CONTEXT-FREE (LANG/GRAMMARS) = P.D.A.

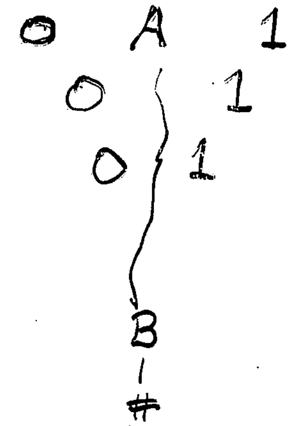
<u>Defn</u>: CFG = $(\underline{V}, \Sigma, R, S)$

<u>finite</u> set of vars. — rules set — start var.

$L(G) = \{ w \in \Sigma^* \mid S \overset{*}{\Longrightarrow} w \}$

<u>eg</u>: $G_1 = (\{A, B\}, \{0, 1, \#\}, R, A)$

R: $A \longrightarrow 0A1$
$A \longrightarrow B$
$B \longrightarrow \#$



$L = 0^n \# 1^n$, for $n \geq 0$

"derivation"

$G_2 = (\{S\}, \{ (,) \}, R, S)$

R: $S \rightarrow (S) \mid SS \mid \varepsilon$

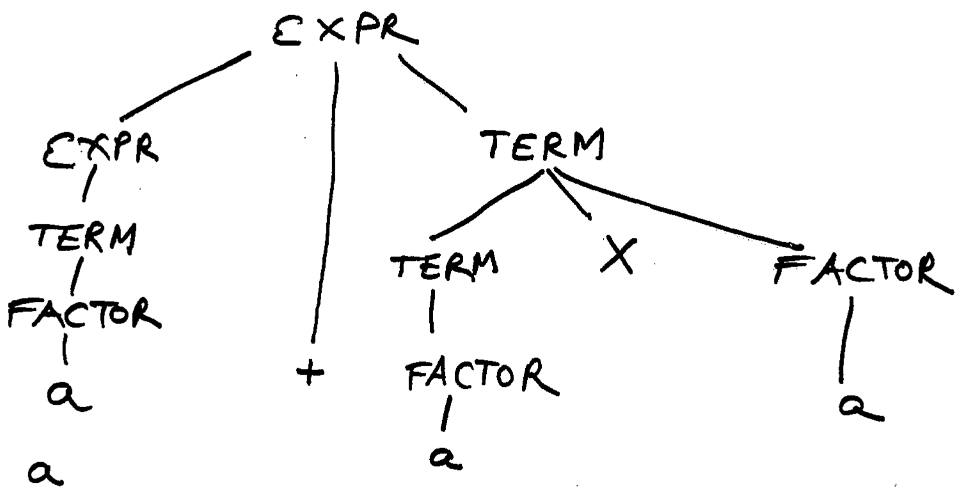
$G_3 = (\{ \text{EXPR}, \text{TERM}, \text{FACTOR} \}, \{ a, +, *, (,) \}, R, \text{EXPR})$

EXPR $\longrightarrow$ EXPR + TERM | TERM

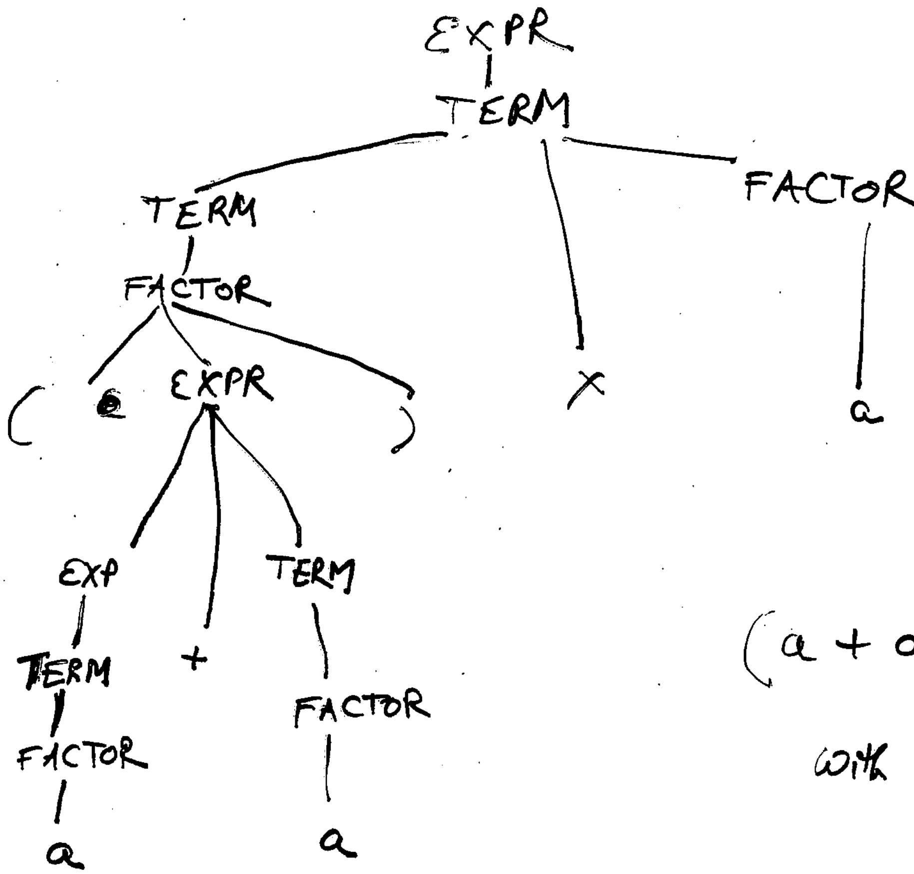
TERM $\longrightarrow$ TERM X FACTOR | FACTOR

FACTOR $\longrightarrow$ a | (EXPR)

G_3' // FACTOR $\rightarrow$ a | EXPR



$a + a \times a$



$(a + a) \times a$

with G_3': $a + a \times a$

Defn: w is derived AMBIGIOUSLY in a grammar if there are $\underbrace{\text{2+ parse trees}}$ → 2+ left-most derivations of w

Grammar G is AMBIGIOUS if there is some w that can be derived ambigiously

* Some CFGs are 'inherently AMBIGIOUS

eg $L = \{ a^i b^j c^k \mid i = j \text{ or } j = k \}$

Designing CFGs:

1) Exploit '$\cup$' of simpler Languages

eg $L = \{ 0^n 1^n \mid n \geq 0 \} \cup \{ 1^n 0^n \mid n \geq 0 \}$

$$S \longrightarrow S_1 \mid S_2$$
$$S_1 \longrightarrow 0 S_1 1 \mid \varepsilon$$
$$S_2 \longrightarrow 1 S_2 0 \mid \varepsilon$$

2) when there "linked" substrings, use rule like $R \to u R v$

3) // 'if L is regular

Convert DFA into CFG systematically

a) each state $\equiv$ var

b) $\delta(q_i, a) = q_j \equiv R_i \to a R_j$

c) $q_0 \equiv R_0$

d) $q_i \in F \equiv R_i \to \epsilon$

CHOMSKY NORMAL FORM

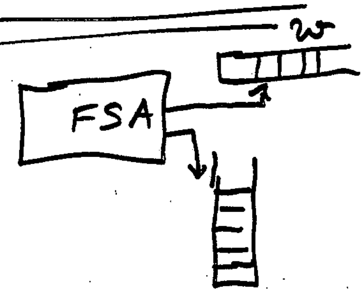
// standard format for CFG

Every rule must be as

$$A \longrightarrow BC$$
$$A \longrightarrow a$$
$$S \longrightarrow \varepsilon$$

Each CFG can be tranformed into CNF

Push-Down Automata (PDA) $\equiv$ CFG = CFL



PDA = FSA + stack (infinite capacity)

$$= (Q, \Sigma, \underbrace{\Gamma}_{\text{stack alphabet}}, \underbrace{\delta}_{Q \times \Sigma_\epsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\epsilon)}, q_0, F)$$

$\mathcal{P}(Q \times \Gamma_\epsilon)$ gives non-determinism // NPDA

$w = w_1 \quad w_2 \quad w_3 \quad \ldots\ldots w_m$

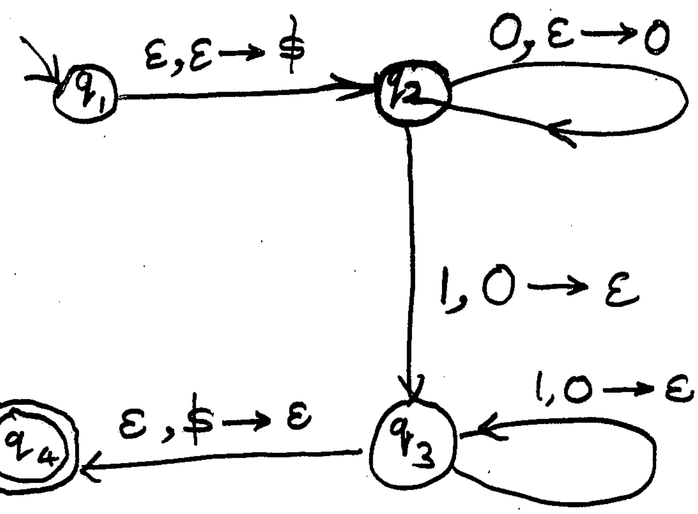
\+ top of stack entries

states: $r_0 \quad r_1 \quad r_2 \quad r_3 \quad r_m$

\+ modify Top entry of stack

$L = \{0^n 1^n \mid n \geq 0\}$

$\Gamma = \{0, \$\}$



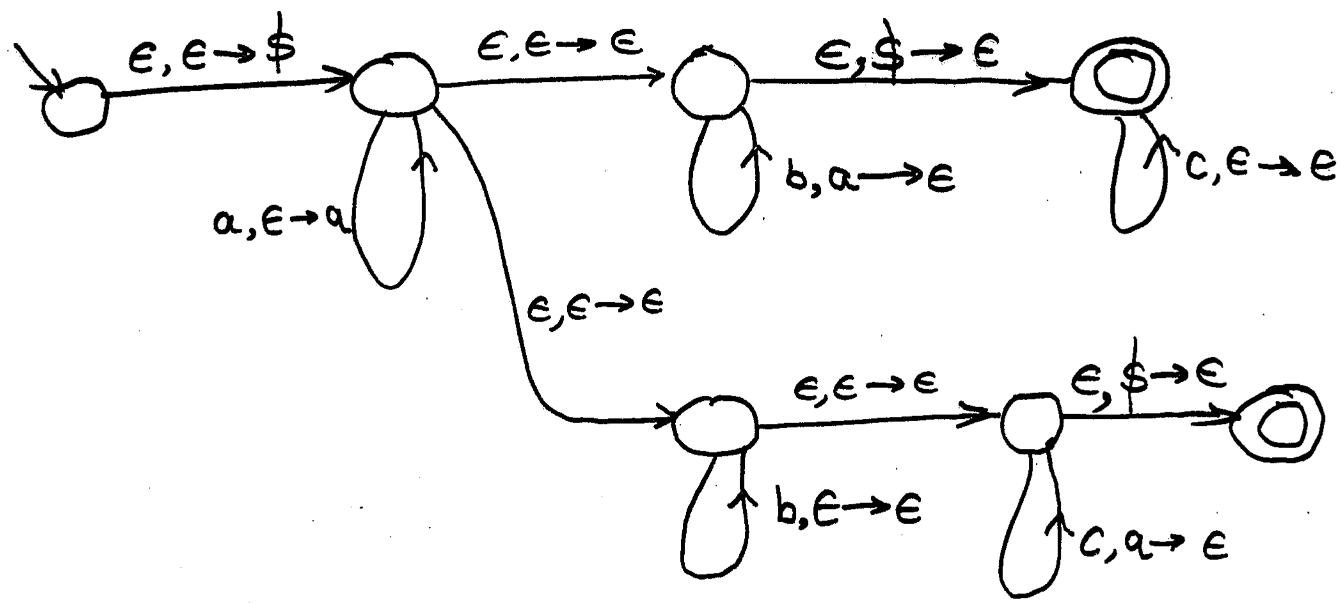
V.I.P:

N-PDA gives more POWER than D-PDA

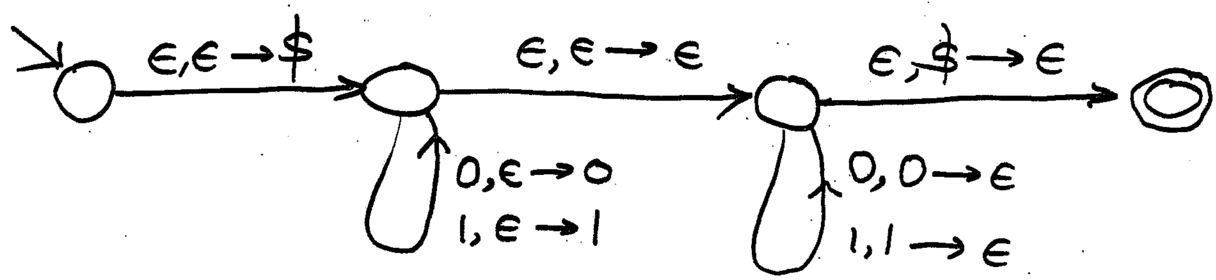
$\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \longrightarrow \{\ldots, (Q \times \Gamma_\epsilon), \ldots\}$

$\Sigma_\epsilon \rightarrow$	0			1			ϵ		
$\Gamma_\epsilon \rightarrow$	0	\$	ϵ	0	\$	ϵ	0	\$	ϵ
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \epsilon)\}$					
q_3				$\{(q_3, \epsilon)\}$				$\{(q_4, \epsilon)\}$	
q_4									

$L = \{ a^i b^j c^k \mid i, j, k \geq 0 \text{ and } [(i=j) \text{ or } (i=k)] \}$



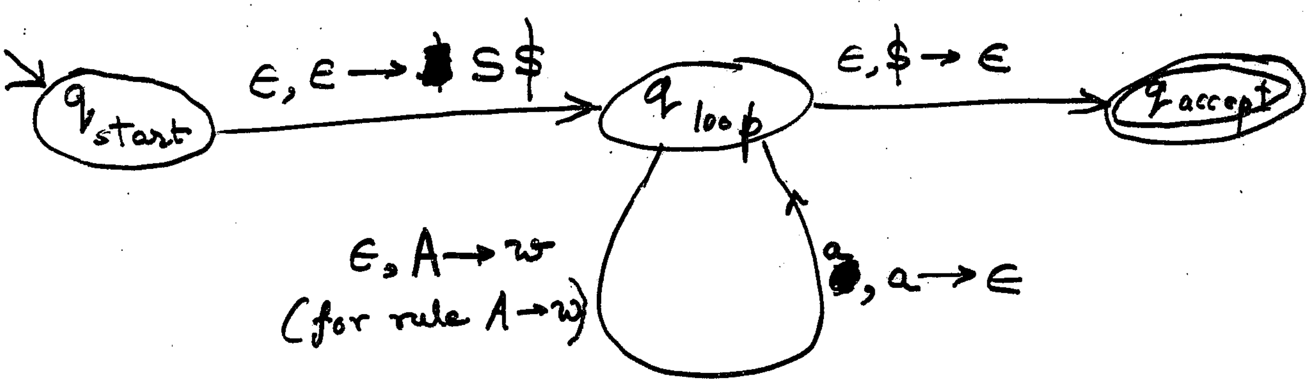
$L = \{ w w^R \mid w \in \{0,1\}^* \}$

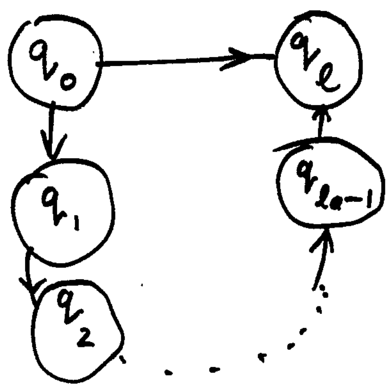


N-PDA $\longleftrightarrow$ CFG $\longleftrightarrow$ CFL

<u>Part 1</u>: Given a CFG, design a PDA ....

1) push \$ & S
2) <u>Loop</u> <u>Case</u>
   a) if top(stack) is a var: select some rule R and expand
   b) if top(stack) is a terminal: try matching it with next(w)
   c) if top(stack) = \$: if next(w) is null, ACCEPT.





read 'a', pop 's', push "$\underbrace{u_1 u_2 u_3 \cdots u_\ell}_{\text{R.H.S. of a rule.}}$"

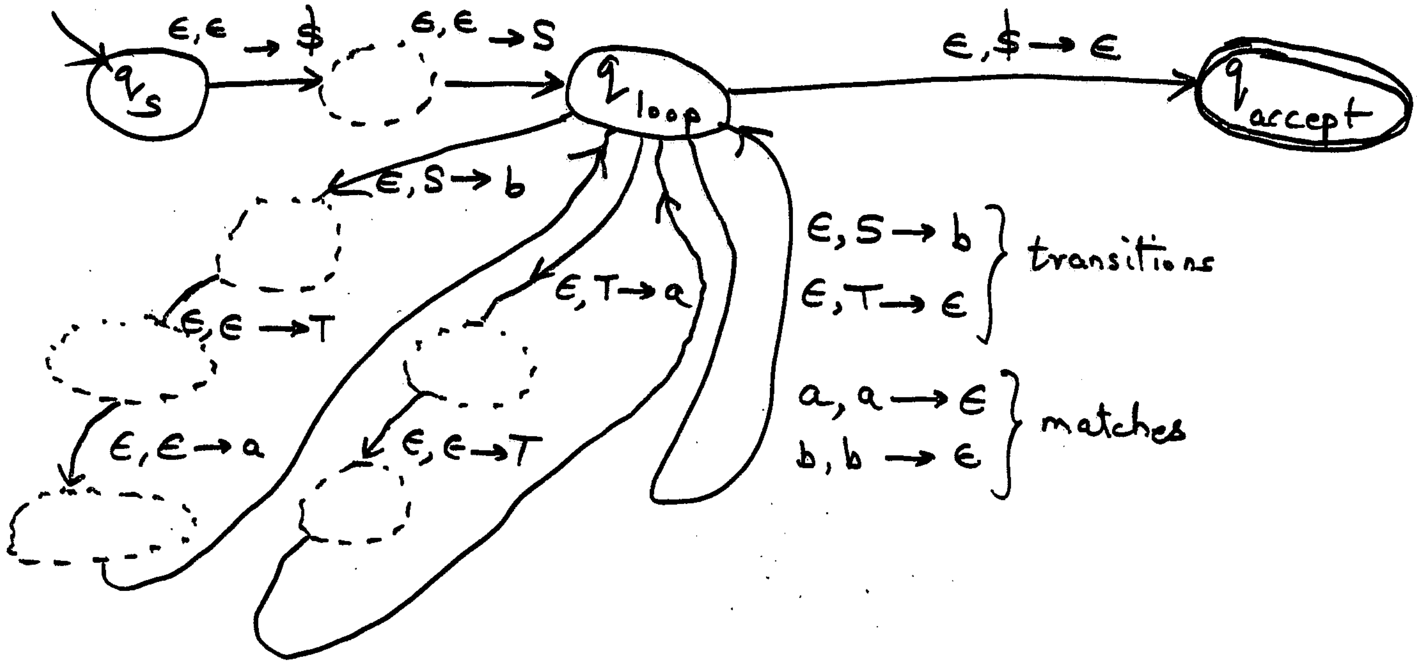
$\boxed{a, s \longrightarrow u_1 u_2 \cdots u_\ell}$

$\delta(q_0, a, s) \longrightarrow \{(q_1, u_\ell)\}$

$\delta(q_i, \epsilon, \epsilon) \longrightarrow \{(q_{i+1}, u_{\ell-i})\}$ $\forall i \in [1, \ell-1]$

$S \longrightarrow aTb \mid b$

$T \longrightarrow Ta \mid \varepsilon$



Part 2: Given a PDA, design a CFG ....

Preprocess given PDA | it satisfies:

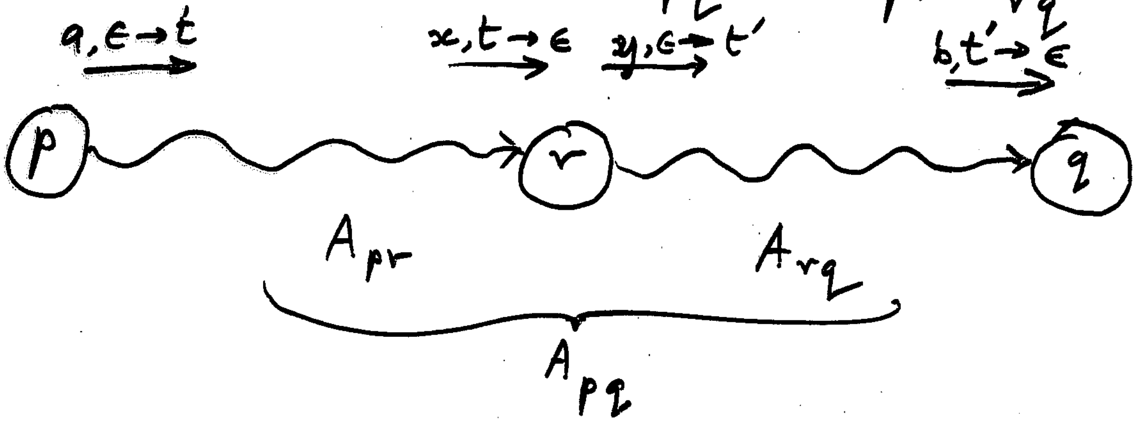
1) Only 1 accept state q_{accept}
2) Empty stack before accepting.
3) Each transition in δ, pushes or pops, but not both

$P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\})$, create $G = (V, \Sigma, R, S)$

- $V = \{A_{pq} \mid p, q \in Q\}$, $S = A_{q_0 q_{accept}}$
- R: $\forall p,q,r,s \in Q, \forall t \in \Gamma, \forall a,b \in \Sigma_\epsilon$:
  ① if $(r,t) \in \delta(p,a,\epsilon)$ && $(q,\epsilon) \in \delta(s,b,t)$: add $A_{pq} \to a A_{rs} b$

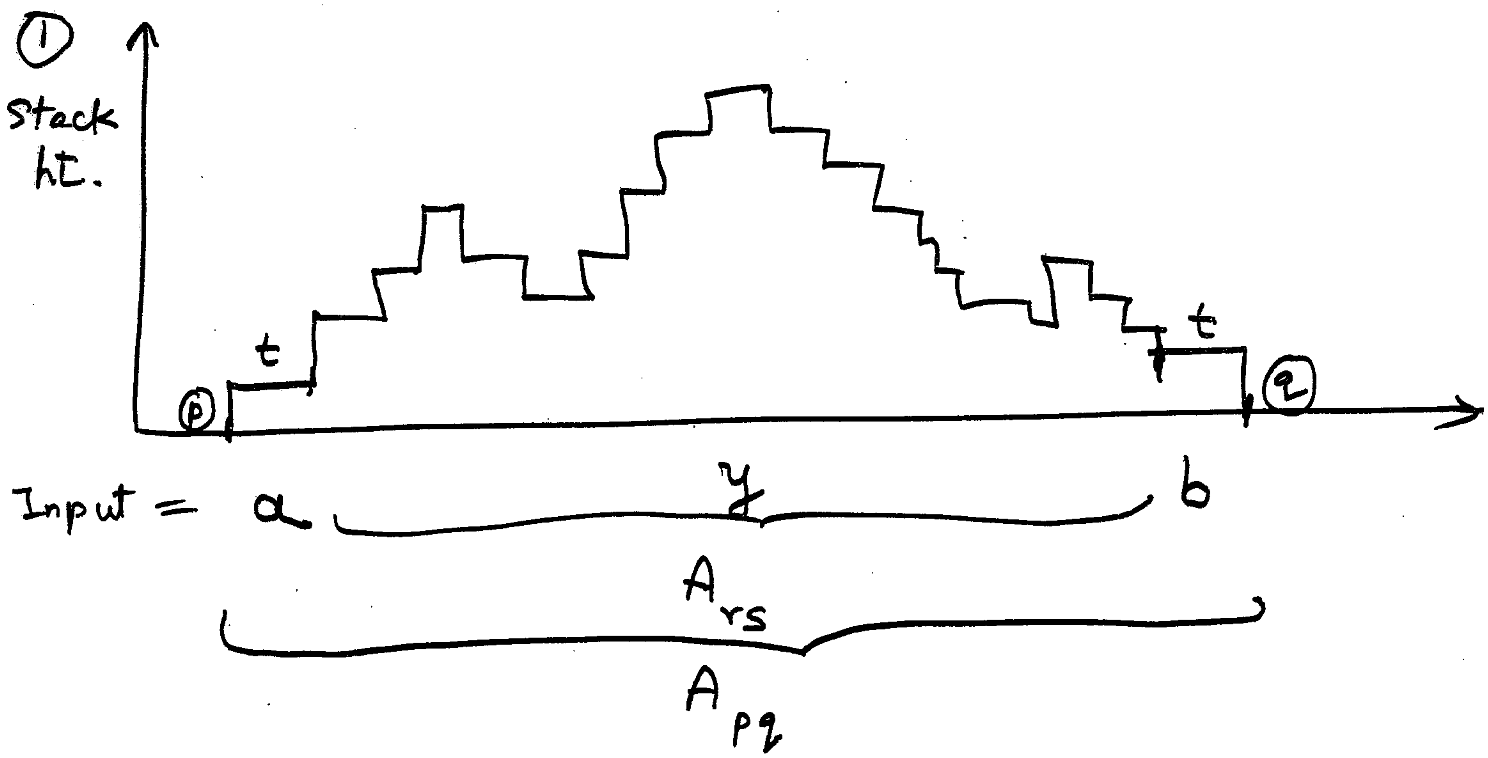
$$A_{pq} \longrightarrow a\, A_{rs}\, b$$

(2) ∀ p, q, r ∈ Q ; add rule $A_{pq} \longrightarrow A_{pr} A_{rq}$

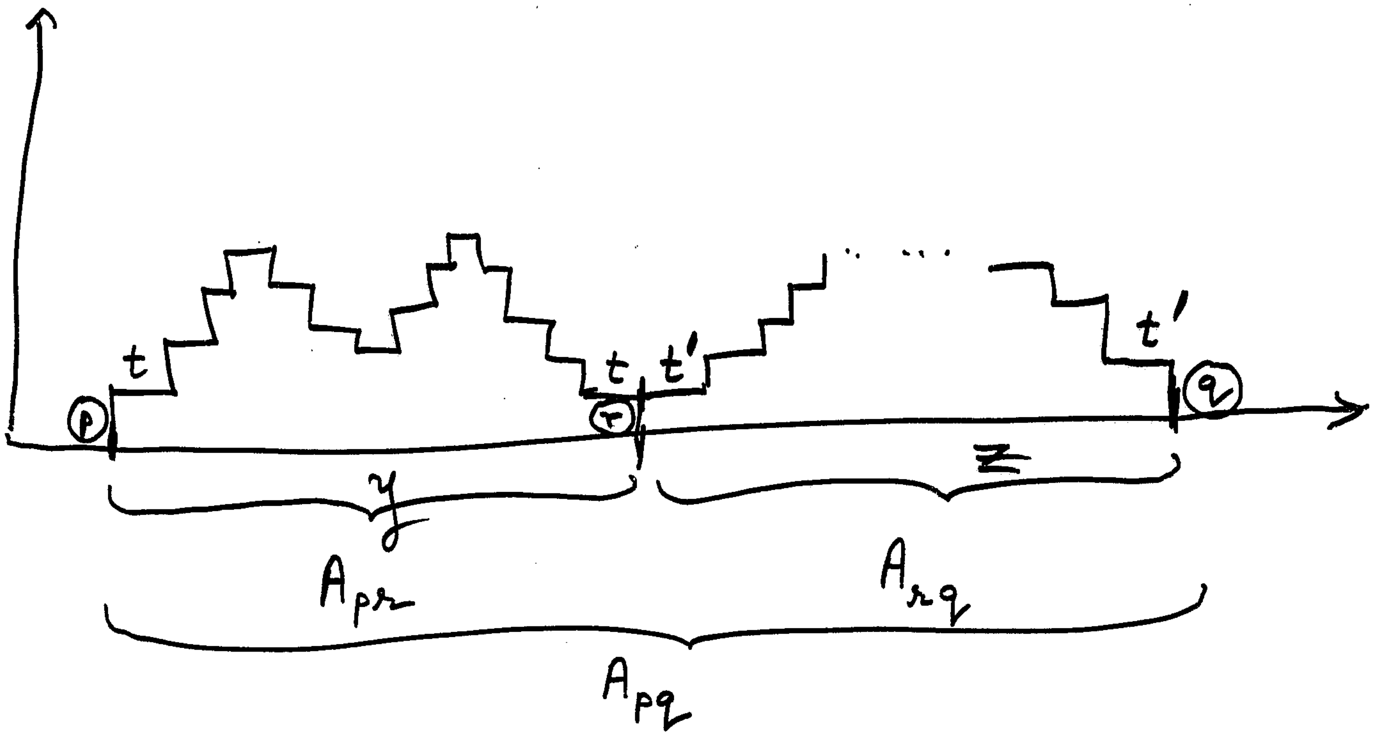


(3) ∀ p ∈ Q : add rule $A_{pp} \longrightarrow \epsilon$

(1)



2

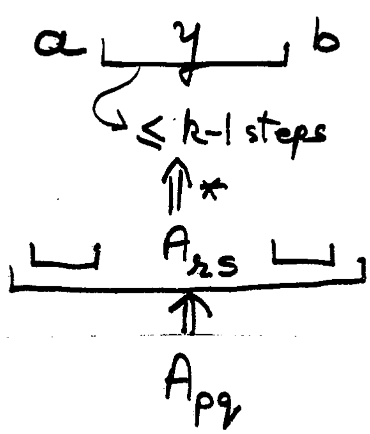


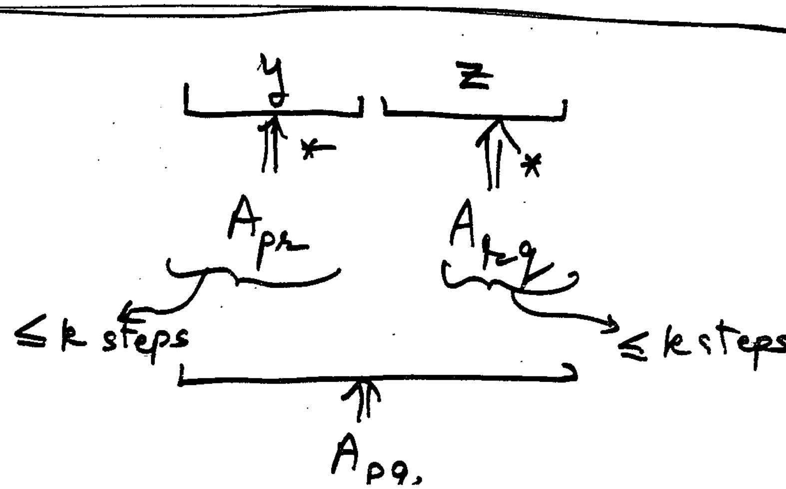
Part 2.1 (Lemma): If grammar A_{pq} generates string x, then x is "accepted" by PDA

Prove by induction on # steps in the derivation of x from A_{pq}

Part 2.2 (Lemma): If x "accepted" by PDA, then grammar A_{pq} that generates x.

Prove by induction on # steps in the computation of by PDA





NON-CFL

Pumping Lemma: For a CFL A, $\exists\ p$ (pumping length)

$\mid \forall s \in A$, where $|s| \geq p$

$\exists$ some way of $s =$ ~~$xy^iz \mid \forall i \geq 0, xy^iz \in A$~~

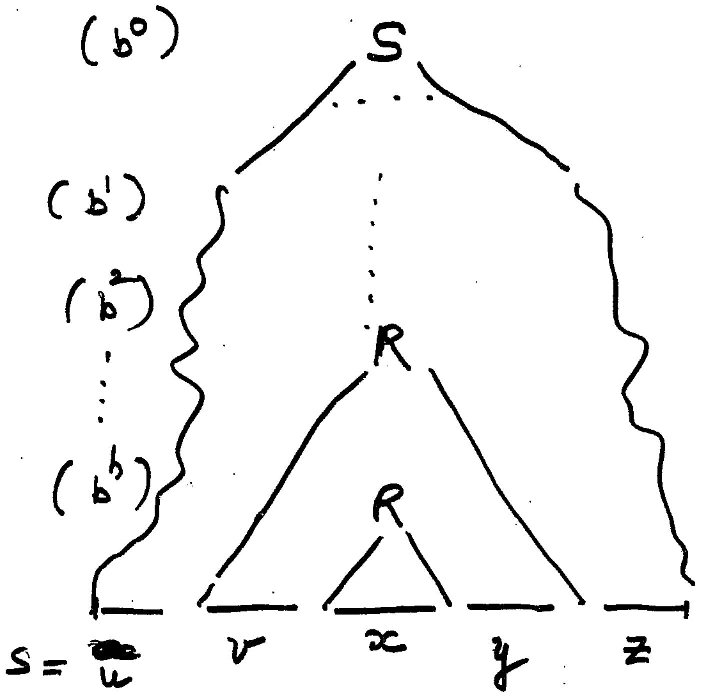
$uvxyz \mid$ ① $\forall i \geq 0,\ uv^ixy^iz \in A$

② $|vy| > 0$

③ $|vxy| \leq p$

// b = max # symbols in RHS of any rule

↓



$p = b^{|V|+1}$

τ: parse tree for s, having minimum # nodes // for cond ②

// height$(\tau) \geq |V|+1$

// some var. R must reappear on longest path in τ

R: the var. that repeats among the lowest $|V|+1$ var. occurrences // for cond. ③

ht = 1 ⇒ 2 lvls // b^0 leaves

ht = x ⇒ $x+1$ lvls // b^x leaves

ht = $|V|-1$ ⇒ $|V|$ lvls

ht = $|V|$ ⇒ $|V|+1$ lvls // $b^{|V|}$ leaves

ht = $|V|+1$ ⇒ $|V|+2$ lvls. // $b^{|V|+1}$ leaves

w/only terminals as leaves

$B = \{ a^n b^n c^n \mid n \geq 0 \}$

$s = a^p b^p c^p$

Burden : show no way of decomposing $s = uvxyz$

$$\overbrace{aa\cdots a\,bb\cdots b\,cc\cdots c}^{3p}$$

$- \ |vxy| \leq p \ \therefore \ vxy$ can contain only a's, only b's, only c's or only a's & b's or only b's & c's

(from ③ of P.L.)

$C = \{ a^i b^j c^k \mid i \geq j \geq k \geq 0 \}$

$s = a^p b^p c^p$

Then 'vxy' case : only 'a's : pump down
only 'c's : pump up
only 'b's : pump up or pump down.

RECAP : RL = RG = DFSA = NFSA == 0-NPDA

CFL = CFG = NPDA ($\supseteq$ DPDA) == 1-NPDA

= (D)-TM == 2-NPDA
(N)-TM