(Context Free Lang & PDA)

1. Standard CFL's & their grammers : (L → G , G → L(G))
2. Derivation of ambiguity : (LMD, RMD, D-tree, PD-tree, Ambiguity)
3. Membership Algorithm : (BNF, CYK, LL(K), LR(K))
4. Algo's in CFG
5. PDA programming

⇒ Standard CFL's :-

Counting comparision
- with order
- without order

String matching

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

[G : $S \to asb \mid \epsilon$]

$L = \{a^n b^n \mid n \geq 1\}$

[G : $S \to asb \mid ab$]

$L = \{w \mid n_a(w) = n_b(w)\}$

[$S \to asb \mid bsa \mid \epsilon \mid ss$]

OR

[$S \to asbs \mid bsas \mid \epsilon$]

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

Even pallindrome

[$S \to asa \mid bsb \mid \epsilon$]

Counting comparision

with order

- $L = \{a^n b^{2n} \mid n \geq 1\}$

  G: $S \to aSbb \mid \epsilon$

- $L = \{a^{2n} b^n \mid n \geq 1\}$

  G: $S \to aasb \mid \epsilon$

- $L = \{a^{2n+3} b^{3n+2} \mid n \geq 0\}$

  G: $S \to aaSbbb \mid a^3 b^2$

- $L = \{a^m b^n \mid m \geq n\}$

  G: $S \to AS \mid \epsilon$
  $S_1 \to aS_1 b \mid \epsilon$
  $A \to aA \mid \epsilon$

- $L = \{a^m b^n \mid m > n\}$

  G: $S \to AS_1$
  $S_1 \to aS_1 b \mid \epsilon$
  $A \to aA \mid a$

- $L = \{a^m b^n \mid m \leq n\}$

  G: $S \to S_1 B$
  $S_1 \to aS_1 b \mid \epsilon$
  $B \to bB \mid \epsilon$

without order

- $L = \{n_a(w) = n_b(w)$ & In every prefix $(n_a \geq n_b)\}$

  (((( )))) properly balanced paranthesis.

  $S \to (S) \mid SS \mid \epsilon$

- $L = \{n_a(w) = 2n_b(w)\}$

  write permutation of "aab"

  a S a S b

  a S b S a

  b S a S a

  $S \to aSaSb \mid aSbSa \mid bSaSa \mid SS \mid \epsilon$

String matching

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

  odd pallindrome

  $S \to asa \mid bsb \mid a \mid b$

- $L = \{ww^R \cup wxw^R$ or $w = w^R \mid w \in \{0,1\}^{even}\}$

  All pallindrome

  $S \to asa \mid bsb \mid a \mid b \mid \epsilon$

Counting comp | string matching

with order | without order

$L = \{ a^m b^n \mid m \neq n \}$

$m > n$ or $n > m$

Q: $S \to S_1 \mid S_2$

$S_1 \to AS_1 b$

$S_2 \to S_2 B$

$S_3 \to aS_3 b \mid \epsilon$

$A \to aA \mid a$

$B \to bB \mid b$

$L = \{ a^m b^n \mid m = 2n+1 \}$

Q: $S \to aasb \mid a$

$L = \{ a^m b^n \mid m \geq 2n+1 \}$

Q: $S \to AS_1$

$S_1 \to aas_1 b_1 \mid a$

$A \to aA \mid \epsilon$

$L = \{ a^m b^n \mid m > 2n+1 \}$

Q: $S \to AS_1$

$S_1 \to aas_1 b \mid a$

$A \to aA \mid a$

$L = \{ n_1(w) = n_2(w) \}$

balanced paranthesis's

$S \to (S) \mid)Sc \mid SS \mid \epsilon$

- $L = \{ a^m b^n \mid m = 2m+1 \}$

G: $S \rightarrow aSbb \mid b$

- $L = \{ a^m b^n \mid n \geq 2m+1 \}$

G: $S \rightarrow S_1 B$
$S_1 \rightarrow aS_1bb \mid b$
$B \rightarrow bB \mid \epsilon$

- $L = \{a^n b^n\} \cup \{a^n b^{2n}\}$

$S \rightarrow S_1 \mid S_2$
$S_1 \rightarrow aS_1b \mid \epsilon$
$S_2 \rightarrow aS_2bb \mid \epsilon$

- $L = \{ a^m b^m c^n \mid m, n \geq 0 \}$

$S \rightarrow S_1 C$
$S_1 \rightarrow aS_1b \mid \epsilon$
$C \rightarrow cC \mid \epsilon$

- $L = \{ a^m b^n c^m \mid m, n \geq 0 \}$

$S \rightarrow aSc \mid B$
$B \rightarrow bB \mid \epsilon$

- $L = \{ a^m b^n c^{m+n} \mid m,n \geq 0 \}$

$a^m b^n c^n c^m$

$S \rightarrow aSc \mid B$

$B \rightarrow bBc \mid \epsilon$

- $L = \{ a^m b^n c^p \mid m=n \text{ or } n=p \}$

$S \rightarrow XC \mid AY$

$X \rightarrow aXb \mid \epsilon$

$Y \rightarrow bYc \mid \epsilon$

$C \rightarrow cC \mid \epsilon$

$A \rightarrow aA \mid \epsilon$

- $L = \{ a^m b^n c^p d^q \mid m+n = p+q \}$

String Matching

(CFL $\rightarrow$ REG)

- $\{ ww^R \mid w \in [0]^* \} = (00)^*$
- $\{ ww^R \mid w \in [0,1] \} = (00, 11)$
- $\{ (ww^R)^* \mid w \in \{0,1\}^* \} = (0+1)^*$
- $\{ (w)^* w^R \mid w \in \{0,1\}^* \} = (0+1)^*$
- $\{ wxw^R \mid w \in \{0,1\}^*, x \in \{0,1\}^* \} = (0+1)^*$

wxw^R $ww^R x$ xww^R

- $\{ wxw^R \mid w \in (0,1)^*, x \in (0,1)^+ \} = (0+1)^+$
- $\{ wxw^R \mid w \in (0,1)^+, x \in (0,1)^* \}$

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

- $L = \{a^m b^n c^p d^q \mid m=q,\ n=p\}$

- $\{\omega x \omega \mid \omega \in (0,1)^+ , x \in (0,1)^*\}$

This is CSL, it is not Regular.

- $\{uxu^R y \mid u,x,y \in \{0,1\}^*\}$

- $L = \{a^m b^n c^p d^q \mid m+p = n+q\}$

$(0x0y + 1x1y)$ This is regular

rearrange,

$(m-n = q-p)$

NOTE :-

PUSH — POP is not clear in PDA then it is CFL but not DCFL. i.e if PDA can't find the middle point in string from where POP opn has to be done.

eg :-

$L_1 = \{ w w^R \mid w \in \{0,1\}^* \}$

aab|baa

no-middle

So, L_1 can be done using NPDA not by DPDA

(CFL)

$L_2 = \{ w x w^R \mid w \in \{a,b\}^*, x \in \{a,b\} \}$

= aabxbaa

middle

L_2 can be done by DPDA

(DCFL)

But if,

$L_2 = \{ w x w^R \mid w \in (a,b)^* \text{ & } x \in \{a,b\} \}$

= aababaa

we can know a is middle but, PDA can't.

L_2 can't recognize by DPDA

(CFL)

Q. which of the following are regular?

(i) ✓ $\{wxw^R \mid w,x \in (0,1)^*\}$

(ii) $\{xww^R \mid w,x \in (0,1)^+\}$

(iii) $\{ww^Rx \mid w,x \in (0,1)^+\}$

⇒ Left-most & Right-most Derivation :-

- Since a language has more than one derivation so, we have to follow some standard way to find derivation.

- So for a grammer G if w such that w has 2 or more than derivation tree, then G is ambiguous.

LMD

↓

Top-down compiler

(Leftmost variable is substituted first)

RMD

↓

Bottom-up compiler

(Rightmost variable is substituted first)

eg:- G: $S \to AB$

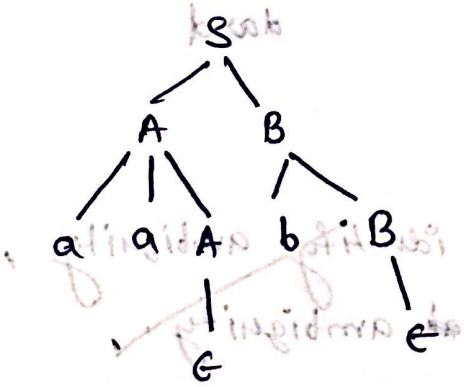
$A \to aaA / \epsilon$

$B \to bB / \epsilon$

$(\therefore L(G) = (aa)^*(b)^*)$

let take $\omega = aab$

LMD: $S \longrightarrow AB \longrightarrow aaAB \longrightarrow aaB \longrightarrow aabB \longrightarrow aab$



Single derivation tree

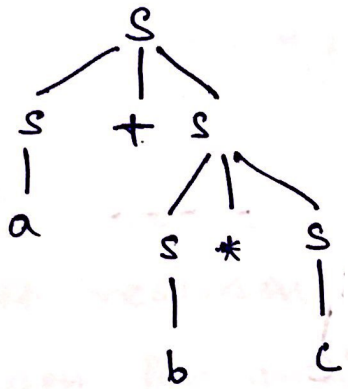
so, G is not ambiguous.

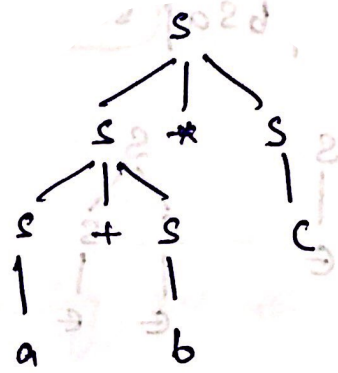
NOTE:- If any w of $L(G)$ has more than one LMD or RMD then it will cause ambiguity.

(Two derivation tree $\equiv$ Two LMD $\equiv$ Two RMD)

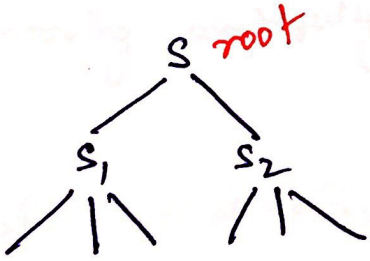
eg:- G: $S \rightarrow S+S \mid S*S \mid a \mid b \mid \epsilon$

Let $w = a+b*\epsilon$





Two derivation tree, so it is ambiguous grammer



D-Tree

P D-Tree

both

root missing

Partial D-tree

where root can be present or not

Ambiguity :-

$S \rightarrow AB \mid cD$

$A \rightarrow aaA \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

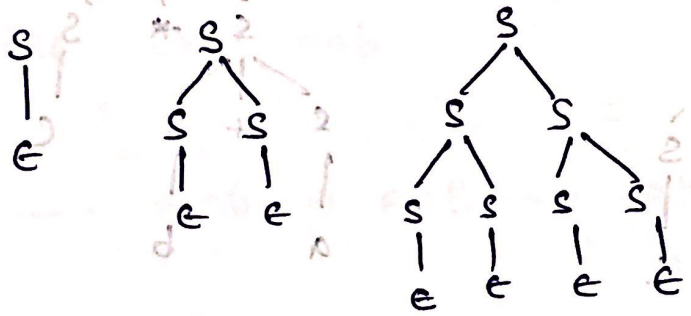
$C \rightarrow ccC \mid \epsilon$

$D \rightarrow dD \mid \epsilon$

if (something is common) { ambiguous }

$S \rightarrow aSb \mid bSa \mid SS \mid \epsilon$

for $w = \epsilon$



So grammer is ambiguous.

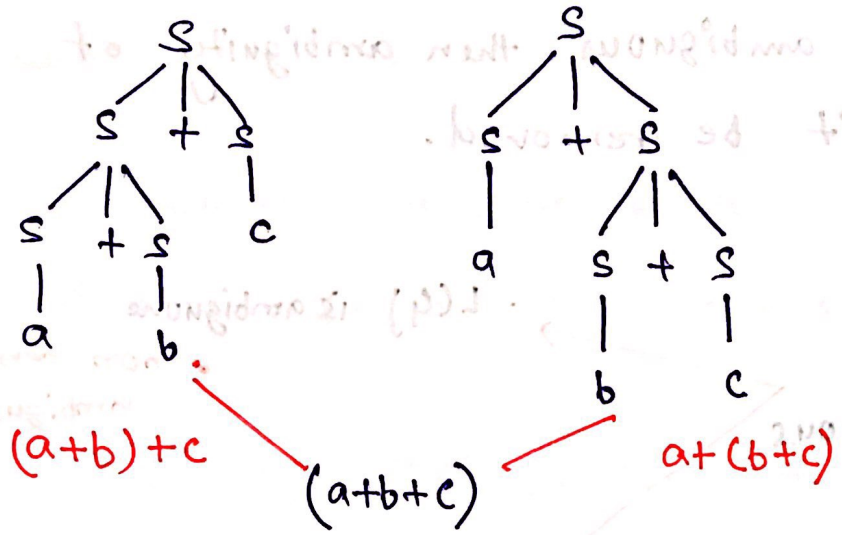
Left recursion :-

$S \rightarrow S+S \mid S*S \mid a \mid b \mid c$

$S \rightarrow S+S \mid a \mid b \mid c$

$S \rightarrow S*S \mid a \mid b \mid c$

Both are ambiguous



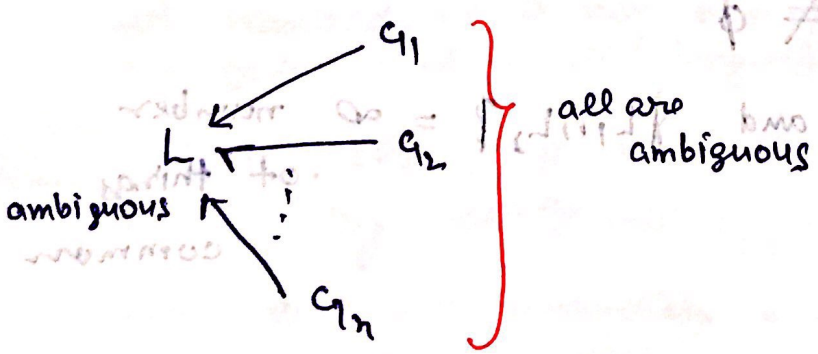
Left factoring :-

$S \rightarrow aA \mid aABC$

$\alpha x \mid \alpha y$

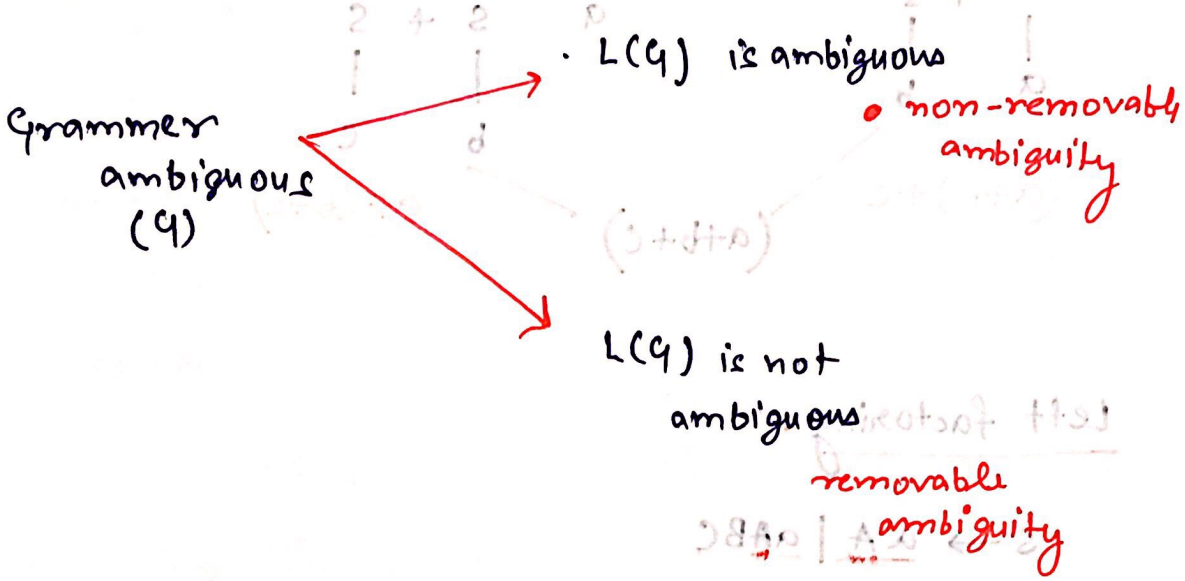
NOTE:

- Left recursion & left factoring are not only the reason for ambiguity.
- we can remove left recursion & left factoring
- Removing ambiguity is undecidable & checking also.



* Language can only be ambiguous when all the grammer generating L is are ambiguous.

- If Language is ambiguous then ambiguity of grammer can't be removed.



⇒ Conditions for ambiguous Language:-

i) union of L must be present
i.e $(L = L_1 \cup L_2 \cup L_3 \dots)$

(ii) union should not combine into one condition i.e

$L = L_1 \cup L_2 \cup L_3 \dots$

$L_1 \cup L_2 \cup L_3$ → should not combine

(iii) $L_1 \cap L_2 \neq \phi$

and $\{L_1 \cap L_2\} = \infty$ number of things common

⇉ Membership Algorithm :- BFP, CYK, LL(K), LR(K)

These algorithm recognize whether $w \in L(G)$ < Yes / No in a finite amount of time where $w \in \Sigma^*$, CFG.

($w \in L(G) \Leftrightarrow \exists$ a Derivation for w using P of G.)

(i) **Brute Force passing :-** Try all possible derivation ~~to find~~ for w. This is not efficient as it can run for infinite amount of time, or very large time. [$O(k^n)$ where $k = |\Sigma|$]

NOTE :-

[G has no ϵ production $\longrightarrow$ $\epsilon \notin L(G)$

$\epsilon \in L(G) \longrightarrow \exists$ ϵ-productions]

eg: $G: S \to aA$

$A \to \epsilon$ ← ϵ-production

as ϵ-production is present but $\epsilon \notin L(G)$

* ϵ-production produce contraction which are not required, as it can shrink the sentential form.

* Both ϵ & unit production causes an infinite loop or hangs the machine.

- After removing ϵ- production & unit production BFP can find or check $\omega \in L(G)$ in a finite time.

(ii) **Cocke-Younger-Kasami (CYK) :-** It employs bottom-up parsing and dynamic programming. It operates only on CFG given in CNF. [$O(n^3 \cdot |G|)$ where n is length of parsed string and $|G|$ is size of CNF grammer G.]

NOTE:-

* For every CFG (G) $\exists$ G' in CNF such that,

$$\left(L(G') = L(G) - \epsilon\right)$$

so, G and G' can't be equivalent always.

* $O(n^3)$ is the best time complexity.

(iii) **LL(K) & LR(K) :** They have linear time complexity.

[LL(K) ⟶ Top-down compiler

LR(K) ⟶ Bottom-up compiler]

LL(K) :- G is LL(K) iff when "K symbols" of the I/p are presented to the compiler at a time.

The production that is to be used in every step of the derivation is uniquely determined.

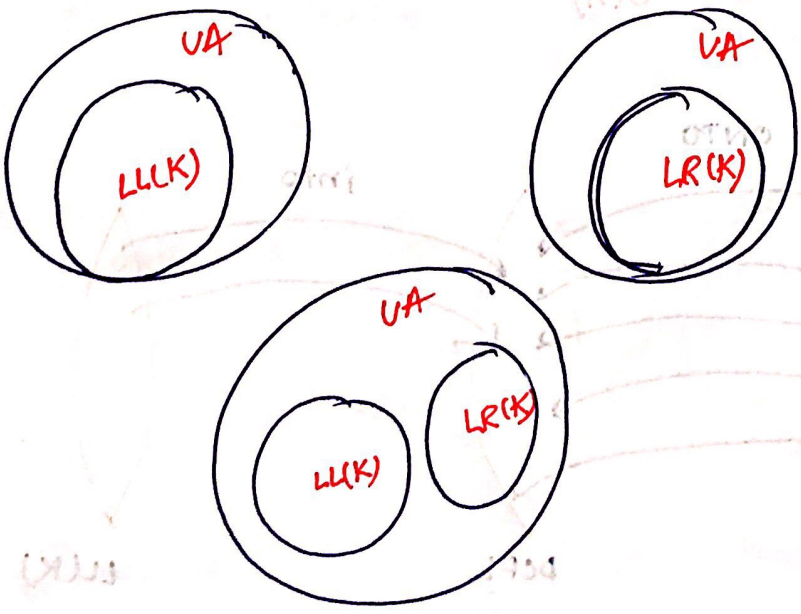
	a	b
S	P_1	P_3
A	P_4	P_2
B	P_6	P_5

NOT AMBIGUOUS

- So, an ambiguous G can't be LL(K) & LR(K) $\forall$ K.

LL(K) → UA (Unambiguous grammer)

LR(K) → UA



- Remove ε- production & unit production
- Remove Left recursion & left factoring as they can cause ambiguity.

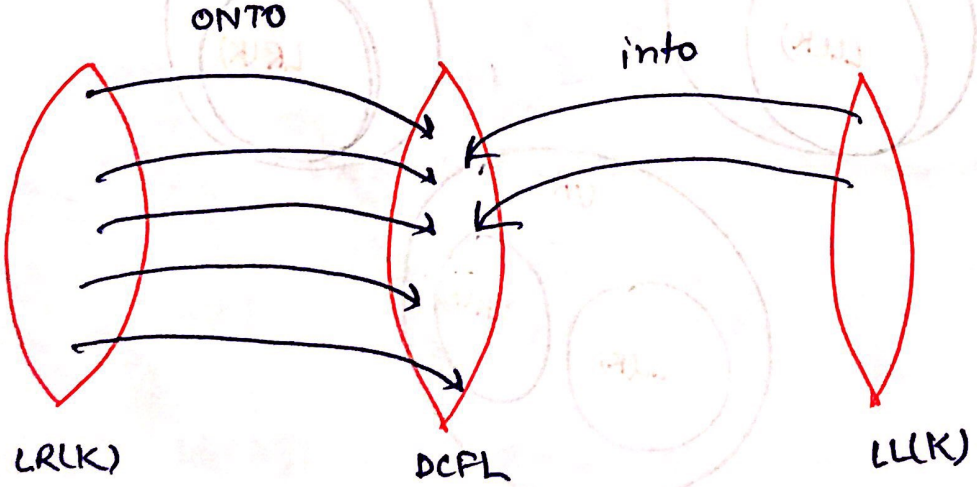
NOTE: (checking G is LL(K) or LR(K) ⟶ Decidable)

So for checking unambiguous grammer:

G —checks (decidable)→ LL(K) or LR(K)
- YES → UA
- NO → can't say (can be ambiguous or unambiguous)

(S (simple grammer) → LL(K))

O(n)



- all o/p of LR(K) maps to all DCFL (ONTO)

- o/p of LL(K) maps to some DCFL (INTO)

CNF

$V \rightarrow VV$

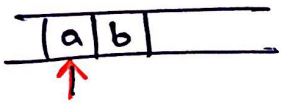
$V \rightarrow T$

GNF

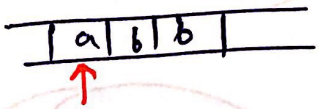
$V \rightarrow TV^*$

LL(K) VS LR(K) :-

- LL(2)
  - 1 current symbol + 1 lookahead symbols



- LR(2)
  - 2-look ahead symbol



- LL(K)
  - left to right
  - LMD

- LR(K)
  - left to right
  - RMD

- Top-down approach

$S \rightarrow \rightarrow \rightarrow \rightarrow w$

- Recursive-descent compiler

- Bottom-up approach

$S \leftarrow \leftarrow \leftarrow \leftarrow w$

- Shift reduce compiler

Properties of LL(K) & LR(K) :-

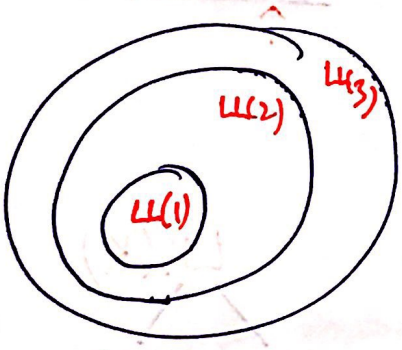
- LL(K) → UA and LR(K) → UA ; since both have only one production for a input symbol.

- Both ~~are~~ have linear time complexity O(n).

- $(\forall_{DCFL} \rightarrow \exists\, LR(K))$ ONTO

- $(\forall\, LL(K) \rightarrow \exists\, DCFL)$ INTO

- $(LL(K) \rightarrow LL(K') \quad \forall\ K' \geq K)$

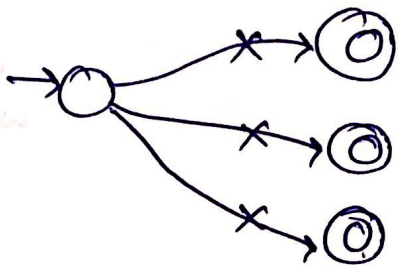


similarly $(LR(K) \rightarrow LR(K') \quad \forall\ K' \geq K)$

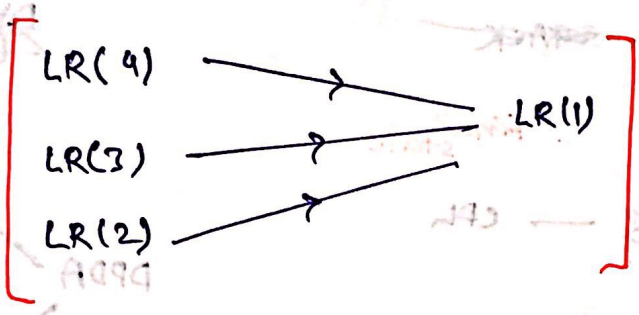
- Given G (CFG) checking if G is $LL(K)_{LR(K)}$ for some fixed K is Decidable.

- $L(M) = \phi$ iff no direct path to final state

for regular lang only



- G is LR(K) $K \geq 2$, we can get an equivalent G' which is LR(1).



- $\forall$ DCFL $\rightarrow$ $\exists$ LR(0) or LR(1) ✓
  $\rightarrow$ $\exists$ LR(1) ✓
  $\rightarrow$ $\exists$ LR(0) ✗
- if we have a DCFL with prefix property $\rightarrow$ $\exists$ LR(0)

prefix property: prefix of any string shouldn't be contained inside L.

✓ $L_1 = \{100, 010, 11\}$

✗ $L_2 = \{100, 010, 01\}$ prefix of '010' is a string in L

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

ϵ is prefix which is contained by L

Not LR(0)

$L = \{ a^n b^n \mid n \geq 1 \}$

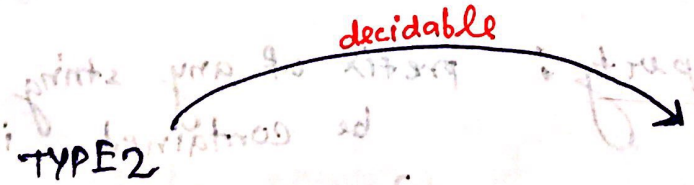
ϵ is absent in L

LR(0)

NOTE:

Low power M/c ⟶ High-power M/c

Decidable



⇒ Algorithm in CFG :-

(i) Removal of ϵ-production

(ii) Removal of unit-production

(iii) Removal of useless production

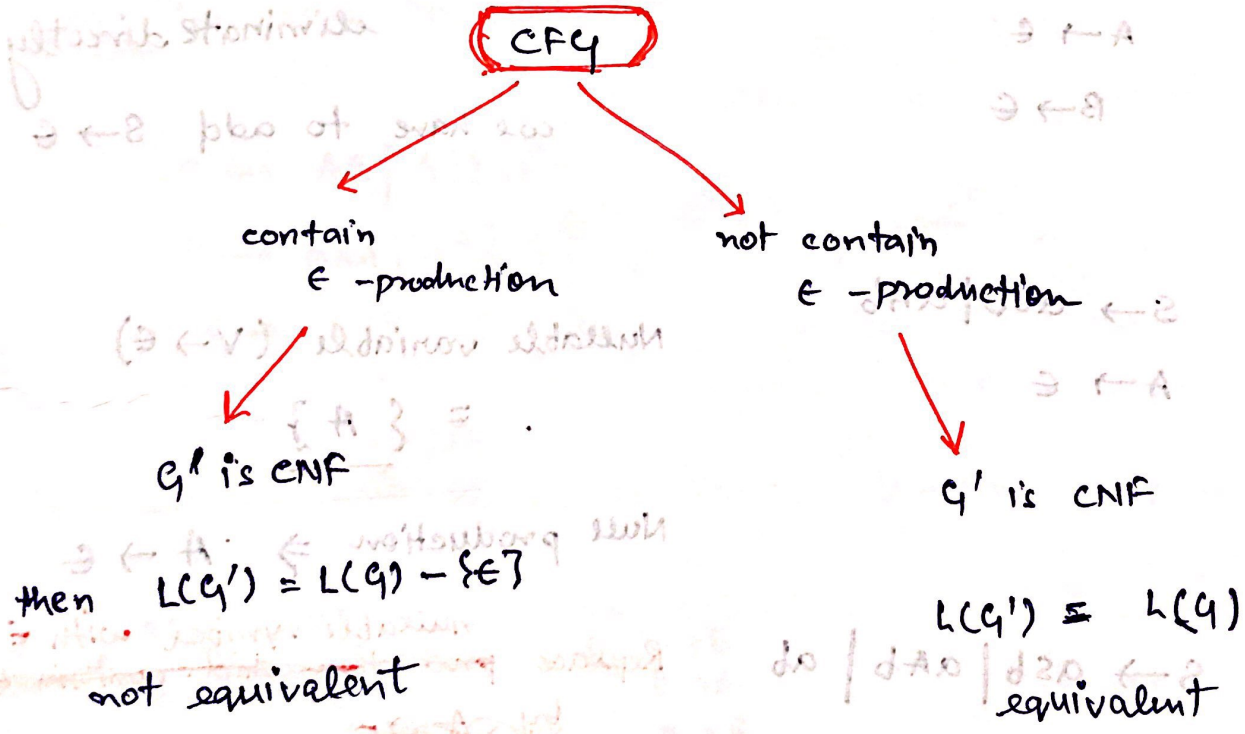
(iv) Removal of Left Recursion

(v) Removal of Left Factoring

$\forall$ CFG $G \equiv G'$ where G' is CFG without unit, LR or LF.

(vi) CFG $\longrightarrow$ CNF

(vii) CFG $\longrightarrow$ GNF



(i) Removal of ϵ-production :-

(if G has no ϵ-production $\longrightarrow$ $\epsilon \notin L(G)$)

- ϵ production can only be eliminated in G when $\epsilon \notin L(G)$ i.e ϵ is not a member of language of G.

eg:-

G_1: $S \longrightarrow aA$ as $\epsilon \notin L(G_1)$ so, ϵ-production can be eliminated
$A \longrightarrow \epsilon$

G_2: $S \longrightarrow aAB \mid a \mid \epsilon$ as $\epsilon \in L(G_2)$ so we can't eliminate directly
$A \longrightarrow \epsilon$
$B \longrightarrow \epsilon$
we have to add $S \longrightarrow \epsilon$

eg:- $S \longrightarrow asb \mid aAb$
$A \longrightarrow \epsilon$

Nullable variable $(V \longrightarrow \epsilon)$ = $\{A\}$

Null production $\Rightarrow$ $A \longrightarrow \epsilon$

Replace nullable symbol with ϵ

$S \longrightarrow asb \mid aAb \mid ab$

$S \longrightarrow asb \mid ab$

aAb is useless production, as A is useless variable.

eg:-

$S \rightarrow AB$
$A \rightarrow aAA \mid \epsilon$
$B \rightarrow bBB \mid \epsilon$

Null variable = $\{A, B, S\}$

Null production : $A \rightarrow \epsilon$
$B \rightarrow \epsilon$

So, substitute $A \rightarrow \epsilon$ and $B \rightarrow \epsilon$ in production rules which have A and B variable. (nullable symbols)

$S \rightarrow AB \mid A \mid B$
$A \rightarrow aAA \mid aA \mid a$
$B \rightarrow bBB \mid bB \mid b$

since ϵ belongs to L(G) as $S \rightarrow \epsilon$ when
$A \rightarrow \epsilon$
$B \rightarrow \epsilon$

so, we have to add it.

$S \rightarrow AB \mid A \mid B \mid \epsilon$
$A \rightarrow aAA \mid aA \mid a$
$B \rightarrow bBB \mid bB \mid b$

(ii) Removal of unit production :-

$V \rightarrow V_1$; $V_1 \rightarrow T$

eg:-

$S \rightarrow Aa \mid B$
$B \rightarrow A \mid bb$
$A \rightarrow a \mid bc \mid B$

$S \rightarrow B$
$B \rightarrow A$
$A \rightarrow B$

$S \rightarrow Aa \mid bb \mid a \mid bc$
$B \rightarrow a \mid bc \mid bb$
$A \rightarrow a \mid bc \mid bb$

(iii) Removal of use-less production :-

Use-ful symbol :- • Reachable from S
• derive some part of string

eg:- $S \to AB \mid a$
$A \to BC \mid b$
$B \to aB \mid C$
$C \to aC \mid B$

$(T = \{a, b\} \; ; \; V = \{S, A, B, C\})$

since B and C are reachable from S but they will never generate a string w.

so, B and C are use-less symbol.

$S \to a$
$A \to b$

remove productions containing useless symbol

A is now useless

so, finally we have $S \to a$

(iv) Removal of left recursion :-

$$A \to A\alpha_1 \mid A\alpha_2 \mid \dots\dots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \beta_3 \dots \mid \beta_n$$

$$\left\{ \begin{array}{l} A \to \beta A' \\ A' \to \alpha A' \mid \epsilon \end{array} \right\}$$

eg:- $S \to S+S \mid S*S \mid a \mid b \mid c \mid \epsilon$
(α_1 = $+S$, α_2 = $*S$; β_1 β_2 β_3 β_4)

$$\left[\begin{array}{l} S' \to +SS' \mid *SS' \mid \epsilon \\ S \to aS' \mid bS' \mid cS' \mid S' \end{array} \right]$$

(v) **Removal of Left factoring :-**

$$A \rightarrow \alpha x \mid \alpha y \mid \alpha z \mid \beta_1 \mid \beta_2 \mid \beta_3$$

$$\begin{bmatrix} A \rightarrow \alpha A' \mid \beta_1 \mid \beta_2 \mid \beta_3 \\ A' \rightarrow x \mid y \mid z \end{bmatrix}$$

eg:-

$$S \rightarrow aAB \mid aABD \mid aD$$

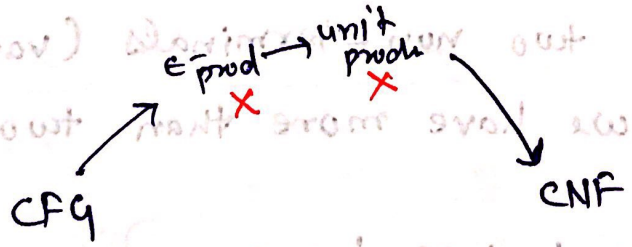
$$\Downarrow$$

$$\begin{bmatrix} S \rightarrow aA' \\ A' \rightarrow AB \mid ABD \mid D \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} S \rightarrow aA' \\ A' \rightarrow ABA'' \mid D \\ A'' \rightarrow \epsilon \mid D \end{bmatrix}$$

(vi) **CFG → CNF :-**



$S \rightarrow aAB \mid BaC \mid a \mid Bd \mid aB$

$A \rightarrow ABa \mid Ba \mid BC$

CNF: $\begin{bmatrix} V \rightarrow VV \\ V \rightarrow T \end{bmatrix}$

- First remove ϵ and unit productions
- Find single terminals and associate them with a variable.

$$\begin{bmatrix} D_a \rightarrow a \\ D_d \rightarrow d \\ D_c \rightarrow c \end{bmatrix}$$

So, $S \rightarrow \underbrace{D_aAB}_{D_1} \mid \underbrace{BD_aD_c}_{D_2} \mid a \mid BD_d \mid D_aB$

$A \rightarrow \underbrace{ABD_a}_{D_3} \mid BD_a \mid BC$

- Now combine two non-terminals (variables) to one wherever we have more than two variables.

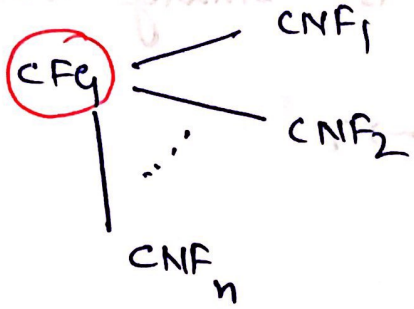
$S \rightarrow D_1B \mid D_2D_c \mid a \mid BD_d \mid D_aB$

$A \rightarrow D_3D_a \mid BD_a \mid BC$

$D_1 \rightarrow D_aA \quad D_3 \rightarrow AB \quad D_d \rightarrow d$

$D_2 \rightarrow BD_a \quad D_a \rightarrow a \quad D_c \rightarrow c$

CNF



* CNF is not unique for a given CFG

NOTE:-

* $w \in L(G)$ where G is in CNF and $|w| = n$

⟶ every derivation of w has $2n-1$ steps.

eg:-

$S \to AB \mid CD \mid a$

$A \to BC \mid DE \mid b$

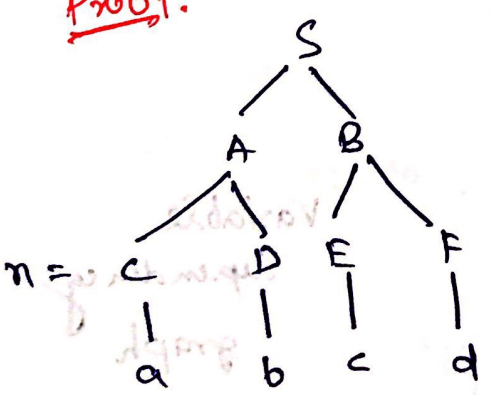
then $w = abcd$; $|w| = 5$

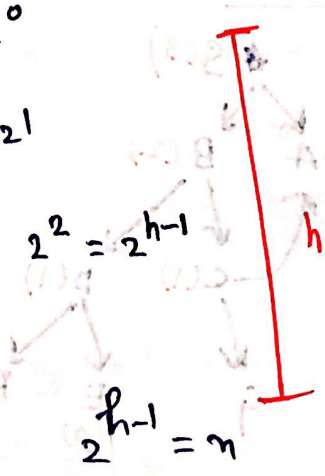
so, no. of steps of derivation $= 2(5) - 1$

$= 9$

* ⟶ height of derivation tree $= \lceil \log_2 n \rceil + 1$

Proof:





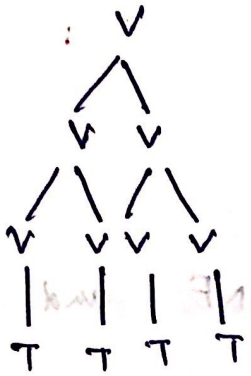
$= \log_2 5 + 1$

$= 3 + 1 = 4$

so, $h = \lceil \log_2 n \rceil + 1$

* Every D-tree in CNF is Binary tree.

as $V \rightarrow VV$ or $V \rightarrow T$



* If the rank of every non-terminal is finite

$\Rightarrow$ L(G) is finite.

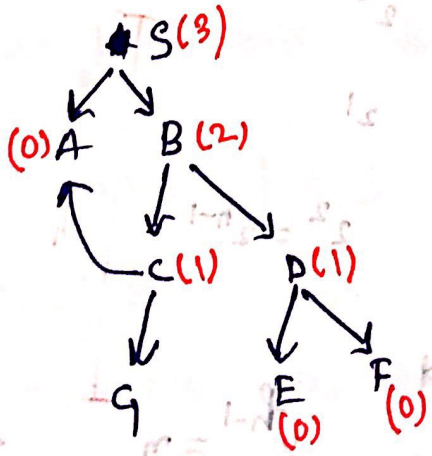
* If G is CFG then :

$(L(G) = \emptyset)$ iff S is useless symbol

starting symbol

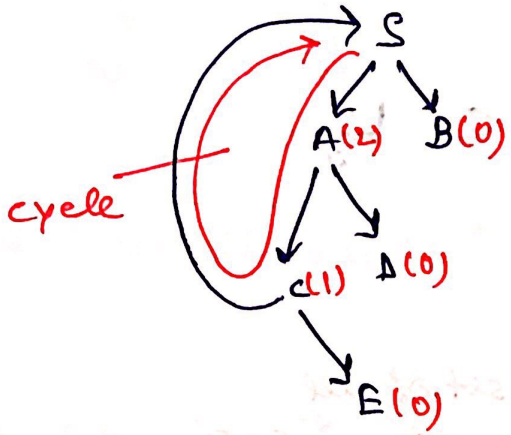
$\Rightarrow$ Rank of non-terminal :-

for a variable rank is the max distance from that variable to other.



Variable dependency graph

Since rank of every non-terminal is finite so & L(G) is finite.



rank of S is not finite

so, L(G) is infinite

* G is CFG without ϵ & unit production and G' is equivalent CNF, such that.

G → P (production rules)

→ T (terminals)

and G' → P' (production rules)

→ T (terminals)

so, $|P'| \leq (K-1) * |P| + |T|$

where K is length of longest string

eg:- $|P| = 10$, $|T| = 5$ and $K = 7$

so, $|P'| \leq (7-1) \times 10 + 5$

$|P'| \leq 65$

set of all CNF lang

set of all GNF lang

set of all CFG

$L_1 = L_2 \subset L_3$

also, $(L_1 = L_2)$ — set of all CFG $-\{\epsilon\}$

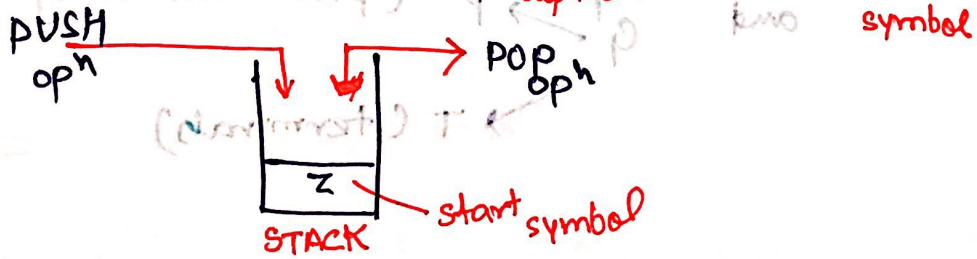
PDA - programming :-

7-tuple

Pushdown Automata $= M(Q, \Sigma, \delta, \Gamma, q_0, Z, F)$

Γ — STACK alphabet

Z — start stack symbol



	DPDA	NPDA
Choice	X	✓
Dead config	✓	✓
ϵ - mov	✓ restricted	✓ free

δ of PDA

DPDA

$\delta_{Q \times (\Sigma \cup \epsilon) \times \Gamma \to Q \times \Gamma^*}$

eg: $\delta(q_0, a, b) = \{q_1, ab\}$

a → i/p string

b → top stack var

NPDA

$\delta_{Q \times (\Sigma \cup \epsilon) \times \Gamma \to 2^{Q \times \Gamma^*}}$ (finite subset)

eg: $\delta(q_0, a, b) = \{(q_1, ab), (q_2, \epsilon)\}$

(PUSH, POP)

$(\delta(q_0, \epsilon, a) \neq \phi \to \delta(q_0, x, a) = \phi)$

PUSH 'a' $\delta(q_0, a, b) = (q_1, ab)$ $b \to ab$

POP 'a' $\delta(q_0, a, a) = (q_1, \epsilon)$ $a \to \epsilon$

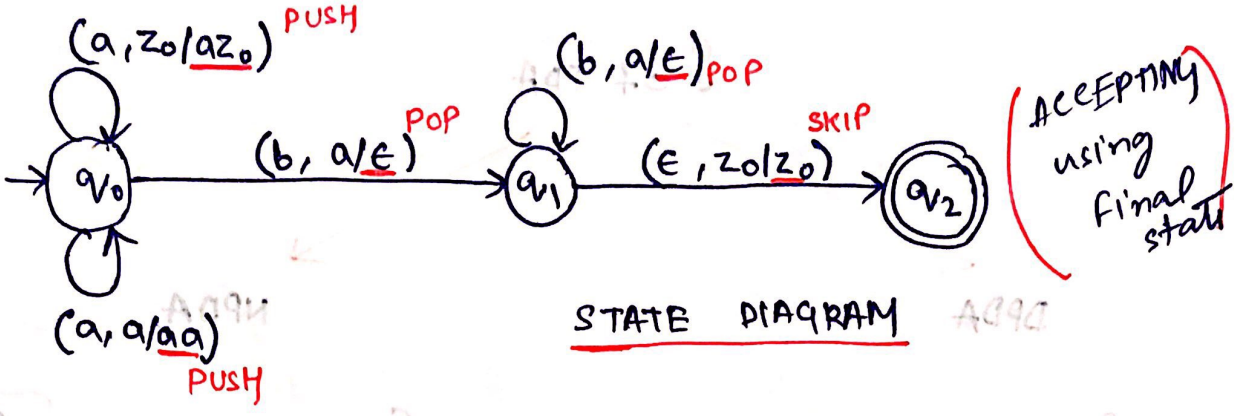
SKIP 'a' $\delta(q_0, a, b) = (q_0, b)$ $b \to b$

REPLACE $\delta(q_0, a, b) = (q_1, a)$ $b \to a$

⇒ COUNTING COMPARISION :-

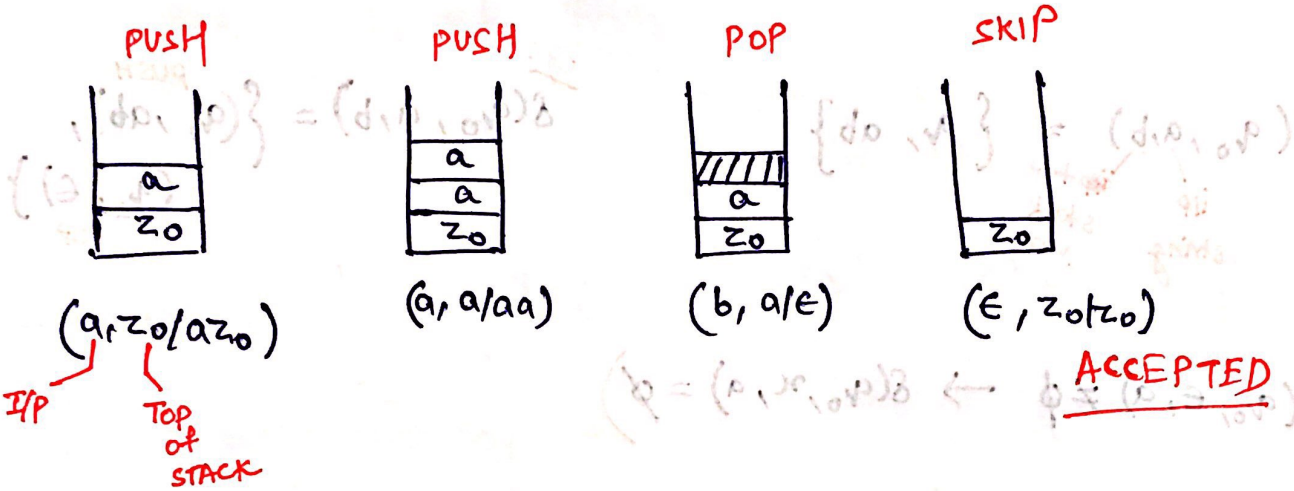
1. $L = \{a^n b^n \mid n \geq 1\}$

let STACK



STATE DIAGRAM

Let $\omega = aabb$



ACCEPTED

[ACCEPT : I/p = ε & TOS = z_0

REJECT : otherwise]

Transition func^n :-

$\delta(q_0, a, z_0) = (q_0, az_0)$

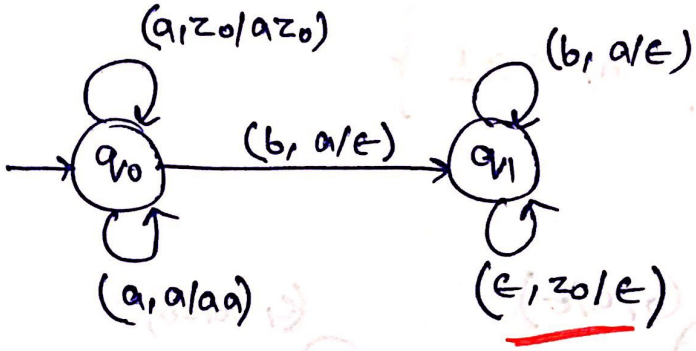
$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

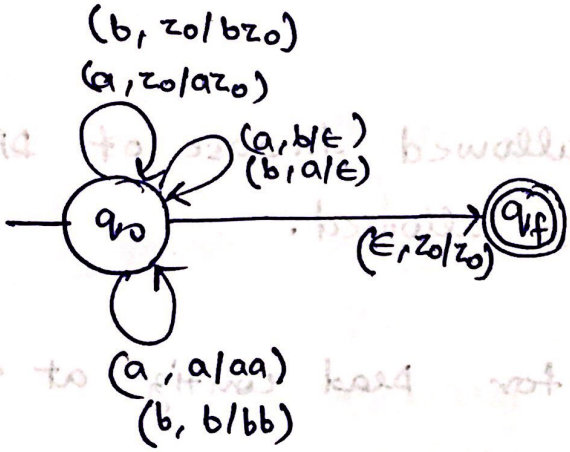
* $\delta(q_1, \epsilon, z_0) = (q_f, z_0)$ → Accept using Final state

* $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$ → Accept using Empty stack

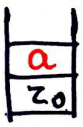


(ACCEPTING using Empty stack)

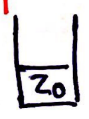
2. $L = \{ n_a(\omega) = n_b(\omega) \}$ Number of a's equals to b's



abbbaaa

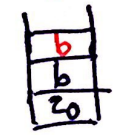


abbbaaa

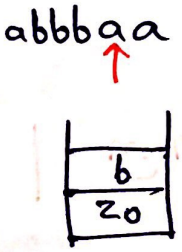


abbbaaa

abbbaaa

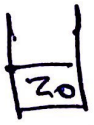


abbbaa



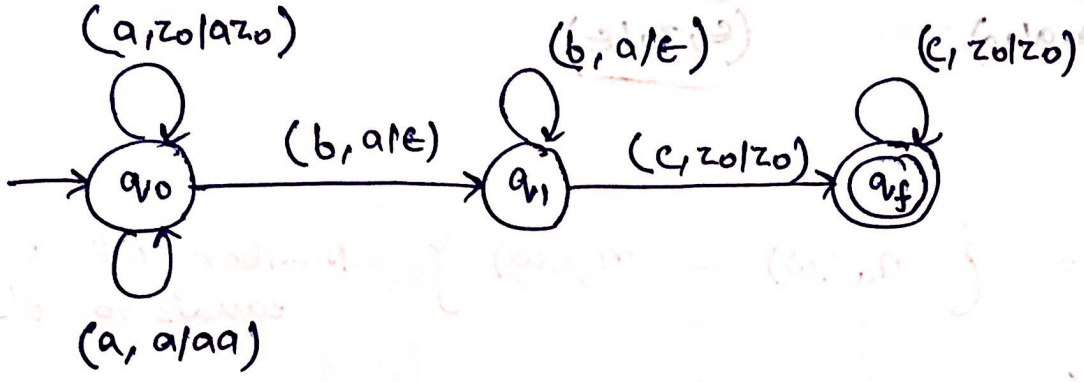
abbbaa

abbbaaϵ



ACCEPTED

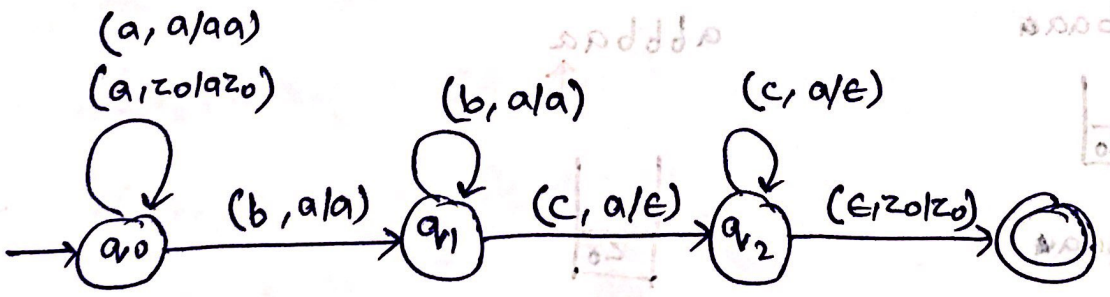
3. $L = \{ a^n b^n c^m \mid n, m \geq 1 \}$



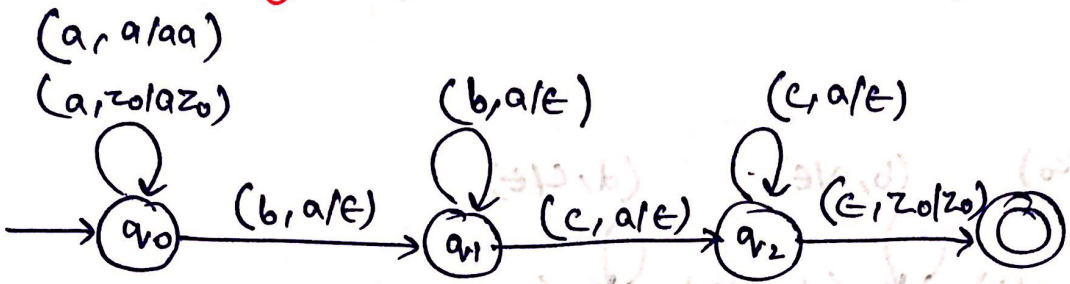
NOTE:- Dead config is allowed incase of DPDA whereas in DFA it is not allowed.

So, DPDA will hanged for Dead config at non-final state.

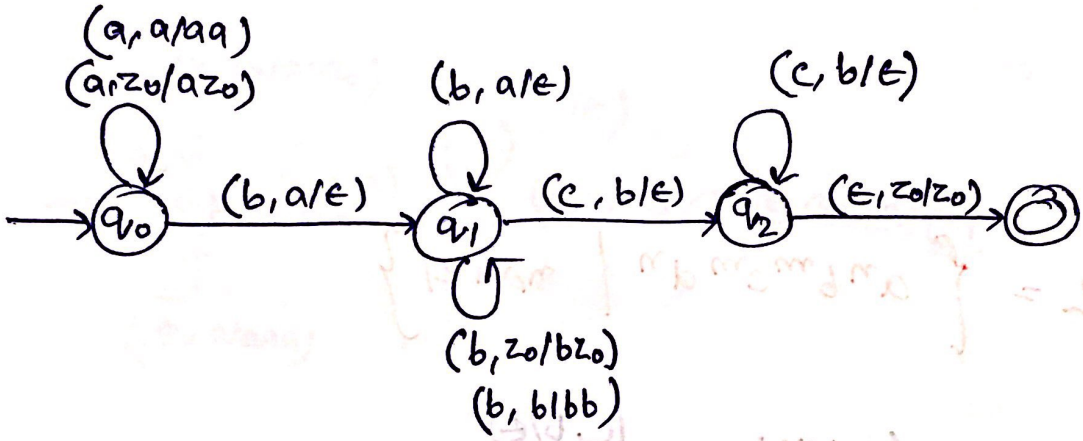
4. $L = \{ a^n b^m c^n \mid m, n \geq 1 \}$



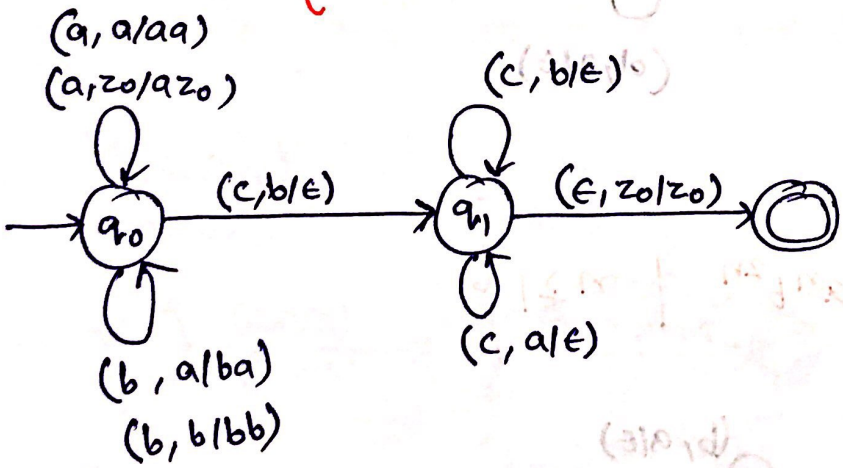
5. $L = \{ a^{m+n} b^m c^n \mid m, n \geq 1 \}$



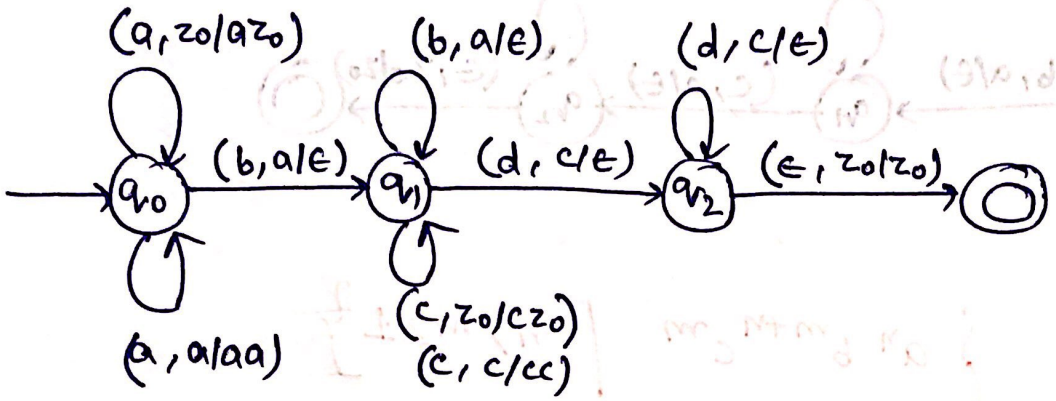
6. $L = \{ a^n b^{m+n} c^m \mid n, m \geq 1 \}$



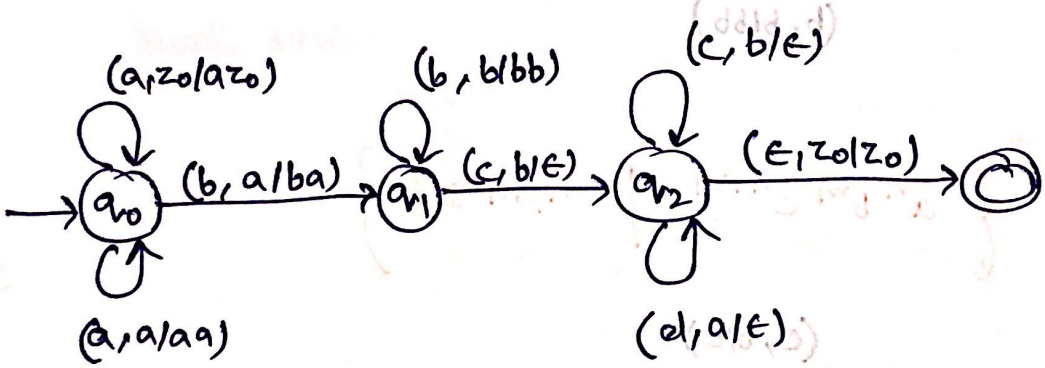
7. $L = \{ a^n b^m c^{n+m} \mid n, m \geq 1 \}$



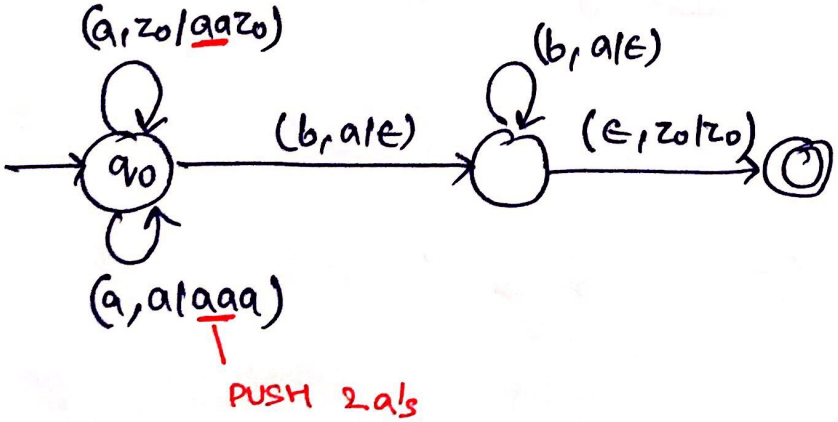
8. $L = \{ a^n b^n c^m d^m \mid n, m \geq 1 \}$



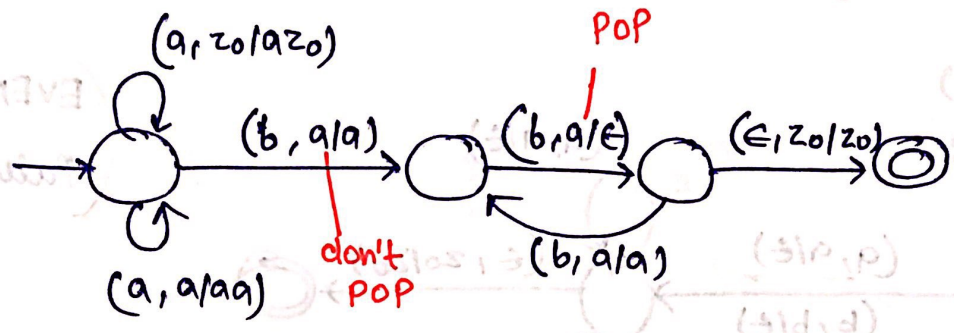
9. $L = \{ a^n b^m c^m d^n \mid n, m \geq 1 \}$



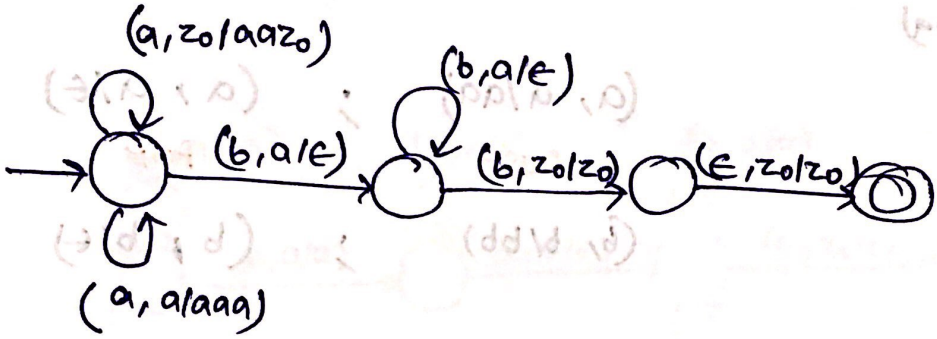
10. $L = \{ a^n b^{2n} \mid n \geq 1 \}$



OR

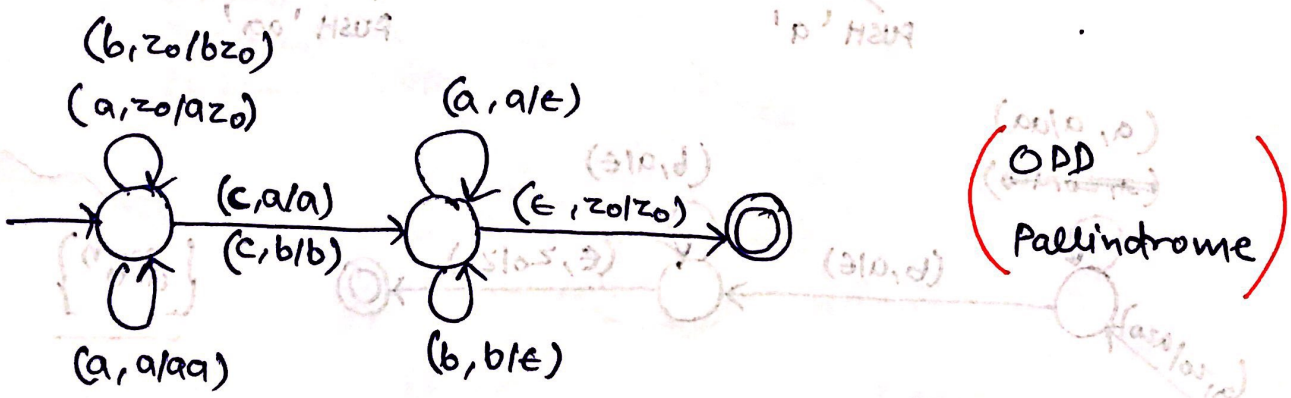


11. $L = \{ a^n b^{2n+1} \mid n \geq 1 \}$

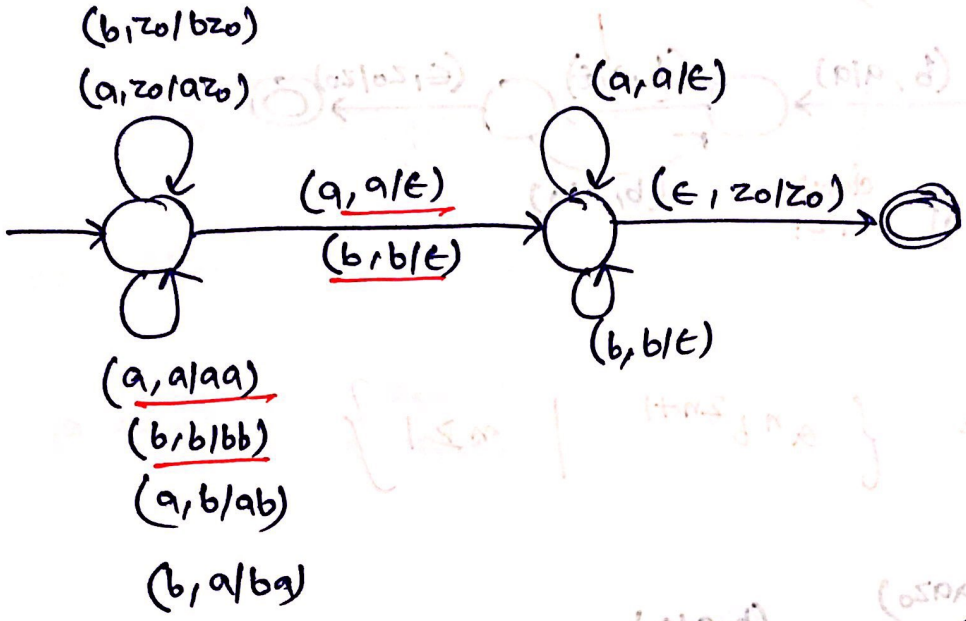


STRING MATCHING :-

12. $L = \{ \omega c \omega^R \mid \omega \in (a,b)^+ \}$



13. $L = \{ww^R \mid w \in (a,b)^+\}$



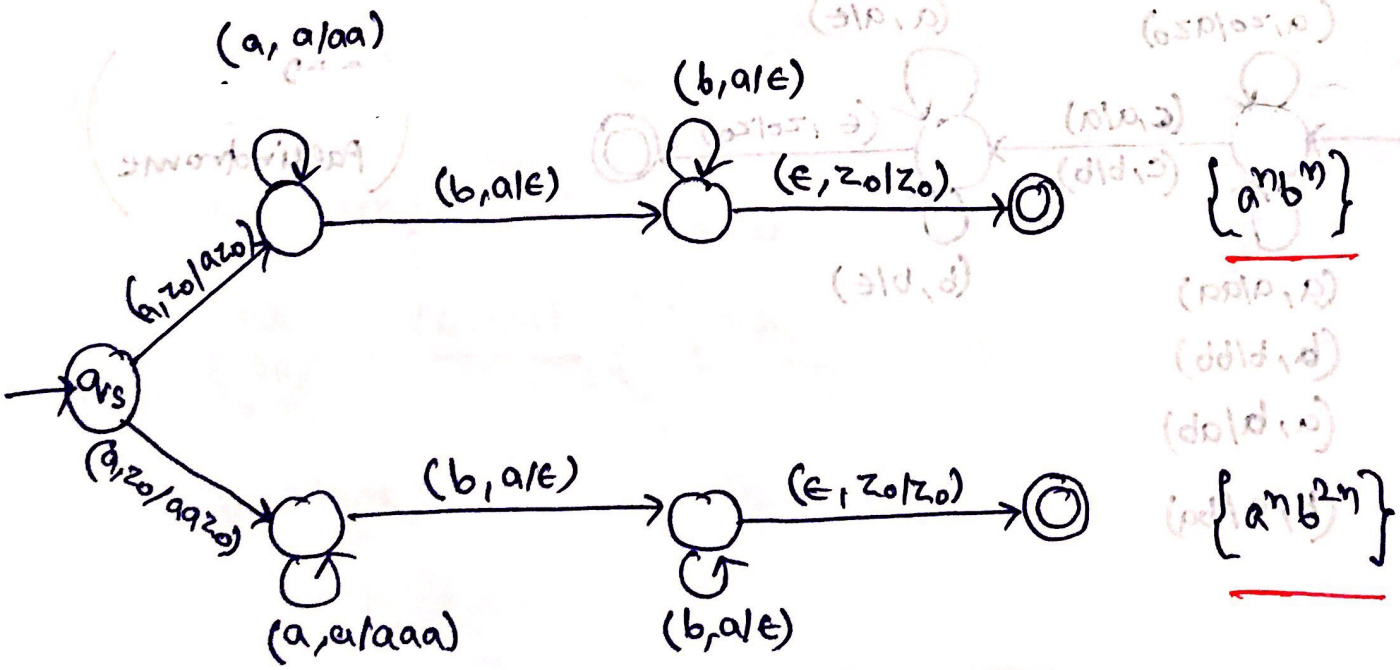
(EVEN Palindrome)

$(a, a/aa)$; $(a, a/\epsilon)$
PUSH POP
$(b, b/bb)$; $(b, b/\epsilon)$

This is NPDA (Non-deterministic)

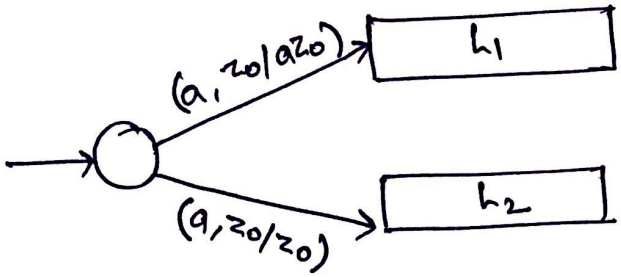
14. $L = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$

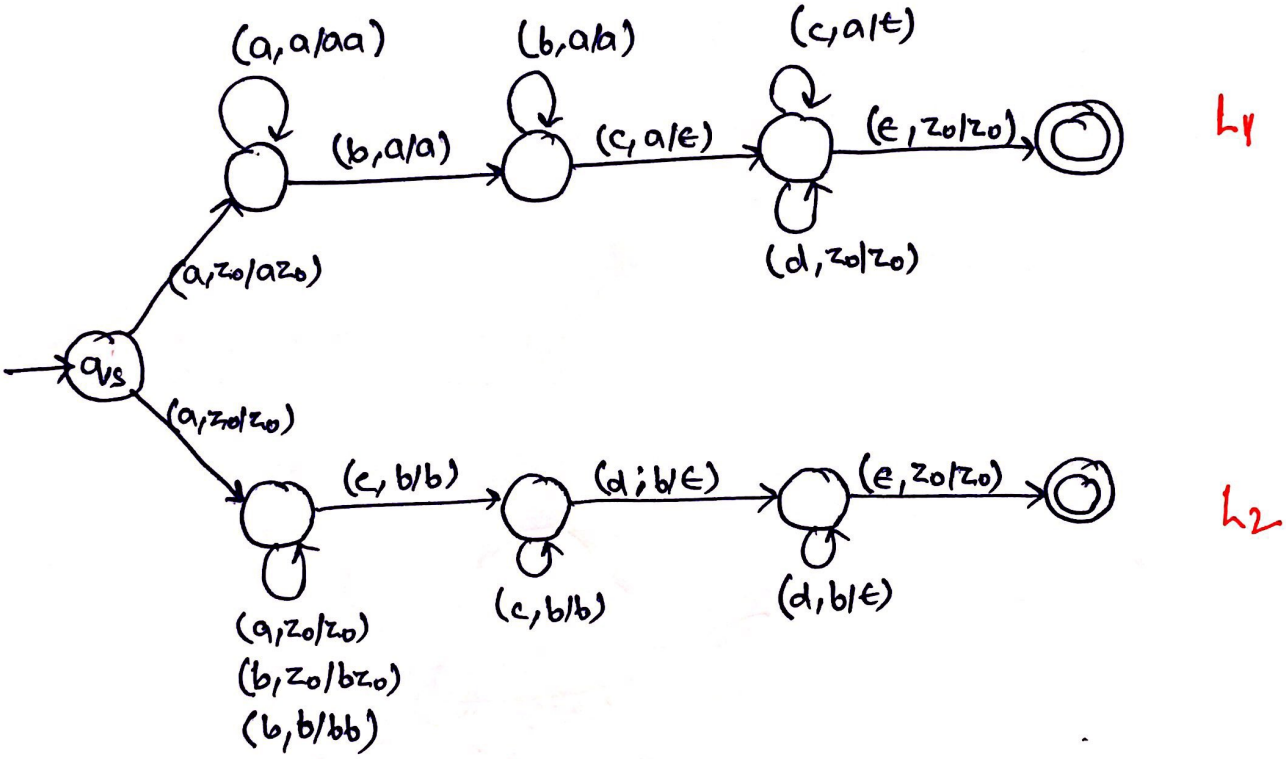
PUSH 'a' PUSH 'aa'



15. $L = \{ a^i b^j c^k d^l \mid i=k \text{ or } j=l \}$ — CFL

$$L = \underbrace{\{ a^m b^j c^m d^l \}}_{L_1} \cup \underbrace{\{ a^i b^m c^k d^m \}}_{L_2}$$





16. $L = \{ a^m b^n \mid m = 2n+1 \}$