Context Free Grammar

→ Type-2 Grammars means CFGs.

$A \rightarrow \alpha$,
$A \in V$
$\alpha \in (V \cup T)^*$

→ CFG.

→ Language are generated by CFGs are CFL.
→ CFL accepted by PDA (Push down automata).

Ex-

$A \rightarrow aAb / ab$.

→ classification of CFGs:

- (1) Ambiguous Grammar
- (2) Unambiguous grammar

- (1) DCFG (Deterministic context free grammar)
- (2) NDCFG (Non-)

- (1) LR (Left recursive grammar)
- (2) RR (Right recursive grammar)

• CFG simplification:
The four main steps will be followed in CFG simplicati
-on-
(i) Eliminate Ambiguity.
(ii) Elimination of ε production.
(iii) elimination of unit production.
(iv) elimination of useless symbols.

(11) Elimination of ϵ-production:

→ $\epsilon \in L(G)$, If ϵ is present in language then remove all the ϵ, except $S \to \epsilon$. [S is start symbol]

→ $\epsilon \notin L(G)$, If ϵ is not present in language then remove all the Null string (ϵ).

ϵx-1

$S \to aSb / aAb$

$A \to \epsilon$

Step-1: find out all the Null production. {A}

Step-2: find out nullable ~~production~~ variable. {A}

$\Rightarrow$ $S \to aSb / \cancel{aAb} / ab$

$\cancel{A \to \epsilon}$

$S \to aSb / ab$

ϵx-2

$S \to AB$

$A \to aAA / \epsilon$

$B \to bBB / \epsilon$

$S \to \epsilon$ means $\epsilon \notin \epsilon \cdot L(G)$

→ here, nullable variables are = {A, B, S}

$S \to AB / B / A / \epsilon$

$A \to aAA / aA / a$.

$B \to bBB / bB / b$

all Null string(ϵ) are removed ~~accepts~~ except $S \to \epsilon$, because $\epsilon \in L(G)$.

Ex-3

$S \rightarrow AbaC$

$A \rightarrow BC$

$B \rightarrow b/\epsilon$

$C \rightarrow D/\epsilon$

$D \rightarrow d$.

→ here, nullable ~~production~~ variable $= \{A, B, C\}$

$S \rightarrow \underline{A}ba\underline{C} / baC / Aba / ba$

$A \rightarrow \underline{B}\underline{C} / B / C$

$B \rightarrow \underline{b}$

$C \rightarrow \underline{D}$

$D \rightarrow \underline{d}$

→ due to B and C, A will be a nullable variable.

(iii) Elimination of unit production:

Ex-1

$S \rightarrow Aa / B$

$B \rightarrow A / bb$

$A \rightarrow a / bc / B$

→ s-1: Write the grammar without unit production

$S \rightarrow Aa / bb / a / bc$	$S \rightarrow \underset{\times}{B} \rightarrow bb$
$B \rightarrow bb / a / bc$	$S \rightarrow \underset{\times}{B} \rightarrow \underset{\times}{A} \rightarrow a$ / bc
$A \rightarrow a / bc / bb$	$B \rightarrow \underset{\times}{A} \rightarrow a$ / bc
	$B \rightarrow \underset{\times}{A} \rightarrow B$
	$A \rightarrow \underset{\times}{B} \rightarrow bb$

→ find out all the chain chains and whatever the result add them, then delete unit production.

Ex-2

$S \rightarrow AB$

$A \rightarrow a$

- $B \rightarrow C/b$
- $C \rightarrow D$
- $D \rightarrow E$

$E \rightarrow a$

→ While the grammar without unit production - then add the results.

$S \rightarrow AB$	
$A \rightarrow a$	$B \rightarrow C \rightarrow D \rightarrow E \rightarrow a$ (X)
$B \rightarrow b/a$	$C \rightarrow D \rightarrow E \rightarrow a$ (X)
$C \rightarrow a$	$D \rightarrow E \rightarrow a$ (X)
$D \rightarrow a$	
$E \rightarrow a$	

(IV) Elimination of useless symbols:

Ex-1

$S \rightarrow AB/a$

$A \rightarrow BC/b$

$xB \rightarrow aB/C$

$xC \rightarrow aC/B$

→ here, Terminals (T) = {a, b}

V = {S, A, B, C}

useful symbols = {a, b, S, A}

useless symbols = {B, C}

(B, C are not able to generate any particular string)

- Delete all useless symbols - ~~from~~

$S \rightarrow aA$

$\times$ $\boxed{A \rightarrow b}$ → delete 'A' because 'A' can't reachable in 'S'.

So, the grammar is, $\boxed{S \rightarrow a}$

Ex-2

$S \rightarrow AB / AC$.

$A \rightarrow aAb / bAa / a$.

$B \rightarrow bbA / aaB / AB$.

$\times C \rightarrow abCA / aDb$.

$\times D \rightarrow bD / aC$.

→ $T = \{a, b\}$

$V = \{S, A, B, C, D\}$.

→ useful symbols = { we can derive some Terminals or some strings }

$= \{a, b, A, B, S\}$

→ Eliminate C and D because they not deriving any ~~string~~ string of terminals.

$S \rightarrow AB$

$A \rightarrow aAb / bAa / a$

$B \rightarrow bbA / aaB / AB$

→ final grammar after removing useless symbols.

Ex-3

$S \rightarrow ABC / BaB$

$A \rightarrow aA / BaC / aaa$

$B \rightarrow bBb / a$

$C \rightarrow CA / AC$

→

$T = \{a, b\}$

$V = \{S, A, B, C\}$

useful symbols $= \{a, b, A, B, S\}$

so, ~~finite~~ grammar;

$S \rightarrow BaB$

$A \rightarrow aA / aaa$

$B \rightarrow bBb / a$

If here, A not reachable 'S', so deleted.

so, final grammar are,

$S \rightarrow BaB$

$B \rightarrow bBb / a$

• Normal Forms of CFGs :

→ To convert a CFG into normal form, we start by trying to eliminate null productions of the form $A \to \epsilon$ and the unit productions of the form $B \to C$.

There are two normal forms -

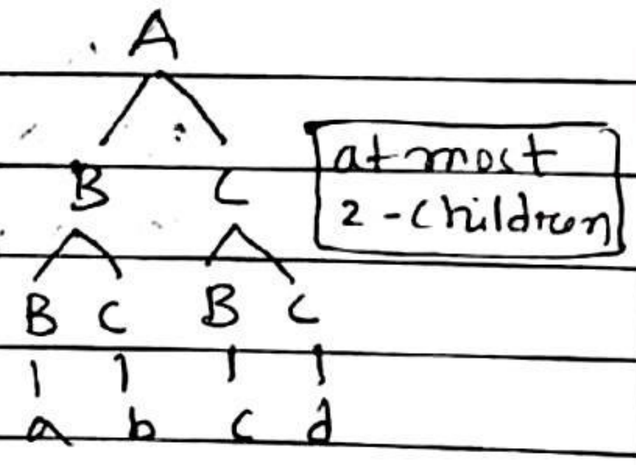
(1) Chomsky normal form (CNF)
(2) Greibach normal form (GNF).

(1) **Chomsky normal form (CNF)**: If all the productions are of form $A \to BC$ or $A \to a$ where, A, B, C are variables and 'a' is a terminal.

Adv :-

→ Length of the each production is restricted.

→ Derivation tree or parse tree obtained from CFG CNF is always binary tree.



→ The number of steps required to derive a string of length $|w|$ is $(2|w|-1)$

Ex- String length '3' = {abb}.

$S \to AB$	$S \to AB$	$2*3-1$
$A \to a$	$\to ABB$	$= 5$ (steps)
$B \to b$	$\to aBB$	5
$A \to AB$	$\to abB$	
	$\to abb$	

→ It is easy to apply CYK membership algorithm.

Q-1 Convert the following into CNF:

$S \rightarrow bA/aB$

$A \rightarrow bAA/aS/a$

$B \rightarrow aBB/bS/b$

$A \rightarrow BC$ $A \rightarrow a$: CNF

→

$S \rightarrow N_bA/N_aB$

$A \rightarrow N_b(AA)/N_aS/a$

$B \rightarrow N_a(BB)/N_bS/b$

$\begin{cases} N_a \rightarrow a \\ N_b \rightarrow b \end{cases}$

→ $S \rightarrow N_bA/N_bB$
$A \rightarrow N_bS/N_aS/b$
$B \rightarrow N_aT/N_bS/b$
$N_a \rightarrow a$
$N_b \rightarrow b$
$\begin{cases} S \rightarrow AA \\ T \rightarrow BB \end{cases}$
} - Converted

• CYK Algorithm:

→ CYK Algorithm only apply for CNF.

Q.1 can be apply

Check whether the string 'baaba' is a valid member of the following CFG.

$S \rightarrow AB/BC$
$A \rightarrow BA/a$
$B \rightarrow CC/b$
$C \rightarrow AB/a$

→ 1 2 3 4 5
baaba → 5 length string

	5	4	3	2	1
1	15 A,S,C	14 ∅	13 ∅	12 A,S	11 B
2	2,5 S,A,C	24 B	2,3 B	22 A,C	
3	35 B	34 S,C	3,3 A,C		
4	45 A,S	44 B			
5	55 A,C				

→ String are generated.
(final cell contain 'S')
(15)

(12) = (11)(22)
= {B}{A,C}
= {BA, BC}
A S

(23) = (22)(33)
= (A,C)(A,C)
= {AA, AC, CA, CC}
X X X B

(34) = (33)(44)
= (A,C)(B)
= (AB, CB)
S,C X

(45) = (44)(55)
= (B)(A,C)
= (BA; BC)
A S

(13) = 1 to 3
= (11)(23)
= (B, B)
X X

= (12)(33)
= (A,S)(A,C)
= (AA, AC, SA, SC)
X X X X

(24) = (2,2)(3,4)
= (A,C)(S,C)
= {AS, AC, CS, CC}
X X X B

= (23)(24)
= {B}{B}
= {BB}
X

(35) = 3 to 5
= (33)(45)
= (A,C)(A,S)
= (AA, AS, CA, CS)
X X X X

= (3,A)(5,5)
= (5,C)(A,C)
= {SA, SC, CA, CC}
X X X B

(14) = 1 to 4
= 1 2 3 4

= (11)(24)
= (B)(B)
= (BB)
X

(12)(34)
= (A,S)(S,C)
= (AS, AC, SS, SC)
X X X X

(13)(44)
= (∅)(B)
= ∅

(25) = 2 3 4 5

= (22)(35)
= (A,C)(B)
= (AB, CB)
(S,C) X

= (23)(45)
= (B)(A,S)
= (BA, BS)
A X

= (24)(55)
= (B)(A,C)
= (BA, BC)
A X

(15) = 1 2 3 4 5

= (11)(25)
= (B)(S,A,C)
= (BS, BA, BC)
X A S

(12)(B5)
= (A,S)(B)
= (AB, SB)
S,C X

(13)(45)
= ∅ (45)
X

(14)(55)
= ∅ ∝ (55)
X

atlantis
Date
Page

Q-1

(*) ~~How~~ What are various substring of the given string 'baaba' that can be generated by above CFG's.

"1 2 3 4 5"
b a a b a

→ Table
- using CYK Algo can be ~~done~~ solve this question:

→ (12), (25), (34), (45)

→ {ba, aaba, ab, ba}

This 4 string can generated by given CFG's.

→ The time complexity $O(n^3)$.

Q-2

check whether the string "aabbb" is a valid ~~member~~ of following grammar or not.

S → AB
A → BB/a
B → AB/b

→

(13) = (11) (23)
= A . (S, B)
= (AS, AB)
S,B
(1,3) = (1,2) (33)
= (B)(BB)
= ∅

	1	2	3	4	5
	a	a	b	b	b
5	15 S,B	14 A	13 S,B	12 ∅	11 A
4	25 S,B	24 A	23 S,B	22 A	
3	35 S,B	34 A	33 B		
2	45 A	44 B			
1	55 B				

(12) = (11) (22)
= (A) (A)
= (AA)

(45) = (44) (55)
= B B

23 = (22) (33)
= (A) (B)
= AB

(24) = (22) (34) | (23) (44)
= (A) (A) | = (S,B) (B)
= (AA) | = {SB, BB}
A

34 = (33) (44)
B B
= BB

3 4 5

$(35) = (33)(45)$	$(34)(55)$	
$= (B)A(A)$	$(A)(B)$	
$= (BA)$ ✗	$= (AB)$ S, B	

$(14) = 1\ 2\ 3\ 4$

$= (11)(24)$	$(12)(34)$	$(23)(44)$
	$\emptyset$	$= (B, B)(B)$
$= (A)(A)$	$= X$	$= (SB, BB)$
$= (AA)$ ✗		✗ $\overline{A}$

$(25) = 2\ 3\ 4\ 5$

$= (22)(35)$	$= (23)(45)$	$(24)(55)$
$= (A)(S, B)$	$= (S, B)(A)$	$= (A)(B)$
$= (AS, AB)$ ✗ (S, B)	$= (SA, BA)$ ✗ ✗	$= (AB)$ ✗ (S, B)

$(15) = (11)(25)$

$= (A)(S, B)$	$(12)(35)$	$(13)(45)$	$(14)(55)$
$= (SS, SB, BS, BB)$	$= \emptyset$	$= (S, B)A$	$= (A)(B)$
✗ ✗ ✗ ✗		$= (SA, BA)$	$= (AB)$
$= (AS, AB)$ A		✗ ✗	A S, B
✗ (S, B)			

→ The Given string "aabbb" are valid member.

(*) What is the substring of 'aabbb' are generated by given Grammar.

(1-3) (2-5) (2-3) (3-5)

→ aab, abbb, ab, bbb.

(2) Greibach Normal form (GNF) :

→ If all the productions are of form $A \rightarrow a\alpha$ where $\alpha \in V^*$, then it is called GNF.

V = Variable (non-terminals)

Ex- $A \rightarrow aBCDE$.

$A \rightarrow a$.

Adv:

→ The number of steps required to generate a string of length $|w|$ is $|w|$.

$A \rightarrow aB$
$B \rightarrow b$

$A \Rightarrow aB$
$\Rightarrow ab$ } 2

to generat 'ab'
String
$|w|=2$

→ GNF is useful in order to convert a CFG to PDA.

- Conversion of CFG to PDA:

→ push the start symbol 'A' on the stack.

$$\delta(q_0, \epsilon, Z_0) = (q_1, AZ_0)$$

→ Push RHS of A as follows

$$\delta(q_1, a, A) = (q_1, \alpha)$$

if $A \rightarrow a\alpha$ is in 'G'

$$\delta(q_1, b, A) = (q_1, \beta)$$

if $A \rightarrow b\beta$ is in 'G'.

→ Add final state with

$$\delta(q_1, \epsilon, Z_0) = (q_f, \epsilon)$$

Not important to gate

Ex-1 Convert following in to PDA :

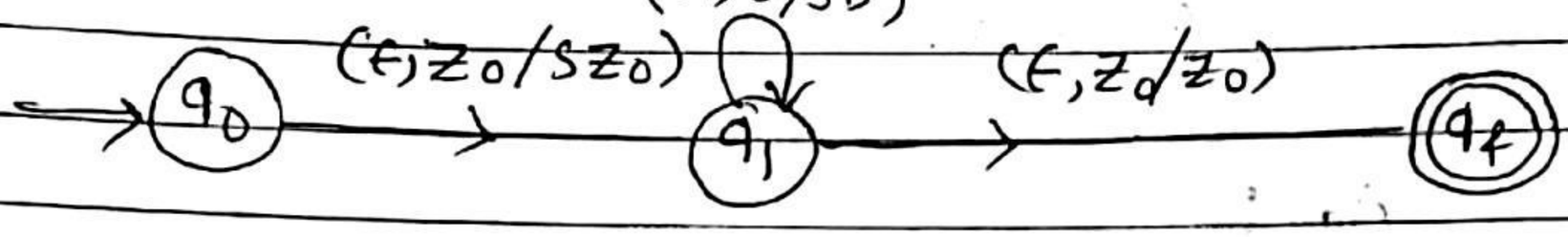
$S \rightarrow aSb/ab$

→ Conversion ~~GNF~~ into GNF grammar.

GNF $\begin{cases} S \rightarrow aSB/aB \\ B \rightarrow b \end{cases}$ $(b, B/\epsilon)$

$(a, S/BB)$

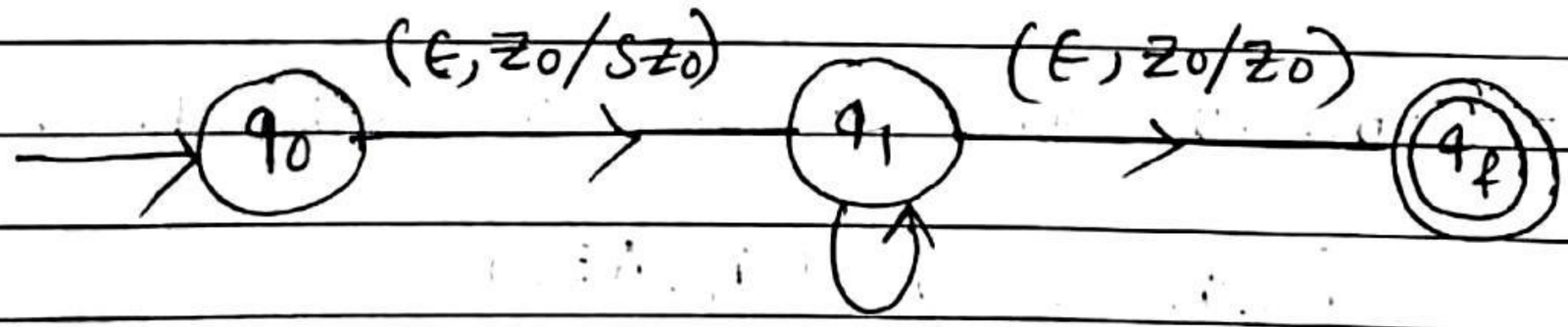
$(a, S/SB)$



Ex-2 convert following in to PDA –

$S \rightarrow aAA$
$A \rightarrow aS/bS/a$ } GNF (greibach)

→ $S \rightarrow aAA$ ✓
$A \rightarrow aS/bs/a$ ✓



$(a, A/aS)$

$(b, A/S)$

$(a, A/\epsilon)$

PDA = (FA + Stack)

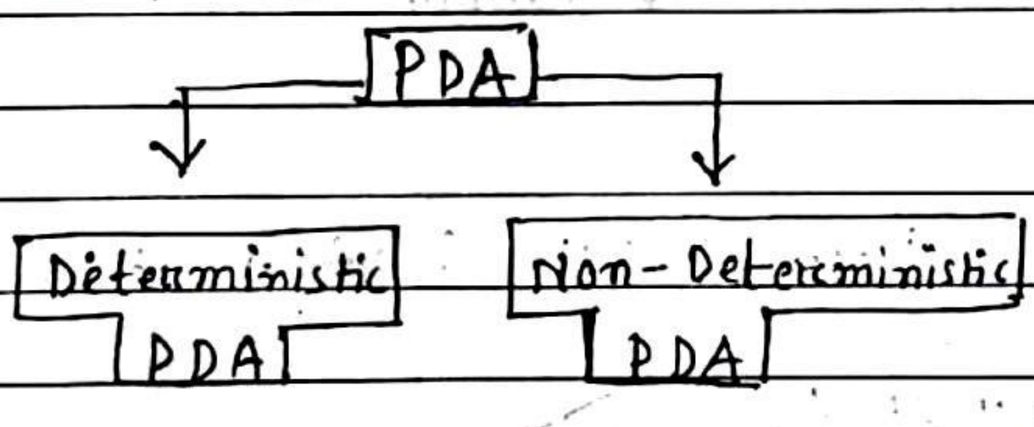
PUSHDOWN AUTOMATA (PDA):

→ A pushdown Automata (PDA) is essentially an NFA with stack.

→ A pushdown automata (PDA) is defined as 7-tuples.
$(Q, \Sigma, \delta, q_0, Z_0, F, \Gamma)$

Where,

Q → Finit set of states.
Σ → input Symbol.
δ → Transition function.
q_0 → Initial state.
Z_0 → Bottom of the stack
F → set of final states.
Γ → stack alphabet.



• For Deterministic PDA -

$$\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma \longrightarrow Q \times \Gamma^*$$

• For non-Deterministic PDA -

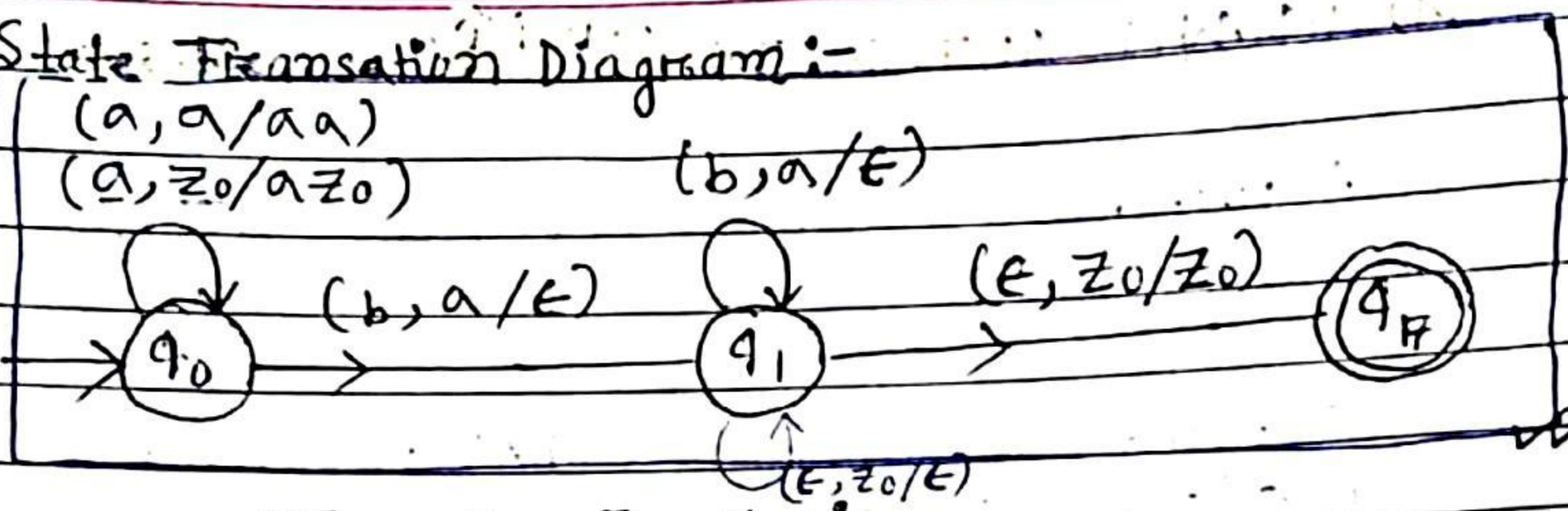
$$\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma \longrightarrow 2^{(Q \times \Gamma^*)}$$

• Ex-1 Construct a D-PDA that accept $\{a^n b^n / n \geq 1\}$ language.

→ L = $\{a^n b^n / n \geq 1\}$

"aabbϵ"

State Transation Diagram :-



Transition function :-

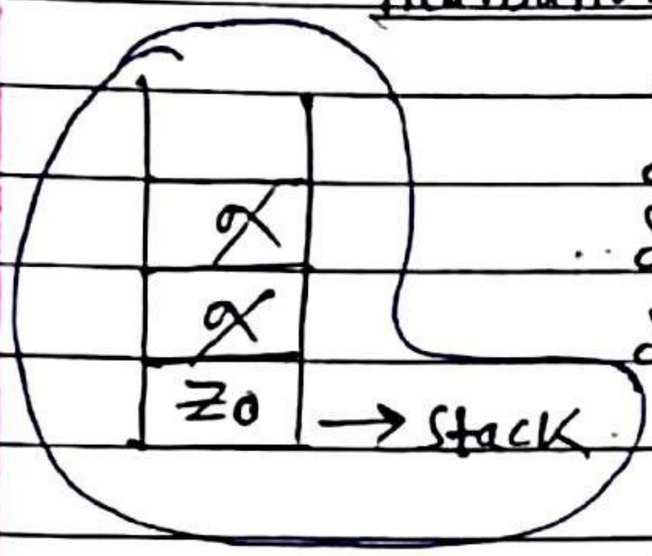
$\delta(q_0, a, z_0) = \delta(q_0, az_0)$

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

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

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

$\delta(q_1, \epsilon, z_0) = \delta(q_f, z_0)$ or $\delta(q_1, \epsilon)$



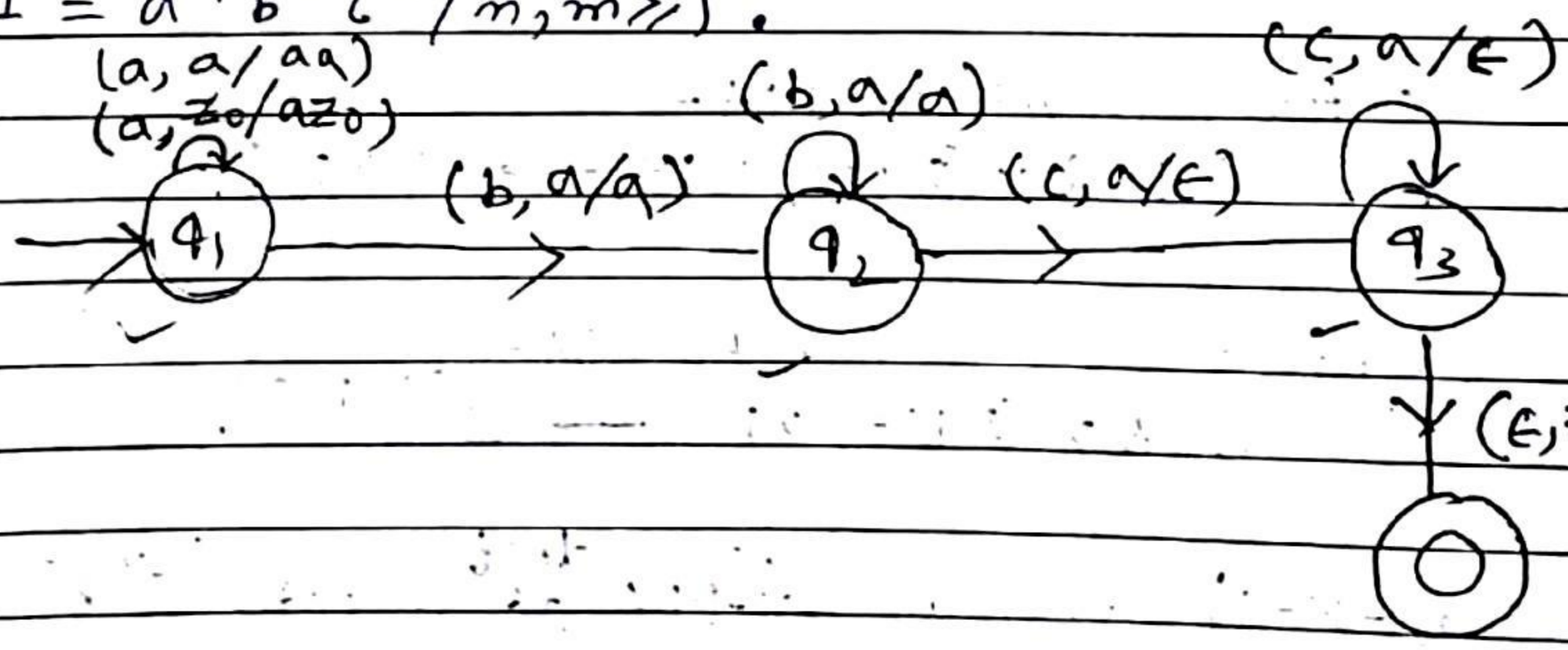
→ ~~PDA~~ In PDA acceptance of language are two type -

(I) acceptance by final state. } -
(II) acceptance by Empty stack. } equal in power.

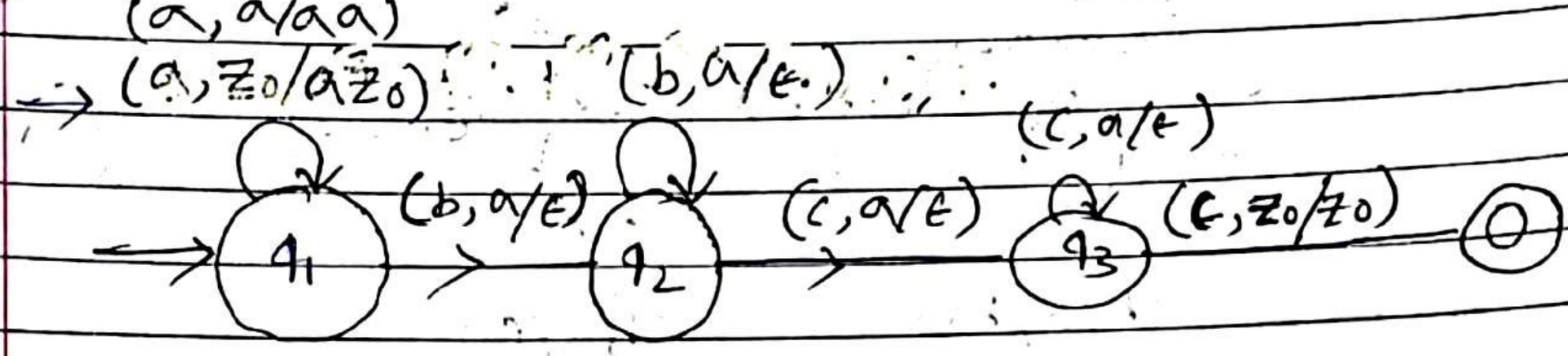
Ex-2

Construct a Deterministic PDA that accepts language

$L = a^n b^m c^n / n, m \geq 1)$.

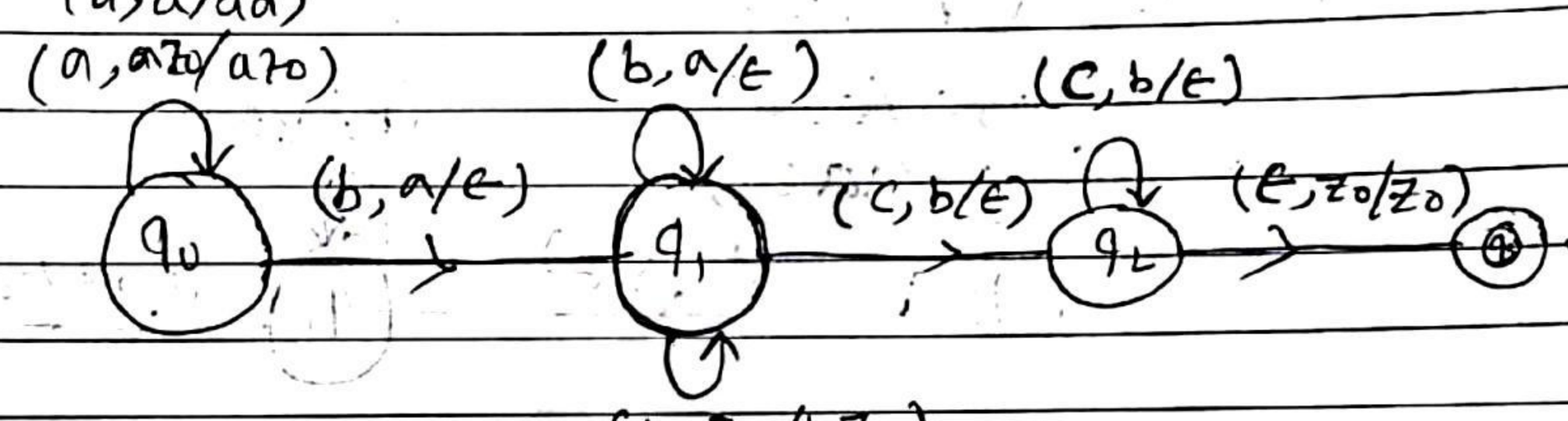


Ex-3 Construct an PDA that accepts $a^{m+n} b^m c^n / m,n \geq 1$.

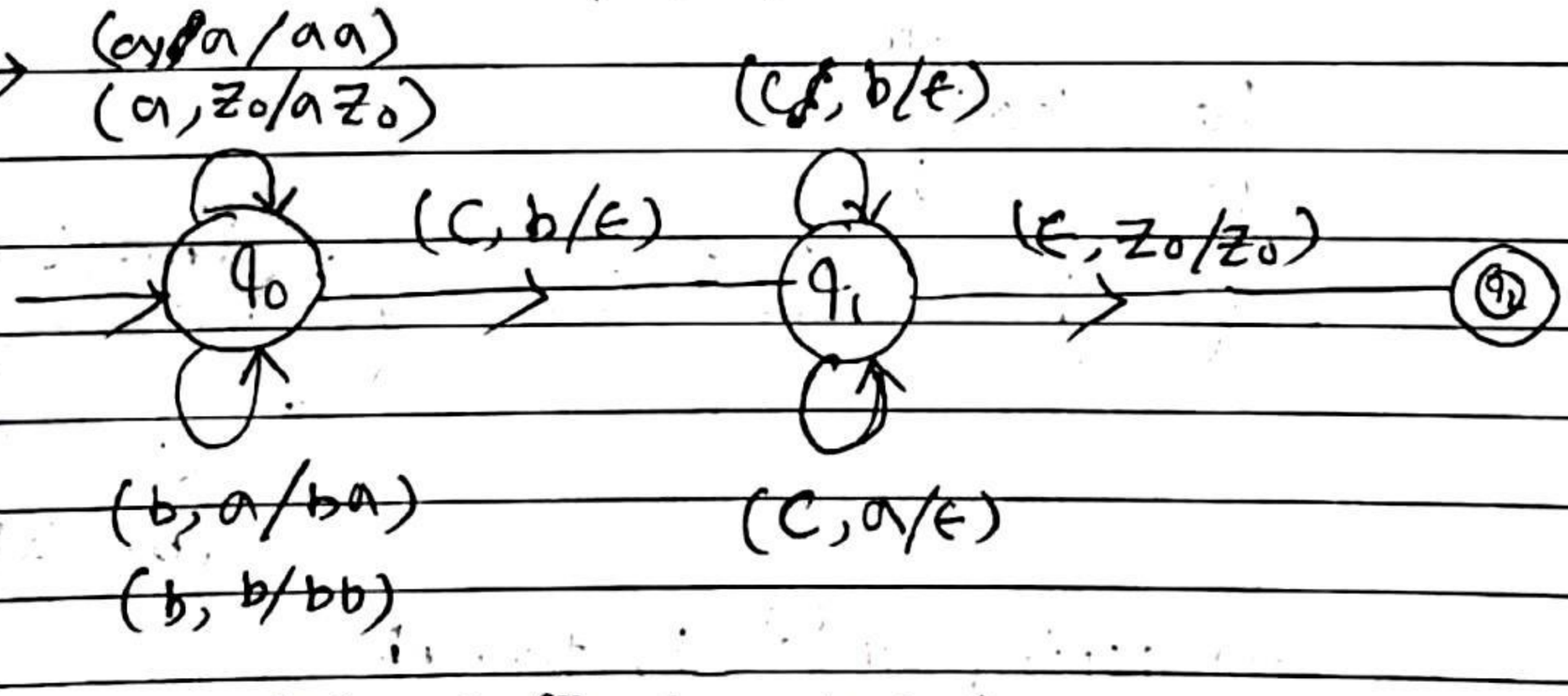


Ex-4 $a^n b^{m+n} c^n / n,m \geq 1$.

→ $L = a^n b^n b^m c^n / n,m \geq 1$.

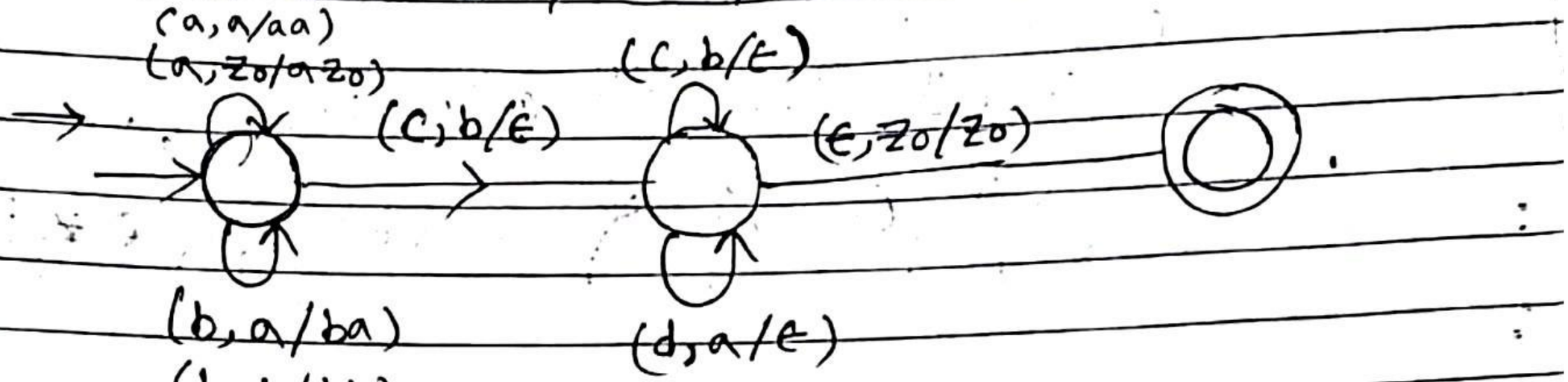


Ex-5 $a^n b^m c^{n+m} / n,m \geq 1$



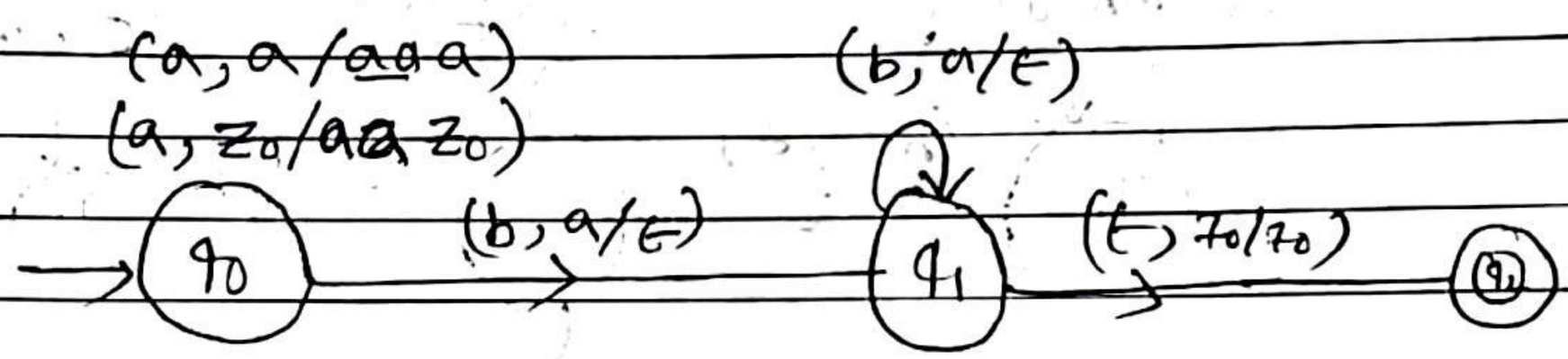
Ex-6 $a^n b^n c^m b^m / n,m \geq 1$

✓ Ex-7 $a^n b^m c^m d^n : / n, m \geq 1$.

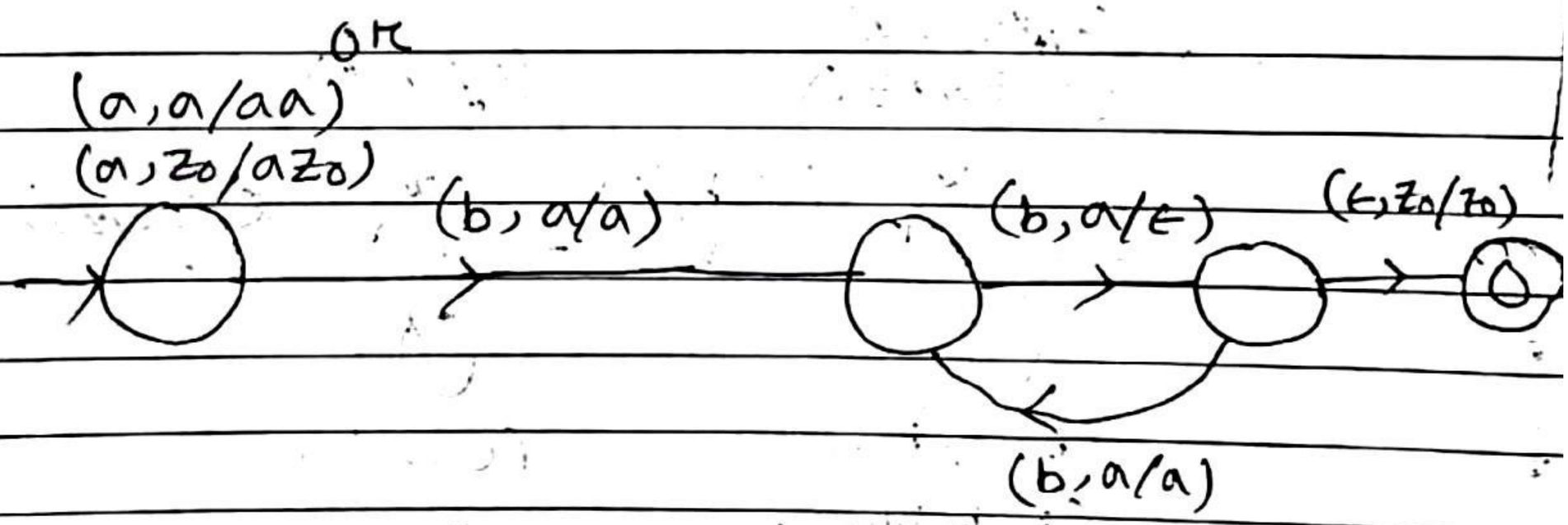


X Ex-8 $a^n b^m c^n d^m / n, m \geq 1$. → This language is not CFL.

Ex-9 $a^n b^{2n} / n \geq 1$

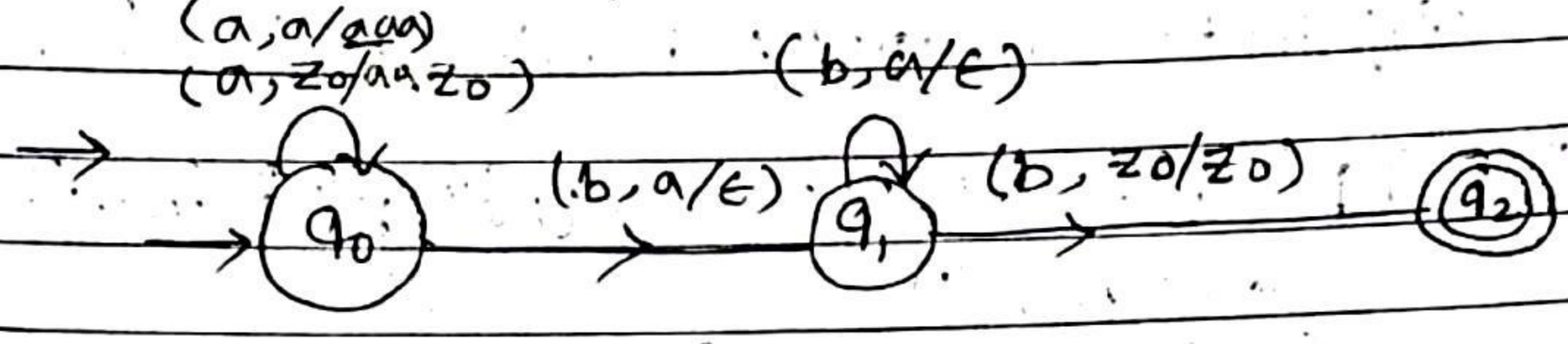


→ for every 'a' (Input) pushing 2 'aa' to the stack.



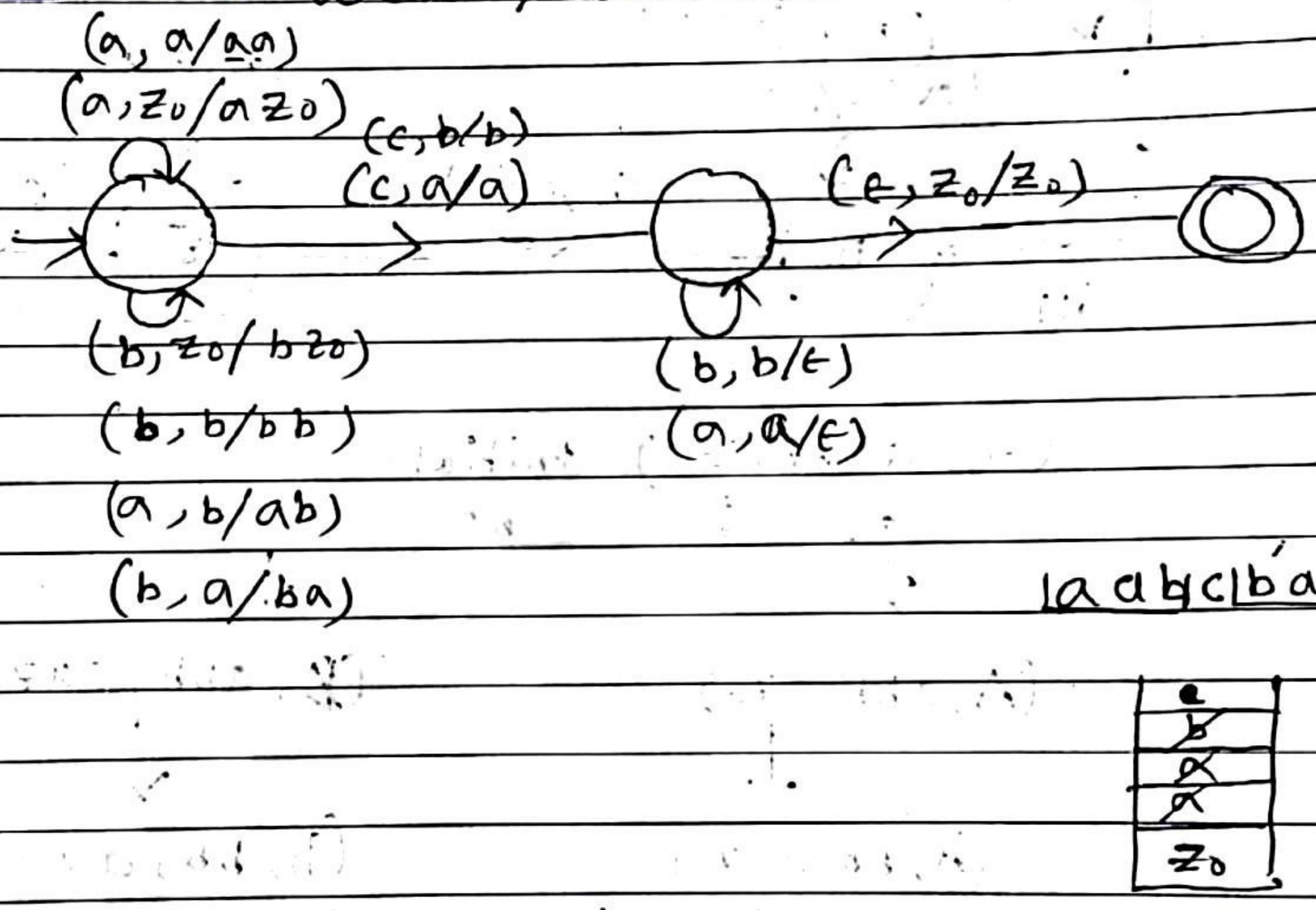
→ poping 1 'a' by two 'bb'

✓ **Ex-10** $a^n b^{2n+1} / n \geq 1$

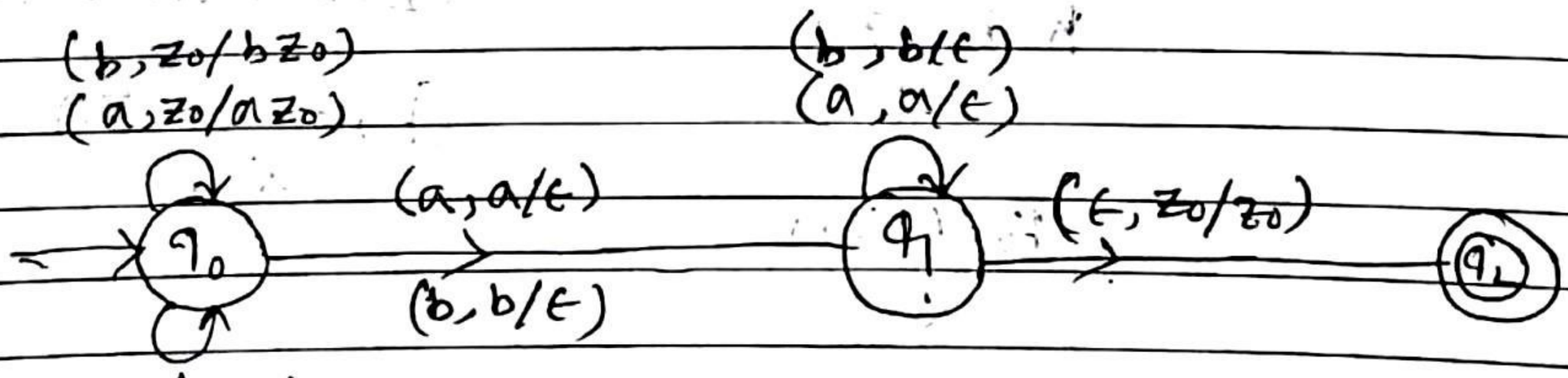


✗ **Ex-11** $a^n b^n c^n / n \geq 1$ → not CFL.

✓ **Ex-12** ~~$a^n b$~~ $wcw^R / w \in (a,b)^+$



NDPDA **Ex-13** $ww^R / w \in (a,b)^+$ (N-D PDA)

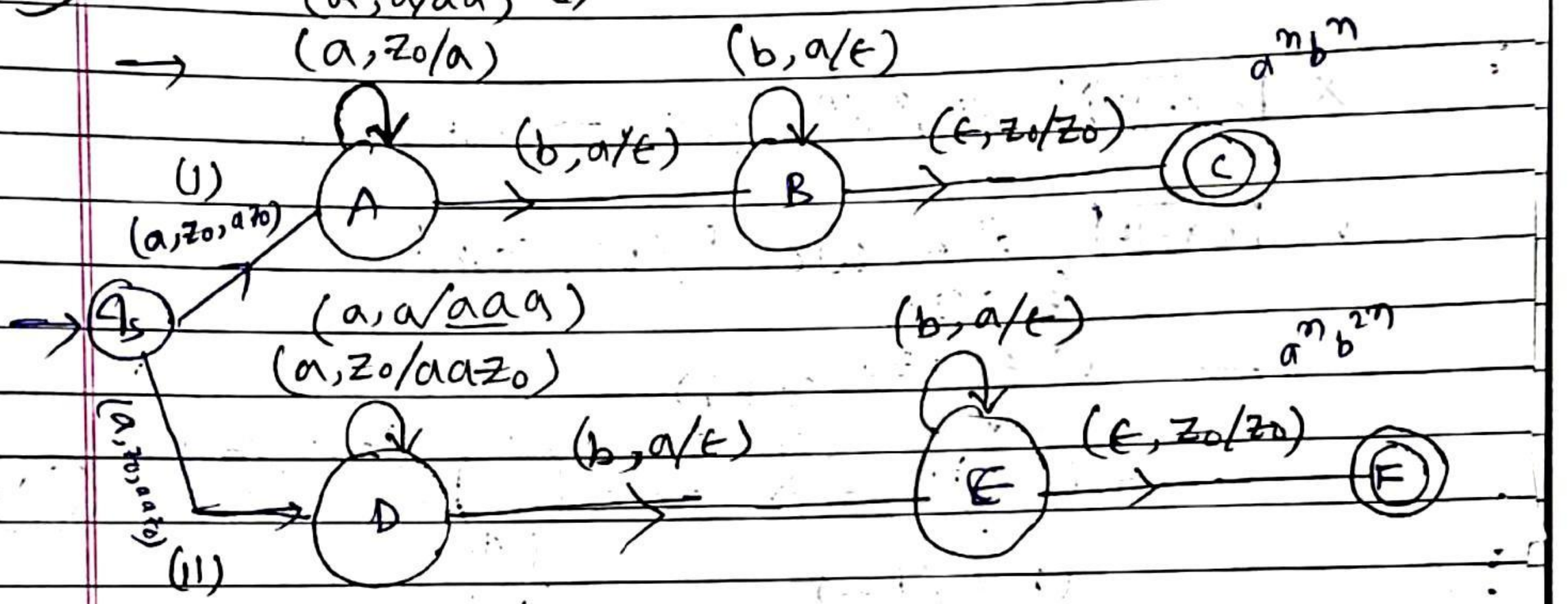


→ $ww^R / w \in (a,b)^*$ this language accepted by only NDPDA. so NDPDA is more powerful than DPDA.

ND-PDA

Ex-13 $L = \{a^n b^n / n \geq 1\} \cup \{a^n b^{2n} / n \geq 1\}$.

(i) $a^n b^n$ (ii) $a^n b^{2n}$



(q0, aabb, z0) initial.

$a^n b^n$ | $a^n b^{2n}$

(A, abb, az0)

(A, bb, aaz0)

(B, b, az0)

(B, ε, z0)

accept.

(D, abb, aaz0)

(D, bb, aaaaz0)

(E, b, aaaz0)

(E, ε, aaz0)

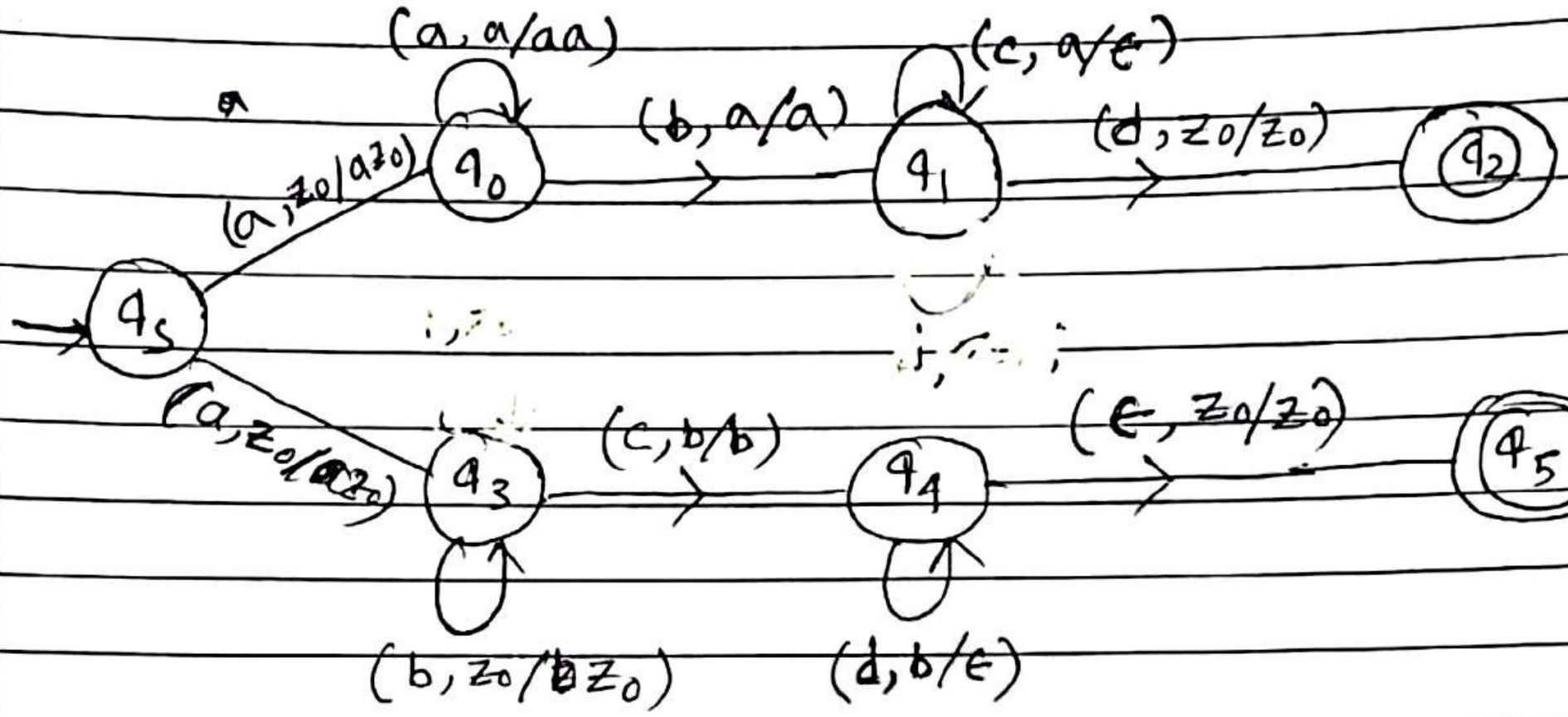
X

→ If atleast 1 PDA reach in final state then it is accepted.

Ex-14

$$L = \{a^i b^j c^k d^l / i = k \text{ or } j = l\}$$

$$= \{a^m b^j c^m d^l\} \cup \{a^i b^m c^k d^m\}$$



*. C programming language is as big as CFL.
CFL is enough to deccribe C language.

atlantis
Date _
Page _

- properties of CFL:

→ CFL are closed under following operations -

1. Union
2. Contatination
3. Kleene closure.

→ CFL are not closed under Intersection and Complementation -

(1) Union: $(L_1 \cup L_2)$

$L_1 \cup L_2$

$\downarrow$ $\downarrow$

CFG_1 CFG_2

$\downarrow$ $\downarrow$

$S_1 \rightarrow$ $S_2 \rightarrow$

$S \rightarrow S_1 / S_2$

$S_1 \rightarrow$

$S_2 \rightarrow$

(2) Contatination: $(L_1 \cdot L_2)$

$S \rightarrow S_1 \cdot S_2$

$S_1 \rightarrow$

$S_2 \rightarrow$

(3) Kleene closure: (L_1^*).

L_1^*

$\downarrow$

S_1

$(S \rightarrow S_1 S / \epsilon)$

→ not closed under intersection: $(L_1 \cap L_2)$

$L_1 = \{a^n b^n c^m / n, m \geq 0\}$

$L_2 = \{a^m b^n c^n / m, n \geq 0\}$

$L_1 \cap L_2 = \{a^n b^n c^n / n \geq 0\} \longrightarrow$ CFL (x)

→ not closed under ~~intersection~~ Complement :- $(\overline{L_1})_2$.

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

↓

CFL

• Decidable problems on CFL's :-

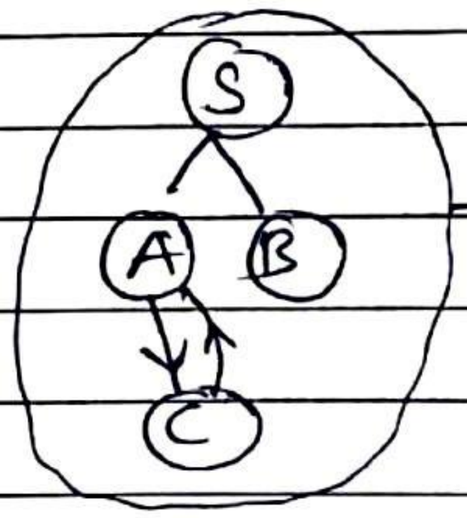
1. Membership problem. ($w \in L$?)
   → ~~To check whether~~ a string belongs to a given grammar or not. (CYK algo)

2. Emptiness problem.
   → to check the language L are empty or not.
   - CFG
   - ↓
   - simplify
   - ↓
   - $S \rightarrow$

   [after simplify the CFG, if the start symbol 'S' is useless, then the 'L' are empty]

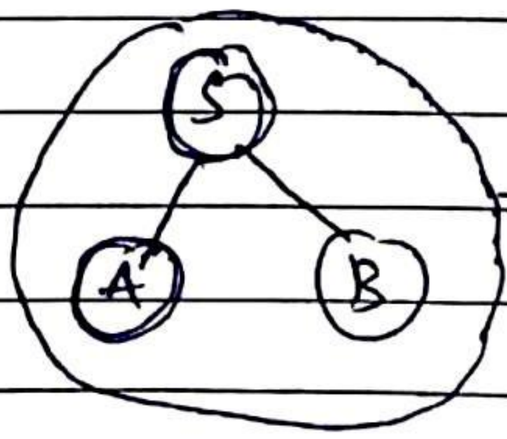
3. Finiteness.
   → CFG
   ↓
   simplify
   ↓

$S \rightarrow AB$
$A \rightarrow aC/a$
$C \rightarrow aA/b$
$B \rightarrow a$



→ for cycle it is infinite.

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b$



→ no-cycle it is finite

Context Free Language.

- Deterministic CFLs (DCFLs) -

1. $\{a^{m+n} b^n c^m / n, m \geq 1\}$
2. $\{a^m b^{m+n} c^n / n, m \geq 1\}$
3. $\{a^m b^n c^{m+n} / n, m \geq 1\}$
4. $\{a^m b^n c^m d^n / m, n \geq 1\}$
5. $\{a^m b^n c^n d^m / m, n \geq 1\}$
6. $\{a^m b^i c^m d^k / m, n \geq 1.\}$
7. $\{a^m b^n / m > n\}$
8. $\{a^n b^{2n} / n \geq 1.\}$
9. $\{w w^R / w \in (a,b)^*\}$ → ND-CFL. but not DCFL.
10. $\{a^n b^n c^n d^m / n \leq 10^{10}$. → RL (also) .$\}$
11. $\{x c y / x, y \in (0,1)^*\}$ — RL (also).
12. $\{x x^R / x \in (a,b)^*, |x| = 2\}$ — RL (also)
13. $\{a^m b^n / m \neq n .\}$
14. $\{a^m b^n / m = 2n+1\}$
15. $\{a^i b^j / i \neq 2j+1\}$
16. $\{a^k / k$ is even $\}$ - RL (also)

17. $\{a^i b^j c^k / j = i+k\}$

18. $\{a^i b^j c^k d^l / i=k \text{ or } j=l\}$ → Non Deterministic CFL.

19. $\{a^m b^l c^k d^n / m,l,k,n \geq 1\}$ — RL (also)

20. $\{a^m b^{4m} / n,m \geq 1\}$ — RL (also)

21. $\{a^3, a^8, a^{13}, \ldots\}$ — RL (also) it is an a A.P.

22. $\{w / w \in \{a,b\}^*, |w| \geq 100\}$ — RL (also)

23. $\{w / w \in \{a,b\}, n_a(w) \geq n_b(w) + 1\}$

- **Not - CFL's -**

1. $\{a^m b^n c^m d^n / m, n \geq 1\}$

2. $\{a^n b^{n^2} / n \geq 1\}$

3. $\{a^n b^{2^n} / n \geq 1\}$

4. $\{ww / w \in (a,b)^*\}$

5. $\{a^n b^n c^m / n > m\}$

6. $\{a^n b^{2n} c^{3n} / n \geq 1\}$

7. $\{www^R / w \in (a,b)^*\}$

8. $\{a^n b^{3^n} / n \geq 1\}$

9. $\{a^{n^2} / n \geq 1\}$

10. $\{a^{2^n} / n \geq 1\}$

11. $\{a^{n!} / n \geq 1\}$

12. $\{a^m / m \text{ is a prime}\}$

13. $\{a^i b^j c^k / i > j > k\}$

14. $\{a^i b^j c^k d^l / i = k \text{ and } j = l\}$

15. $\{a^{n^n} / n \geq 1\}$

16. $\{w / w \in (a,b,c)^* \cdot n_a(w) = n_b(w) = n_c(w)\}$