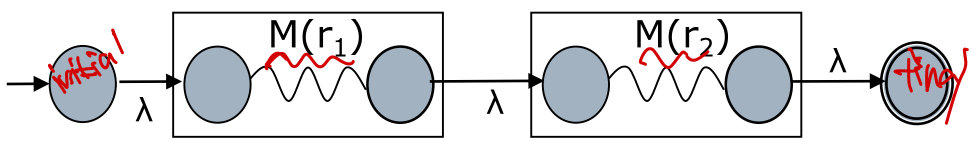
3. Construct automata for $r_1 + r_2$, r_1r_2 , r_1^*

$r_1 \cdot r_2$

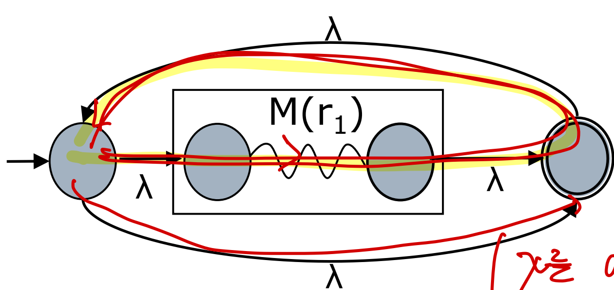
~~만들어 붙임~~



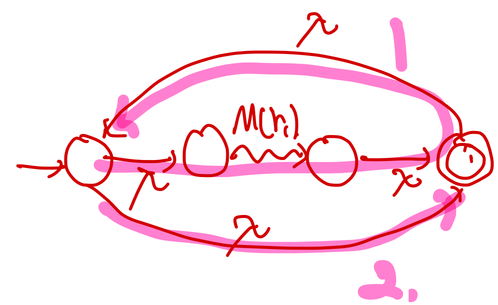
→ 자동적으로 확장하기 좋게.



$r_1 \cdot r_2$



x를 accept 할 수 있게 해야함

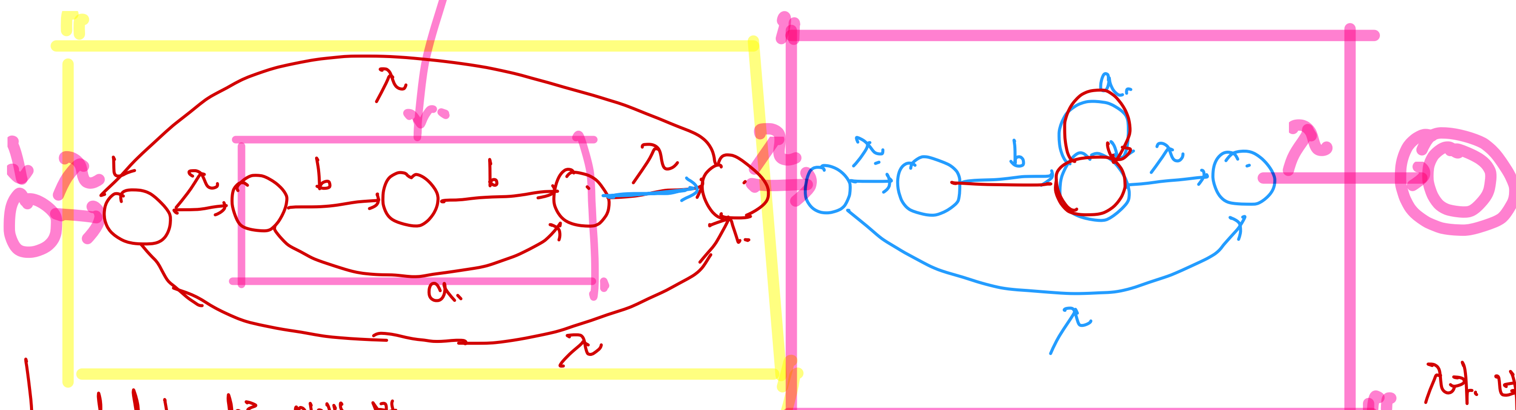


Reg. Expressions denote reg. languages(3/3)

Ex 3.7] nfa accepting L(r)

– $r = (a+bb)^*(ba^*+\lambda)$ → Regular expression.

baa.
b
ba



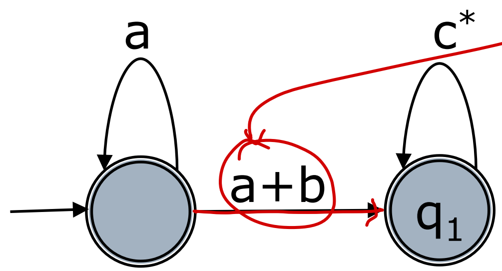
↳ redundant node를 없애는 법.
⇒ 교과서 3장 마지막.

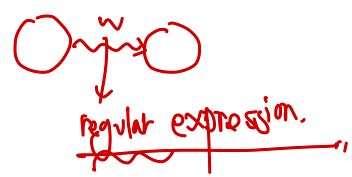
λ가 너무 많다.

Reg. Expressions for reg. languages(1/3)

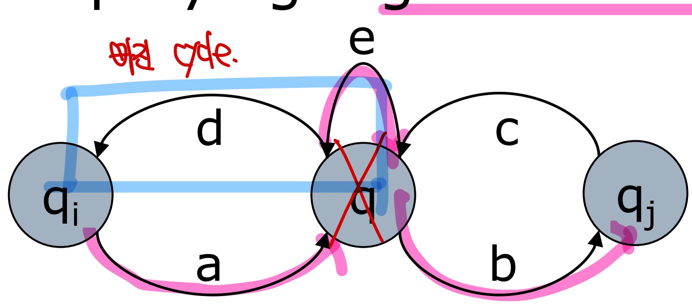
- Goal
  - Find a reg. expressions corresponding to a reg. language
- Generalized transition graph
  - Edges are labeled with "regular expressions"

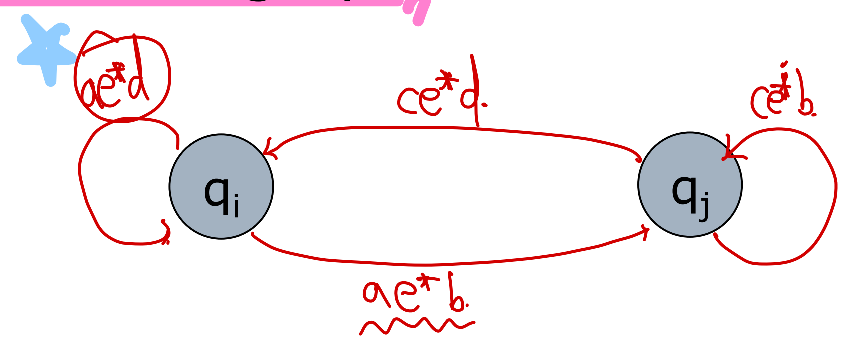
Ex3.8]





- Simplifying a "generalized transition graph"





Reg. Expressions for reg. languages(2/3)

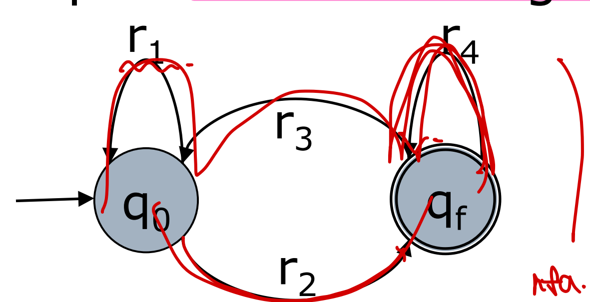
[Theorem 3.2] L: regular language

There exists a regular expressions r such that L=L(r)

Proof) Let M be an nfa accepting L

(M has only one final state (which is not the initial state))

1. Interpret M to a generalized transition graph
2. Applying the simplifying techniques until reaching the situation



3. Then the regular expression is

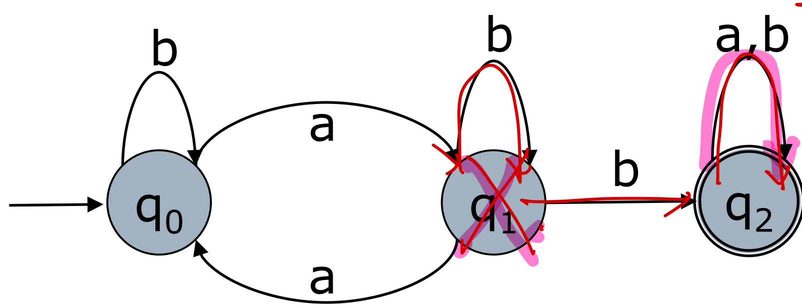
$$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$$

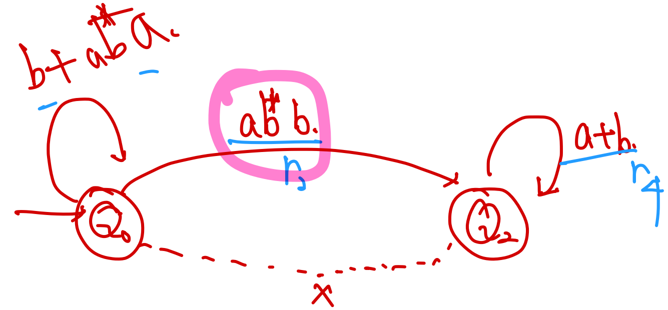
$r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$

(r_4^*) 있어도 되고 없어도 된다

Reg. Expressions for Reg. Languages (3/3)

Ex3.9] nfa to regular expressions





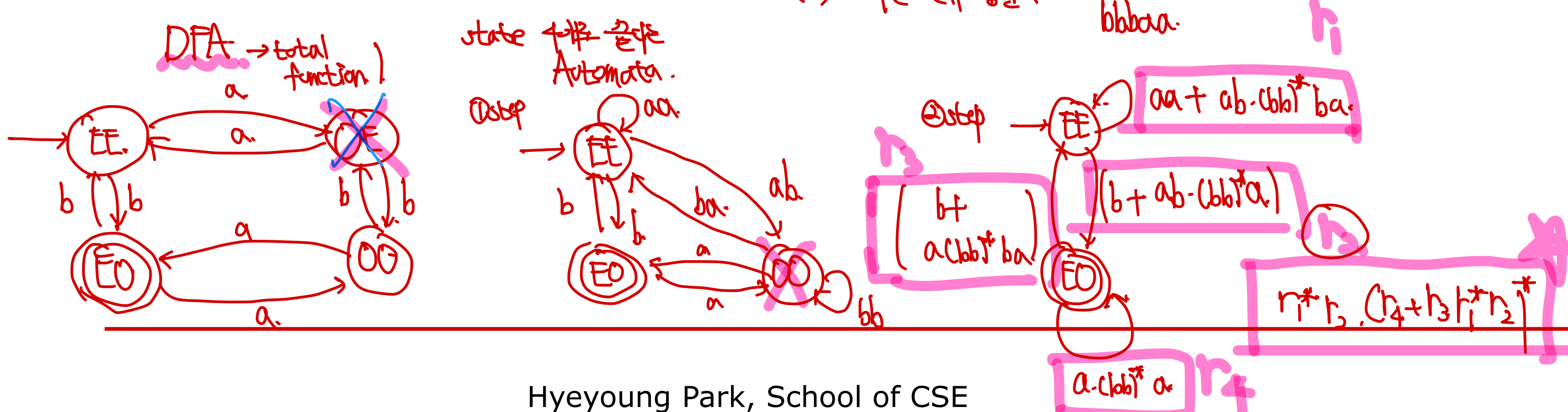
Ex3.10] Find a regular expressions for

L={w {a,b}*| n_a(w) is even, n_b(w) is odd}

a의 개수는 짝수 , b의 개수는 홀수 개

→ 어떤 순서 상관x

babab
bbbaa



Variable Terminal Start: product

Right- and Left-Linear Grammars (1/2)

- A grammar G=(V,T,S,P) is **right-linear**
  - all products are of the form

    가장 오른쪽에 variable 딱 1개!

    A → xB | x

    변수 1개 변수 1개

    where A, B ∈ V, and x ∈ T*

    terminal로만 이루어진 string
- A grammar G=(V,T,S,P) is **left-linear**

  변수가 1개있는 선형식 (1차식)

  f(x) = ax+b

  - all products are of the form

    A → Bx | x

    A BB(x) B

    where A, B ∈ V, and x ∈ T*

  혼합 X
- A regular grammar is either right-linear or left-linear

  변수 1개
  - left side of production
    - at most one variable
  - right side of production
    - Variables must consistently be either the rightmost or leftmost symbol

$A \to xB \mid x.$ | $A, B \in V$, and $x \in T^*$.

$G = (V, T, S, P)$

right-linear.

$G = (V, T, S, P)$

Right- and Left-Linear Grammars (2/2)

Ex3.12]

$S \to aS$ $S \Rightarrow aS \Rightarrow aaS \Rightarrow \cdots \Rightarrow a^n S$

- Right-linear grammar $G_1=(\{S\},\{a,b\},S,P_1)$

  $S \to abS \mid a$ → right linear grammar, regular grammar.

  $S \Rightarrow abS \Rightarrow ababS \Rightarrow \cdots \Rightarrow (ab)^n S \Rightarrow (ab)^n a$

  r= $(a \cdot b)^* a.$

- Left-linear grammar $G_2=(\{S,S_1,S_2\},\{a,b\},S,P_2)$ → left linear grammar.

  $S \to S_1 ab$, $S_1 \to S_1 ab \mid S_2$, $S_2 \to a$

  $S \Rightarrow S_1 ab \Rightarrow S_1 abab \Rightarrow \cdots \Rightarrow S_1 (ab)^n \Rightarrow S_2 (ab)^n \Rightarrow a(ab)^n.$

  r= $a \cdot (ab)^* ab.$ 무조건 ab가 포함되어야 함.

  $a\,ab(ab)^*.$

Ex3.13] A linear grammar G =({S,A,B},{a,b},S,P) regular grammar (X) linear grammar (O)

$S \to A$, $A \to aB \mid \lambda$, $B \to Ab$

right left

$S \Rightarrow A \Rightarrow \lambda$

$S \Rightarrow A \Rightarrow aB \Rightarrow aAb \Rightarrow ab$

$aAb \Rightarrow aaBb \to aaAbb \Rightarrow aabb.$

$aaaBbb \stackrel{*}{\Rightarrow} a^3 b^3.$

RL G. LL R. > Reg Grammar ⟺ Reg Lang (FA) Theorem 3.4 Theorem 3.3 Theorem 3.5 4개 다 증명해야 함.

Right-linear grammars generates Reg. Lang.(1/2)

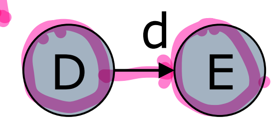
Theorem 3.3] Grammar → FA (transition graph)

오토마타를 찾아서!

If G=(V,T,S,P) is a right-linear, then L(G) is a regular language.

Note) When **ab…cD ⇒ ab…cdE** is arrived by using **D→dE** d가 붙었으니까!

In nfa, there is an edge labeled with d from D to E.



- A state of nfa: variable in sentential form
- String already processed: terminal prefix of sentential form

Proof) For production rules of G of the forms

기본 형태.

$V_i \rightarrow a_1 a_2 \ldots a_m V_j$ and $V_i \rightarrow a_1 a_2 \ldots a_m$

we have corresponding transition graphs:

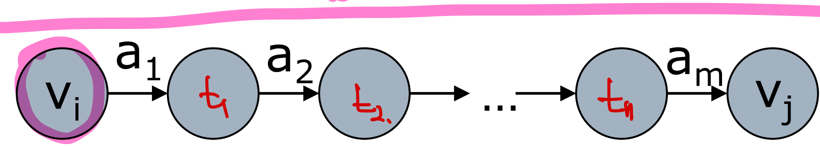
G^R 기본 조건

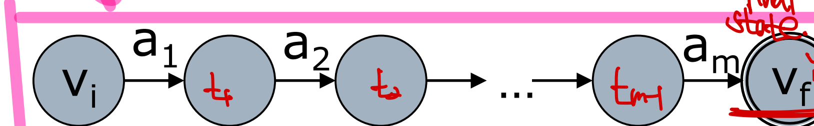
D→dE

FA

D →d E

derivation의 "끝"





G와 M이 대응.

우리가 보여야할 것. G가 있다면. L(G)는 regular language 이다.
→ show (automata)

Right-linear grammars generates Reg. Lang.(2/2)

Proof Continued)

G^R이 있을때 $L(G^R) \Rightarrow$ FA가 있어서 accept.

이 transition으로 정의된 Automata.

i) For an arbitrary $w \in L(G)$, we have

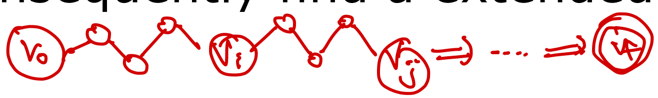
$V_0 \Rightarrow v_1V_i \Rightarrow v_1v_2V_j \Rightarrow v_1v_2...v_kV_n \Rightarrow v_1v_2...v_kv_l = w$

Since we have a transition graph for each derivation,

we can consequently find a extended transition $V_f \in \delta^*(V_0, w)$.

accept 가능

final state



ii) For an arbitrary $w \in L(M)$,

이 automata가 accept 하는 건 전부 G로 derivation 되는 아이들이다.

there exist a sequence $V_0, V_i, ..., V_f$ with edges labeled as $v_1, v_2 ... v_l$.

Thus, w must have the form $w = v_1v_2...v_kv_l$, and

the derivation $V_0 \Rightarrow v_1V_i \Rightarrow v_1v_2V_j \Rightarrow v_1v_2...v_kV_n \Rightarrow v_1v_2...v_kv_l$ is possible.

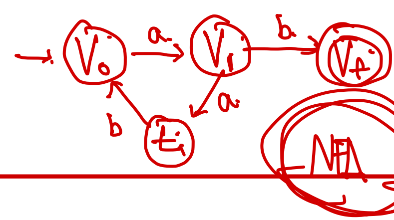
$\therefore w \in L(G)$.

Automata 안에 해당하는 production이 없을 수 밖에 없다.

이 w는 L(G)로 만든

Ex3.14] Construct a FA accepting the language generated by

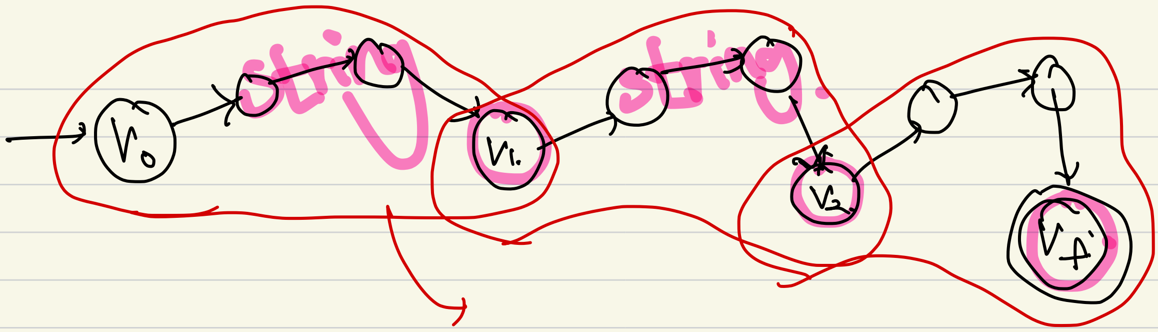
$V_0 \rightarrow aV_1, \quad V_1 \rightarrow abV_0 \mid b$



RL Grammar → Regular Language 로 보인 것.

NFA

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



이 한 뭉텅이 들을.
production 으로 하자 이 방법.

RL → DFA를 만들 수 있음 → Regular 이다.

Right-linear grammars for Reg. Lang.(1/2)

- DFA ➔ grammar
  - States of dfa : variables in grammar
  - Symbols causing transitions : terminals in productions

DFA Automata

Theorem 3.4] DFA → grammar

If L is a regular language on Σ then we have a right linear grammar $G=(V,\Sigma,S,P)$ such that $L=L(G)$

<u>Construction of a right-linear grammar G</u>

From definition,

we have a dfa $M=(Q, \Sigma, \delta, q_0, F)$ accepting L, where

$Q=\{q_0,q_1,\ldots,q_n\}$, $\Sigma=\{a_1,a_2,\ldots a_m\}$, $\delta(q_i,a_j)=q_k$.

We can construct a grammar $G=(V,\Sigma,S,P)$ with $V=\{q_0,q_1,\ldots,q_n\}$,

For $\delta(q_i,a_j)=q_k$ ➔ $q_i \rightarrow a_j q_k$ terminal!

variable

"끝"나고 variable이 사라질 경우

If q_k is in F ➔ $q_k \rightarrow \lambda$

final state에 해당하는 variable 만 없앨 수 있다.

if. Q_k is final

$Q_k \rightarrow \lambda$

Right-linear grammars for Reg. Lang.(2/2)

Proof of L(G)=L(M))

(1) Show that all $w= a_ia_j...a_ka_l$ in L(M) can be generated by G.

From the fact that we have transitions,

$\delta(q_0,a_i)=q_p$, $\delta(q_p,a_j)=q_r$, … , $\delta(q_s,a_k)=q_t$, $\delta(q_t,a_l)=q_f \in F$. transition

We can make the corresponding derivation

$q_0 \Rightarrow a_iq_p \Rightarrow a_ia_jq_r \Rightarrow a_ia_j...a_kq_t \Rightarrow a_ia_j...a_ka_lq_f \Rightarrow a_ia_j...a_ka_l$ derivation

Therefore, $w \in L(G)$. production ⟷ transition

(2) For all $w \in L(G)$, we have the same derivation, which directly implies $\delta^*(q_0, a_ia_j...a_ka_l)=q_f$.

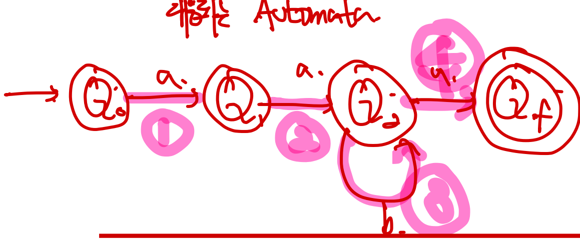
$\delta^*(Q_0, a_i, a_j ... a_k, a_l) = Q_f \in F$

Therefore, w is accepted by the corresponding dfa.

$G=(\{Q_0, Q_1, Q_2, Q_f\}, \{a,b\}, Q_0, P)$

☐ Ex3.15) Construct a right-linear grammar for $L(aab^*a)$ Regular Language

accept 하는 Automata



1. $Q_0 \to aQ_1$
2. $Q_1 \to aQ_2$
3. $Q_2 \to bQ_2 \mid aQ_f$
4.
5. $Q_f \to \lambda$

$G^{RL} \Longrightarrow L(G^{RL})$ is Regular. T3.3
$\Longleftarrow$ T3.4

Equivalence between Reg. Lang. & Reg. Grammars

Theorem 3.5] Theorem 3.3, 3.4를 활용한 설명.

L is regular iff there exist a left-linear grammar G such that L=L(G)

[Propositions] 가정하는 사실 (참인 명제)

① We have a G^{LL} generating L ⇔ we can find a G^{RL} generating L^R

② L is regular ⇔ L^R is regular

L이 Regular 면 우리는 G^{LL}을 찾을 수 있다.

G^{LL} 이 있으면 $L(G)$는 regular.

[Proof of Theorem 3.5]

L is regular ⇒ We have a G^{LL}	We have a G^{LL} ⇒ L is regular
↳② L^R is regular. Theorem 3.4 ⇒ $\exists G^{RL}$ generating L^R. ① ⇒ $\exists G^{LL}$ generating L.	↳① G^{RL} generating L^R. ⇒ Theorem 3.3 L^R is Regular. ② ⇒ L is Regular.

Equivalence between Reg. Lang. & Reg. Grammars

[Proof of Propositions ①] $\forall G^{LL}$ generating L, $\exists$ G^{RL} generating L^R

G^{LL}의 P: $A \to Bv$, $A \to v$ $(A, B \in V,\ v \in T^*)$.

G^{RL}의 P': $A \to v^R B$, $A \to v^R$ $\quad L(G^{RL}) = L^R$.

ex) $L = \{(ab)^n \mid n \geq 1\}$.

G^{LL}: $S \to Sab \mid ab$

G^{RL}: $S \to baS \mid ba$ $\quad L(G^{RL}) = \{(ba)^n \mid n \geq 1\} = L^R$

[Proof of Propositions ②] L is regular $\Leftrightarrow$ L^R is regular

수정예정.

Equivalence between Reg. Lang. & Reg. Grammars

Theorem 3.6]
L is regular iff there exist a regular grammar G such that L=L(G)

Regular expressions

Theorem 3.1 Theorem 3.2

Reg Langs ⟷ dfa and nfa

Theorem 3.3 Theorem 3.4

Regular grammars

[Right Linear]
[Left Linear]

L is regular.

Regular expression
dfa or nfa.
Regular grammar
셋중 1개만 찾으면 됨

L → Regular 인지
찾는 문제 가능!

Hyeyoung Park, School of CSE