Pumping Lemma → necessary but not sufficient
→ can be used to prove language is not regular

① Adversary says he has a finite automaton, it has n no. of states

② You choose a string $w \ni |w| \geq n$

③ Adversary will break $w = xyz$, such that $|xy| \leq n$
$y \neq \varepsilon$

④ you should choose some q such that $xy^q z \notin L$

Prove that $a^n a b^n$ is not regular.

Let n be a constant

$w = xyz = a^n a b^n$

$= a^{n-i} a^i a b^n \qquad |xy| \leq n$

if $q = 2 \qquad xy^2 z = a^{n-i} a^{2i} a b^n$

$= a^{n+i} a b^n \notin L$ as no. of a's > no. of b

because $i \neq 0$ as $|y| > 0$

$n+i \neq n$

Therefore L is not regular!

Pumping theorem for CFL (Sec 3.5)

Let $G = (V, \Sigma, R, S)$ be a Context-free Grammar.

Then $w \in L(G)$ of length greater than $\emptyset(G)^{|V-\Sigma|}$ can be written as $w=uvxyz$ in such a way that v and y are non empty and uv^nxy^nz is in $L(G)$ for every $n \geq 0$

Then it is CFL otherwise not.

For every CFL A $\exists$ $k>0$ $\ni$ every $z \in A$ of length at least k can be broken into 5 pieces $z = uvwxy$ such that $vx \neq \varepsilon$ & $|uvx| \leq k$ and now $\forall$ $i \geq 0$, $uv^iwx^iy \in A$

Contrapositive holds

$L = \{a^n b^n a^n \mid n \geq 0\}$ prove that its not a CFL

① demon picks k
② you pick $z = a^k b^k a^k$, $|z| = 3k > k$
③ demon break $z = uvwxy$, $vx \neq \varepsilon$ $|vwx| \leq k$

① either v or x contain one 'a' & atleast one b

$v \leftarrow$ $a^{k-i} a^i b^{k-j} b^j a^k$

$q=2$ uv^2wx^2y : $a^n b^{k-j} a^r b^q b^k$

which is not of the form $a^* b^* a^*$

② v contains only a's $\quad uv^2wx^2y = \underbrace{a \cdots\cdots a}_{\#a\text{'s} > \#b\text{'s}} b \cdots b a \cdots a$

$\notin L$

There are more cases

③ v contains only b's

DPDA

$S \to aAS \mid bS \mid c$

$A \to d \mid \varepsilon$

$\text{first}(S) = \{a, b, c\}$

$\text{first}(A) = \{d, \varepsilon\}$

$\text{follow}(S) = \{\$\}$

$\text{Follow}(A) = \{a, b, c\} = \text{First}(S)$

$(p, e, e) \quad (q, S)$

$(q, a, e) \quad (q_a, e)$

$(q, b, e) \quad (q_b, e)$

$(q, c, e) \quad (q_c, e)$

$(q, d, e) \quad (q_d, e)$

$(q, \$, e) \quad (q_\$, e)$

$(q_a, e, S) \quad (q_a, aAS)$

$(q_d, e, A) \quad (q_a, d)$

$\left.\begin{array}{ll} (q_a, e, A) & (q_a, a) \\ (q_b, e, A) & (q_b, e) \\ (q_c, e, A) & (q_c, e) \end{array}\right\}$ follows(A)

$(q_a, e, a) \quad (q_a, e)$

$(q_b, e, b) \quad (q_b, e)$

$(q_c, e, c) \quad (q_c, e)$

$(q_d, e, d) \quad (q_d, e)$

Bottom up parser

(1) $((p,a,e),(p,a))$ → push a for each $a \in \Sigma$

(2) $((p,e,\alpha^R),(p,A))$ reverse for each rule $A \to \alpha$ in R set of rules

(3) $((p,e,S),(q,e))$

(Everything opposite)

Rule-1:

((p,a,e),(p,a)),
((p, b, e),(p, b)),
((p,c,e),(p,c)),
((p, d, e),(p, d)),

Rule-2:

((p,e,SAa), (p, S)),
((p,e,Sb), (p,S)),
((p,e,c), (p,S)),
((p,e,d), (p,A)),
((p,e,e), (p,A)),

Rule-3:

((p,e,S), (q,e))

To accept 'ac'

State	Input	stack
p	ac	e
p	c	a
p	c	aA
p	e	aAc
p	e	aAS
p	e	S
q	e	e