1. Regular Languages

1.1 Finite Automata

0. computer → 꽤나 복잡

idealized computer의 computational model 분석

⇒ the simplest model (finite state machine) 으로 시작!

1. Finite automata : limited amount of memory 갖는 모델

확률 X (↔ markov chains : 확률 O , pattern 인식)

the heart of electromechanical devices

(EX) automatic door

↳ OPEN / CLOSE의 two states로 동작

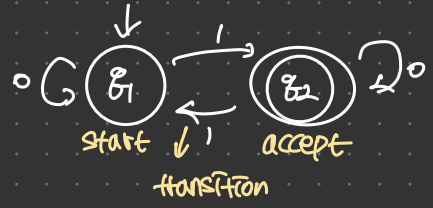
⇒ single bit of memory

⇒ 기계들의 기초 구성!

2. Finite Automaton (state diagram)

[start / accept state , state
transitions] > string accept / reject.

double circle



3. Formal Definition of a Finite Automata (informally)

A finite automation is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta : Q \times \Sigma \to Q$ is the **transition function**, ⇒all path
   ↳ combination
4. $q_0 \in Q$ is the **start state (initial state)**, and
5. $F \subseteq Q$ is the set of **accept states (goal states)**. ⇒ 종점할 수들

i) accept state ≥ 1

ii) δ : input symbol에 대해 다음 state가 정해져 있음

(Ex) $M_1 = (Q, \Sigma, \delta, q_1, F)$

$Q = \{q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

q_1 is start state

$F = \{q_2\}$

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

(A) : the set of all strings that M accepts.

↳ Language of machine , $L(M) = A$

= M recognizes A

= M accepts A

4. Formal Definition of Computation (formally)

⇒ 앞전 def.들 mathematically 하게 formalize

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

$\omega = \omega_1 \omega_2 \cdots \omega_n$

$\Sigma \in \omega_i$

① $r_0 = q_0$

② $\delta(r_i, \omega_{i+1}) = r_{i+1}$ for $i = 0 \cdots n-1$

③ $r_n \in F$

$L(M) = A \in \omega$

$Q \in \{r_0, r_1 \cdots r_n\}$

↳ A는 Regular Language

↳ finite automaton이 recognize하는 Ⓛ.

5. Designing Finite Automata → creative process..

⇒ only contain crucial information 필요!

i) R.L 파악

ii) state 설정 → possibility 파악할 것

iii) transition 가능성 파악

iv) accept, start 설정

6. Regular Operations

operation이라는 것은 operation을 반복하도 regular language여야 함.

: regular languages의 operations!

⇒ finite automata의 사칙연산

In math, numbers / operations

In theory of computation, languages → operations

① UNION : $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$ ⇒ 합집합

② Concatenation : $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$ ⇒ 교집합

↳ strcat과 유사
$x = ab$
$y = cd$
$x \cdot y = abcd$

③ Star : $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and } x_i \in A\}$ ⇒ n승

↳ n승

↳ empty string ε

⇒ regular language is closed.

6-1. Union Operation

pf. Idea) A_1, A_2 ⇒ regular language

A_1 recognize M_1, A_2 recognize M_2

$A_1 \cup A_2$ recognize M이라고 할 것 ⇒ M_1과 M_2의 accept state 모두 accept 즉, simulating할 것이다.

그렇다면 순차적으로 accept? No. 동시에 (simultaneously) ...

pf) $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

$Q = \{(r_1, r_2) \mid r_1 \in Q \text{ and } r_2 \in Q_2\}$

$= Q_1 \times Q_2$

$q_0 = (q_1, q_2)$

Σ은 M_1과 M_2가 같음. → 같은 걸로 해야 의미 O

$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$

$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

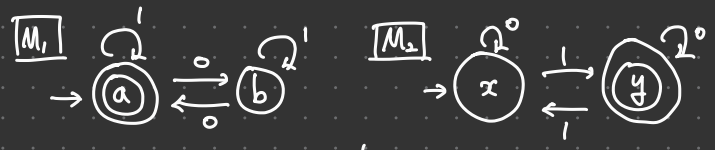
(Q, a)

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

M을 $A_1 \cup A_2$가 recognize.

Ex) A_1 : 0이 짝수

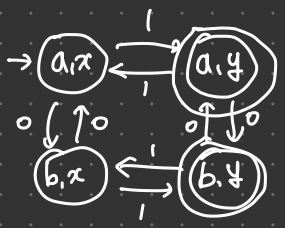
A_2 : 1이 홀수



i) state 생각

ii) transition ← Updating both!

iii) accept ← 둘 중에 하나라도 accept



6.2 Concatenation?

→ M은 input이 어디서 멈춰야 하는지 모른다.

⇒ nondeterminism으로 해결.

1.2 Nondeterminism

0. DFA는 하나의 symbol을 읽으면 다음 state는 정확히 하나로 확정됨.
   ⇒ deterministic, deterministic finite automata (DFA)
   ↔ several choices가 하나 아니라 0개 이상 존재 ★
   ⇒ nondeterministic, nondeterministic finite automata (NFA)

1. NFA [transition이 여러 개 있을 수 있다. / ε이 가능] ↔ DFA
   ⇒ 동시에 가능한 경로 모두 따라감. → 여러 가지 경로 생성
   ⇒ 가능한 경로 중 하나라도 accept state에 도달한다면 그 string은 accept string
   ⇒ 독립적인 processes들이나 threads들이 can be running concurrently.
   ↳ intuitive하게 !!
   ⇒ NFA가 더 간단할 때도 있음.

2. ε transition의 장점?
   ⇒ understand를 쉽게 할 수 있도록 diagram 생성 가능

3. Formal Definition.
   i) DFA↔NFA의 essential한 차이점 : the type of transition function.
   ↳ the set of possible next states
   ii) $P(Q)$: the collection of all subsets of Q (states)
   ↳ power set of Q
   iii) $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ ★ 입실론을 포함.
   iv) $\delta : Q \times \Sigma_\varepsilon \to P(Q)$

Definition 1.37

A **nondeterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$.

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

↳ 정함

iv) NFA가 accepts하는 w

$w = y_1 y_2 \cdots y_m$, $y_i \in \Sigma_\varepsilon$

$r_0 r_1 \cdots r_m \rightarrow r_i \in Q \ (i = 0 \sim m)$

① $r_0 = q_0$ start state

② $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0 \cdots m-1$ and

③ $r_m \in F$

DFA는 같아야 했지만 NFA는 집합 형태이기에 속하기만 하면 된다.

accept state들 중 하나이기만 하면 된다.

4. Equivalance of NFAs and DFAs

i) DFA와 NFA는 같은 language를 recognize할 수 있다.

↳ NFA가 ε을 포함하기에 더 많은 language를 recognize할 것이라고 생각함.

ii) NFA를 표현하는 것이 DFA보다 간단할 때도 있음.

"equivalent" : syntactically equal

↳ transition func은 다르지만 recognize하는 language가 같음.

5. Proof (NFA recognizes R.L)

i) proof Idea

① equivalance of NFA and DFA

② the set of states : DFA

↳ 2^k subsets of states : NFA

③ start, accept state ?

ii) Proof (일단, ε 없이 먼저 증명)

$N = (Q, \Sigma, \delta, q_0, F)$ recognizes some language A

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

① $Q' = P(Q)$

② $R \in Q'$, $a \in \Sigma$

$\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a), r \in R\}$

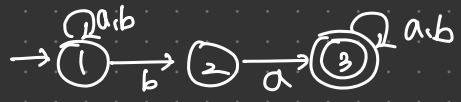
$= \bigcup_{r \in R} \delta(r, a)$

↳ UNION

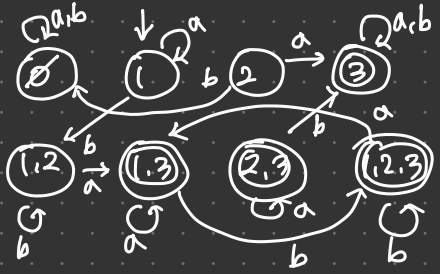
③ $q_0' = \{q_0\}$

④ $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$

EX



DFA 그래프 (Q)



NFA $2^{\text{상태}}(Q')$

① ✓
② ✓
③ ✓
④ ✓

> Proof!

iii) Proof (to consider the ε arrows)

① R : NFA에서 state들의 집합 ⇒ E(R) : R의 ε arrow 추가

② Formally, $R \subseteq Q$

let $E(R) = \{q \mid q \text{ can be reached from } R \text{ by traveling along 0 or more } \varepsilon \text{ arrows}\}$

③ $\delta(r, a) \Rightarrow E(\delta(r, a))$

ex). ε로 갈 수 있는 state가 q_2, q_3라면 ↳ 추가해줄 것.

$E(\{q_1, q_2\}) = \{q_1, q_2, q_3\}$

④ the start state도 확장 필요

⇒ $q_0' = E(\{q_0\})$

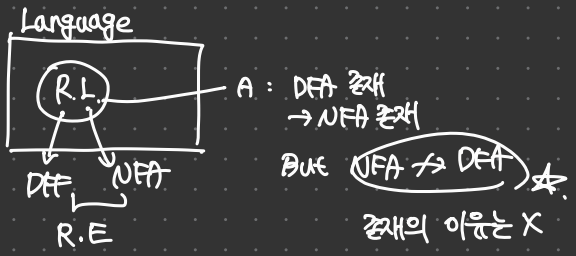
6. Collary 1.40

: A language is regular ⟷ some NFA recognizes it

⇒ NFA와 DFA는 서로 convert 가능.

pf) ① → : cuz a R.L has a DFA recognizing it
and any DFA is also an NFA.

② ← : If an NFA recognizes some languages,
so does some DFA, and hence the language is R.L



7. Converting an NFA to a DFA

: 순서 항상 잘 지킬것. 순서가 다르다고 해서 틀린 것은 아니지만
증명을 이용하여 converting 할 경우, 증명부터 바꿔야 함.

① state → P(Q) 찾을 것 , ∅ 빼먹지 X

② start, accept 설정.
└ $q_0' = E(\{q_0\})$ ★

③ transition 결정

④ E 써줘서 설정 (맨 마지막에 써줄것!!)

+) minimal로 변경도 가능하나
굳이 default하게 다르지는 X

8. Closure under the Regular Operation

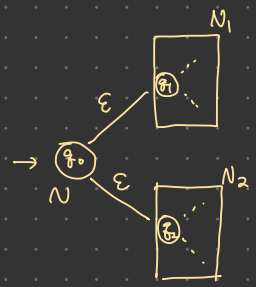
↳ operation 증명을 NFA로 대신!

i) UNION : The class of R.L is closed under the union operation.

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ " A_2

$N = (Q, \Sigma, \delta, q_0, F)$ " $A_1 \cup A_2$



① $Q = \{q_0\} \cup Q_1 \cup Q_2$
↳ start state를 새로 생성

② q_0 → start state of N

③ $F = F_1 \cup F_2$ → 변경 X

④ $q \in Q$, $a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \\ \delta_2(q,a) & \text{if } q \in Q_2 \\ \{q_1, q_2\} & \text{if } q = q_0 \text{ and } a = \varepsilon \\ \emptyset & \text{if } q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

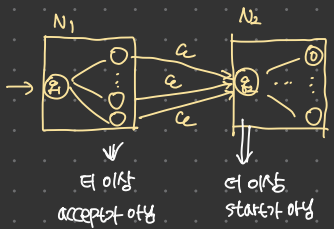
↳ 다른 alphabet accept 하지 않음!!

ii) Concatenation : The class of R.L is closed under the concatenation operation

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ " A_2

$N = (Q, \Sigma, \delta, q_1, F_2)$ " $A_1 \circ A_2$



① $Q = Q_1 \cup Q_2$

② q_1, 즉 N_1의 start state

③ F_2, 즉 N_2의 accept state

④ $q \in Q$, $a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & \text{if } q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & \text{if } q \in Q_2 \end{cases}$$

↳ 새로운 ε transition 유지할 것

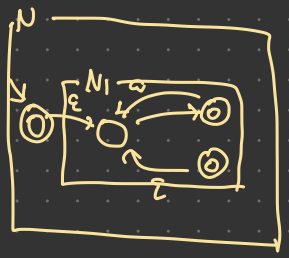
iii) star

① idea : start state → accept? ↳ additional ε arrow !!

→ 기존에 accept 하지 않았던 것까지 accept. ⇒ 새로운 start state

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \rightarrow A_1$

$N = (Q, \Sigma, \delta, q_0, F) \rightarrow A_1^*$



① $Q = \{q_0\} \cup Q_1$ ↳ new start state

② q_0 ⇒ start state

③ $F = \{q_0\} \cup F_1$ ⇒ old accept + new start

④ $q \in Q \quad a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \text{ and } q \notin F_1 \rightarrow \text{accept X} \\ \delta_1(q,a) & \text{if } q \in F_1 \text{ and } a \neq \varepsilon \rightarrow \text{accept, } \varepsilon \text{X} \\ \delta_1(q,a) \cup \{q_1\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \rightarrow \text{" } \varepsilon \text{O} \\ \{q_1\} & \text{if } q = q_0 \text{ and } a = \varepsilon \rightarrow \text{new start} \\ \emptyset & \text{if } q = q_0 \text{ and } a \neq \varepsilon \rightarrow \text{포함되지 않는 경우에!} \end{cases}$$

↳ 추가

1.3 Regular Expressions

: the regular operations to build up expressions describing languages.

(ex) $(0 \cup 1)0^* \Rightarrow (\{0\} \cup \{1\})\{0\}^*$ ⇒ language + operation
↳ Language

의미: starting 0 or 1 followed by any number of 0s.

i) ∘ : 생략 가능

ii) important in computer science applications.

iii) generally, Σ is any alphabet. ⇒ 길이 n다양.

Σ^0 : 길이가 0

$\Sigma \circ \Sigma = \Sigma^2$: 길이가 2

1. Formal Definition of a Regular Expression

① inductive definition

length = 1
- i) a for some a in the alphabet Σ → $\{a\}$
- ii) ε → $\{\varepsilon\}$ → empty string
- iii) $\emptyset$ → empty language (doesn't contain any strings)

length = n
- iv) $(R_1 \cup R_2)$
- v) $(R_1 \circ R_2)$
- vi) (R_1^*)

} 할수록 길이가 ↑

② 괄호가 명백할 때는 생략 가능

③ 계산 순서: star → concatenation → union

④ R^+ : 한 번 이상이라는 뜻 → $R^+ = RR^*$

⑤ L(R): language of R $\quad R^+ \cup \varepsilon = R^*$

2. Example of regular expression.

assume the alphabet Σ is $\{0,1\}$

① 0^*10^* → 0…010…0

② $\Sigma^*1\Sigma^*$ → 1을 substring

③ $\Sigma^*001\Sigma^*$ → 001을 substring

④ $1^*(01^+)^*$ → empty str 혹은 1…101…1

⑤ $(\Sigma\Sigma)^*$ → 짝수 length

⑥ $(\Sigma\Sigma\Sigma)^*$ → 3의 배수 length

⑦ $01 \cup 10 = \{01, 10\}$

⑧ $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$ = 시작과 끝이 같은 alphabet.

⑨ $(0 \cup \varepsilon)1^* = 0^*1^* = (0 1 \cdots 1) \cup 1^*$ (0이 없어도 됨)

⑩ $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$

⑪ $1^*\emptyset = \emptyset$

⑫ $\emptyset^* = \{\varepsilon\}$

3. Identity
   - i) $R \cup \emptyset = R$
   - ii) $R \circ \varepsilon = R$
   - iii) $R \cup \varepsilon \neq R$
   - iv) $R \circ \emptyset \neq R$

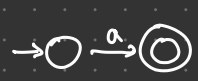
4. A Language is regular ⟺ some R.E describes it.

① ← 증명

Pf Idea) R.E가 해당 R.L을 describing하는 R.E가 있다고 가정.
R.E를 NFA로 converting → 그 NFA를 recognize하는 R.L 찾기!

Proof) by induction을 이용.

i) $R = a$ for some a in Σ → $L(R) = \{a\}$

→ NFA: 

$N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$

$\delta(q_1, a) = \{q_2\}$, $\delta(r, b) = \emptyset$ ($r \neq q_1$ 또는 $b \neq a$)

ii) $R = \varepsilon$ → $L(R) = \{\varepsilon\}$

→ NFA: 

$N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$

$\delta(r, b) = \emptyset$ for any r and b

iii) $R = \emptyset$ → $L(R) = \emptyset$

→ NFA: 

$N = (\{q_1\}, \Sigma, \delta, q_1, \emptyset)$

$\delta(r, b) = \emptyset$ for any r and b

iv) $R = R_1 \cup R_2$

v) $R = R_1 \circ R_2$

vi) $R = R_1^*$

> Closured under the operation.
> → 증명 완료!

② → 증명

Pf Idea) a Language is regular → R.L is described by a R.E

↳ DFA → GNFA → R.E

generalized NFA

* GNFA : transition arrow에 alphabet 하나가 아니라 R.E 형태로 표현된 것.

GNFA 방법이 여러 가지일 수 있음.

i) Proof를 위한 GNFA 2건

① start state : 다른 모든 state로 가는 arrow 존재해야 함.

But, start state로 오는 arrow는 존재하면 X

② accept state : 하나만 있어야 함.

start state와 달라야 함.

다른 모든 state에서 오는 arrow 존재해야 함.

But, accept state에서 가는 arrow는 존재 X

③ 다른 state : 각 state에 자기 자신을 포함하여 state 전체 개수만큼 transition 존재

ii) convert : DFA → GNFA

① add a new start state with an ε arrow to the old.

② add a new accept state "

③ multiple label → union으로 변경

④ add arrows labeled ∅

↳ Language가 달라지지 않도록 삽입.

ㅠ) convert : GNFA → R.E

① GNFA : k개의 states , k는 최소 3이상 (k>2)

⇒ construct an equivalent GNFA with k-1 states.

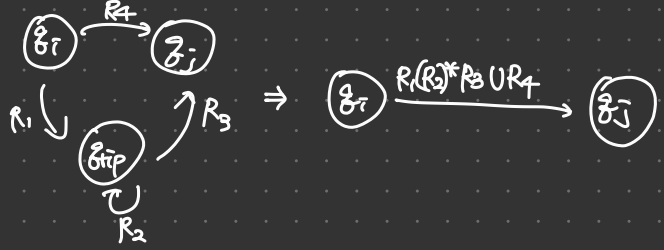
by reduced to states.

⇒ k=2, (start) ⟶ (accept) 형태가 될 것임.

즉, single arrow ★.

② constructing an equivalent GNFA with one fewer state when k>2

: after we removing some state we repair the machine by altering the regular expressions that label each of the remaining arrows.



③ k=2가 될 때까지 arrow를 유지하면서 state 지우기.

④ 마지막에 남은 single arrow의 label이 R.E ★.

주의 ; 지우는 순서가 달라도 결과적으로 나오는 R.E는 syntaxically equal 해야 함!

1.4 Nonregular Languages ↔ decidal question

: regular가 아니다. 즉, finite automata로 recognize X

ex) $B = \{0^n 1^n \mid n \geq 0\}$ → 0의 개수와 1의 개수가 같은 것.
↳ 0의 개수가 infinite하기에 finite automata로 recognize 불가함.
↳ if-else로 표현 가능

$C = \{w \mid w$ has an equal number of 0s and 1s$\}$

⇒ B. $C \notin$ R.L을 어떻게 증명?

1. The Pumping Lemma for R.L

: all regular language have a special property ⇒ Pumping lemma

① Theorem

0) Pumping lemma 만족 X ⇒ not regular.

The property states that all strings in the language can be pumped if they are at least as long as a certain special value.
↳ pumping length

⇒ each string contains a section that can be repeated any number of times with the resulting string remaining in the language. *
*: 없어서 없어도 pumped되어도 여전히 그 language.

1) A : R.L
P : pumping length
s : any string in A of length at least (P) → 2보다 큰 수
= xyz
i) for each $i \geq 0$, $xy^i z \in A$ → concat
ii) $|y| > 0 \rightarrow y \neq \varepsilon$, $y^0 = \varepsilon$
★ iii) $|xy| \leq p$

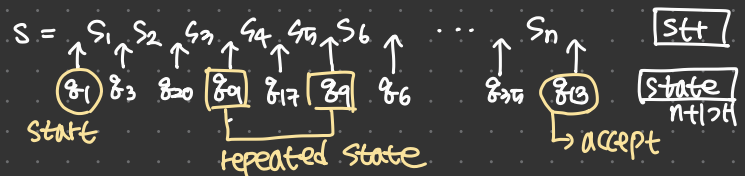
$s = xyz \in A$
$s' = xyyz \in A$

Pf Idea) Let. $M = (Q, \Sigma, \delta, q_1, F)$ recognizes A
p = the number of states of M.

Show. any string s in A of length at least p
⇒ $s = xyz$일 때 세가지 조건 모두 만족하지 않음을 보일것.

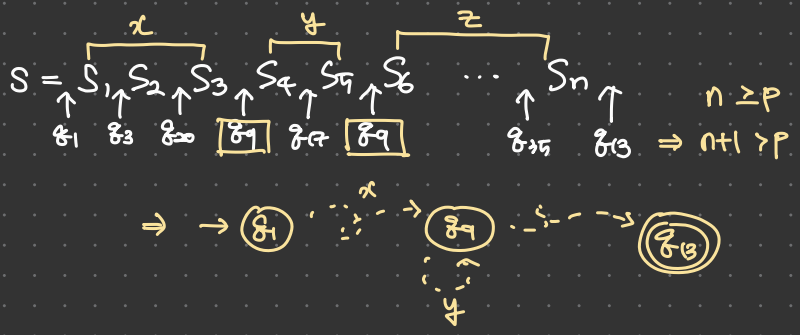
Why? : (P) → 조건이 이미 참인지 상관X
false true인걸 이용 (vacuously)
⇒ 현재 조건에서의 length가 p보다 큼

Assume. if s in A has length at least p,
M의 sequence of state를 고려해보자.
⇒ $s \in A$, $|s| \geq p$, $n \geq p$

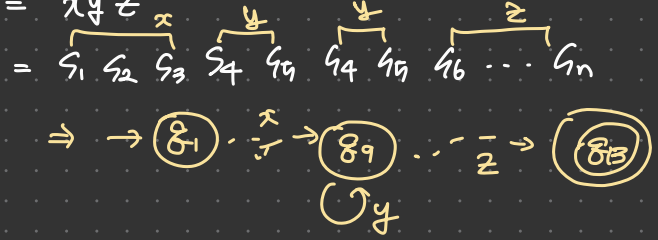


⇒ repeated > 1
(∵ n이 p보다 같거나 state가 적으면 n+1개가 n이하로)
⇒ pigeonhole principle.

Condition 1 for each $i \geq 0$, $xy^iz \in A$



$S' = xy^2z$



⇒ accept!!

Condition 2 입력 길이가 0이면 repeated state X

⇒ $(n \geq p)$ && $(n+1 > p)$ 만족 X

∵ S에서도 q_9의 다른 두 가지의 arrow 존재

Condition 3 p+1이 되는 순간에는 무조건 repeated state가 있어야 함.

|xy| 가 무한정 길어지면 repeated 존재 X

∴ $|xy| \leq p$

2. Pf for nonregular languages.

① contradiction을 이용하여 증명

i) R.L로 가정

ii) pumping length 설정

↳ pumped 되지 않는 적어도 p의 길이를 가진 string 찾기

iii) s를 x, y, z로 나눌 수 있는 모든 방법을 고려 → pumped X

$(\forall) s \in A$, $(\forall) |s| \geq p$ ⇒ $(\exists) s \in A$ $(\exists) |s| \geq p$

$(\exists)\ xyz$ ⇒ $(\forall)\ s \in A$

AND (① ② ③) → R.L

OR [①' ②' ③'] → 증명! → non R.L

Ex 1.73 $B = \{ 0^n 1^n \mid n \geq 0 \}$

Assume. ① B is regular.

② P = the pumping length given by the pumping lemma.

③ P가 존재함

⇒ $S = 0^p 1^p = \underbrace{0000 \cdots}_{x}\ \underbrace{00\ 11}_{y}\ \underbrace{\cdots 111}_{z}$

i) y가 0만 가지고 있을 경우,

$s' = xy^2z = xyyz = \underbrace{0000 \cdots 00}_{xyy}\underbrace{11 \cdots 11}_{z}$

⇒ 0의 개수는 늘어나고 1의 개수는 그대로이기에

$xyyz \notin B$ 이기에 case① 위반 → assume false

ii) y가 1만 가지고 있을 경우,

$s' = xyyz = \underbrace{000\cdots00}_{x}\underbrace{111}_{y}\underbrace{111}_{y}\underbrace{\cdots1}_{z}$

⇒ 0의 개수는 그대로고 1의 개수는 늘어남.

$xyyz \notin B$ 이기에 Case① 위반 ⟩ assume false

⇒ $|xy| > p$ 가 되어 Case③ 위반

iii) y가 0과 1을 가지고 있을 경우

$s' = xyyz = \underbrace{00\cdots}_{x}\underbrace{00011\cdots11}_{y}\underbrace{000111\cdots111}_{y}\underbrace{1111\cdots1}_{z}$

⇒ 0과 1의 개수는 같더라도 순서가 달라짐

$xyyz \notin B$ Case① 위반 ⟩ assume false

⇒ $|xy| > p$ 가 되어 Case③ 위반

Ex1.74 $C = \{w \mid w$ has an equal number of 0s and 1s$\}$

Assume. ① C is the regular.

② p = the pumping length given by the pumping lemma.

③ $s = 0^p1^p = \underbrace{00\cdots00}_{p}\underbrace{11\cdots11}_{p}$, $s' = xyyz = 0^{2p}1^{2p}$

x와 z의 길이가 0이고 y가 0^p1^p 라면 xy^2z도 pumped 가능

즉, any $i \geq 0$, $xy^iz \in C$

⇒ string을 어떻게 고르더라도 다르게 pumping lemma 성립 여부 다름.

반례 하나만 찾으면 그 Language는 nonregular !!

③' $s = 0^p1^p =$ $\overbrace{000\cdots00}^{p}\overbrace{111\cdots11}^{p}$

x y z

$s' = xyyz =$ $\underbrace{000}_{x}\underbrace{\cdots00}_{yy}\underbrace{111\cdots11}_{z}$

$|xy| > p$ → condition 3 만족 X

③'' 만약 $s = (01)^p$ 라면 $x = \epsilon$, $y = 01$, $z = (01)^{p-1}$ ⇒ $xy^iz \in C$

Ex 1.75 $F = \{ww \mid w \in \{0,1\}^*\}$

Assume. ① F is regular

② p : the pumping length given by the pumping lemma.

③ $s = 0^p10^p1 =$ $\overbrace{00\cdots000}^{p}\overbrace{1\ 00\cdots000}^{p}1$

x y z

$|xy| \le p$, $|y| > 0$ → y는 0으로만 이루어져 있음.

$s' = xyyz =$ $\underbrace{00\cdots000}_{x}\underbrace{\cdots}_{yy}\underbrace{00\cdots000}_{z}1$

p보다 수 p

⇒ $|xy| > p$가 되어 $xyyz \notin F$ case ③ 위반

③ 만약 $s = 0^p0^p$ 라면 $s' = xyyz \in F$ 로 pumped 0

→ regular

Ex 1.76 $D = \{1^{n^2} \mid n \geq 0\}$

Assume. ① D is regular

② p : the pumping length given by the pumping lemma.

③ $s = 1^{p^2} = \underbrace{111\cdots111}_{p^2}$

$|xy| = p^2$
$|xy| \leq p$ → $|xy^2z| \leq p^2 + p = |xyz| + |y|$
$|y| \leq p$ → $< p^2 + 2p + 1 = (p+1)^2$
$|xy^2z| < (p+1)^2$

$\Rightarrow p^2 < |xy^2z| < (p+1)^2$

↳ 정수의 제곱이 될 수 없음.

$\Rightarrow xy^2z \notin D$ → Condition ① 위반

Ex. 1.77 $E = \{0^i 1^j \mid i > j\}$

Assume. ① E is regular

② p : the pumping length given by the pumping lemma.

③ pumping up proof ⇒ y를 늘려서 증명하는 것.
pumping down ⇒ y를 줄여서 증명, 즉. xyz → xy^0z 증명! (xz)

④ $s = 0^{p+1}1^p = \underbrace{000\cdots00}_{x\,y}\underbrace{01\cdots11}_{z}$ (p, p)

y가 0만 consist (by. Condition ③ : $|xy| \leq p$)

$\Rightarrow s' = xy^0z = \underbrace{00\cdots00}_{x}\underbrace{1\cdots11}_{z}$ ⇒ 0개수 $\leq p$, 1개수 : p개) $i > j$