Regular Grammar

Grammer, G consist of 4 tuples

$$G = (V, T, S, P)$$

V = set of variables / Non-terminals (Capital letters)
T = set of terminals (small letters)
S = starting variable
P = set of Productions

LHS → RHS

Regular Grammar : A grammar is said to be regular if either it is Right linear grammar or Left linear.

RG → Right, Left

Right Linear Grammar

A grammar G is said to be right linear if all the productions are of the following form.

$$\boxed{\begin{array}{l} A \rightarrow xB \\ A \rightarrow x \end{array}} \quad \text{—— (1)}$$

or,

$$\boxed{A \rightarrow xB \mid x} \quad \text{—— (2)}$$

where,

$A, B \in V$
$x \in T^*$

$$\boxed{V \rightarrow T^* V \mid T^*}$$

small letters → T^*; small letters → T^*; single variable (Capital letters) → V

V → single variable; T^* → zero or any number of variables

Ex

① $A \rightarrow aB$ (small → capital), $B \rightarrow b$ } RLG

② $A \rightarrow Ba$, $B \rightarrow b$ } Not RLG

③ $A \rightarrow a$ } RLG

④ $A \rightarrow aBB$, $B \rightarrow b$ (single line) } Not RLG

⑤ $A \rightarrow aaabB$, $B \rightarrow b$ } RLG

Left Linear Grammar

A grammar G is said to be left linear, if all the productions are of the following form.

$$\boxed{\begin{array}{l} A \rightarrow Bx \\ A \rightarrow x \end{array}} \Leftrightarrow \boxed{\begin{array}{l} A \rightarrow xA \\ A \rightarrow x \end{array}} \text{ RLG}$$

or,

$$\boxed{A \rightarrow Bx \mid x}$$

$A, B \in V$
$x \in T^*$

Ex

1. $A \rightarrow Baaa \mid a$, $B \rightarrow b$ } LLG
2. $S \rightarrow aSa \mid a$ } Not LLG
3. $S \rightarrow Ssa \mid a$ } Not LLG

Regular Grammar

Ex

$S \rightarrow A$ → Both (Not Regular)
$A \rightarrow aB \mid \lambda$ → RLG
$B \rightarrow Ab$ → LLG

Linear Grammar

It is a grammar in which at most one variable can occur on the right hand side of any production, without restriction on position of the variable.

Examples →
$S \rightarrow A$
$A \rightarrow aB \mid \lambda$
$B \rightarrow Ab$

NOTE : All regular grammars are linear grammar.

Regular Language to Regular Grammar

RL → FA → RG Process

Ex ① $L = \{a^n : n \geq 0\}$

$= \{\epsilon, a, aa, aaa \ldots\}$

FA:

$\delta(q_0, a) = q_0$ — ① (from automata)

LHS RHS

R.G

$q_0 \rightarrow aq_0$

capital letters

$S \rightarrow aS$

$S \rightarrow \epsilon$

Or,

Q: $S \rightarrow aS \mid \epsilon \rightarrow aaa$

Generating strings

① $\epsilon \rightarrow S \rightarrow \epsilon$

② $a \rightarrow S \rightarrow aS$

$a\epsilon$ ($\because S \rightarrow \epsilon$)

$\rightarrow a$

③ $aa \rightarrow S \rightarrow aS$

$= a\boxed{aS}$ ($\because S \rightarrow aS$)

$= aa\epsilon$ ($\because S \rightarrow \epsilon$)

$= aa$

aaa=? RG $\begin{cases} S = \epsilon & — ① \\ S = aS & — ② \end{cases}$

Is this RG accepts aaa? If yes then show how

$S = aS$

Using ② and S in RHS, we get

$S = aaS$ aab ✗

$S = aaaS$ Yes

Using ①

$aaa + \epsilon = aaa$

Another Method

$L = \{a^n : n \geq 0\}$

1) Write the smallest string generated
2) Take care of what loop is going on.

$\boxed{S \rightarrow aS \mid \epsilon}$ → minimum string

Loop / Repeat

2. $L = \{a^n : n \geq 1\} \rightarrow a$

$= \{a, aa, aaa \ldots\}$

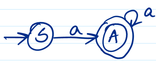
2nd method

$S \rightarrow aS \mid a$ // Myself try

Also correct

1st Method

F.A



$S \rightarrow aA$
$A \rightarrow aA$
$A \rightarrow \epsilon$

$S \rightarrow aA$
$A \rightarrow aA \mid \epsilon$ (combined)

3. $L = \{(a+b)^n : n \geq 0\}$

$= \{\lambda, a, b, \ldots\}$

FA:

$S \rightarrow aS \mid bS \mid \epsilon$ // Direct try ✓

$S \rightarrow aS$
$S \rightarrow bS$
$S \rightarrow \epsilon$

$S \rightarrow aS \mid bS \mid \epsilon$

babb ✓

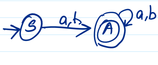
4. $L = \{(a+b)^n : n \geq 1\}$

$= \{a, b, aa, ab \ldots\}$

$S \rightarrow aS \mid bS \mid a \mid b$ // Direct try

correct

FA:

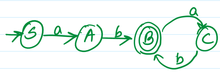


$S \rightarrow aA \mid bA$
$A \rightarrow aA \mid bA \mid \epsilon$

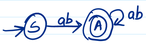
5. $L = \{(ab)^n : n \geq 1\}$

$= \{ab, abab \ldots\}$

$S \rightarrow abS \mid ab$ // Direct try



$S \rightarrow aA$
$A \rightarrow bB$
$B \rightarrow aC \mid \epsilon$
$C \rightarrow bB$



$S \rightarrow abA$
$A \rightarrow abA \mid \epsilon$

6. $L = \{(ab)^n : n \geq 1\}$

F.A

R.G

① $S \rightarrow aA$
$A \rightarrow bB$
$B \rightarrow aA \mid \epsilon$

// Process

② $\boxed{S \rightarrow abS \mid ab}$

// Direct Try

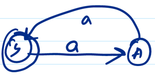
7. $L = \{ n_a(w) \text{ and } n_b(w) \text{ are both even}$
$\Sigma = \{a, b\}$

$n_a(w) \mod 2 = 0$

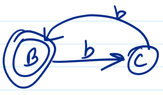
$n_b(w) \mod 2 = 0$

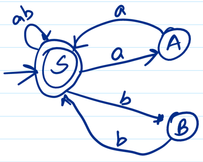
$L = \{ \epsilon, aabb, abab, aa, ab$

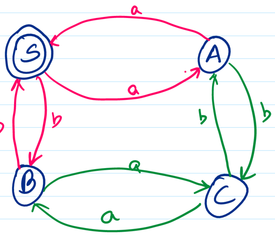
$n_a(w) \rightarrow$



$n_b(w) \rightarrow$







R.L.

$S \rightarrow aA \mid bB \mid \epsilon$

$A \rightarrow aS \mid bC$

$B \rightarrow aC \mid bS$

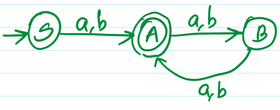
$C \rightarrow aB \mid bA$

8. $L = \{ n_a(w) - n_b(w) \text{ is odd}\}$

$(n_a(w) - n_b(w)) \mod 2 = 1$

odd − even
even − odd

$L_v = \{a, b, aaa, bbb, aab, abb \ldots\}$



R.G

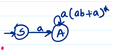
$S \rightarrow aA \mid bA$

$A \rightarrow aB \mid bB$

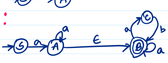
$B \rightarrow aA \mid bA$

Regular Expression to grammar

R.E : $aa^*(ab+a)^*$



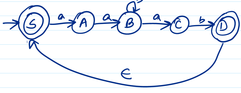
NFA :



RG :

$S \to aA$
$A \to aA \mid \epsilon B$
$B \to aB \mid ac \mid \epsilon$
$C \to bB$

2. $(aab^*ab)^*$

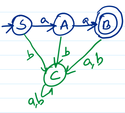


R.E

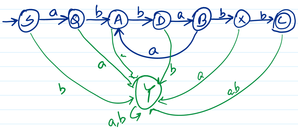
$S \to aA \mid \epsilon$
$A \to aB$
$B \to bB \mid aC$
$C \to bS$

Regular Grammar to DFA

1. $S \to aA$
$A \to aB$
$B \to \lambda$



2. $S \to abA$
$A \to baB$
$B \to aA \mid bb$

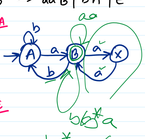


Grammar to Expression

(R.G → FA → RE)

1. $A \to aB \mid bA$
$B \to aaB \mid bA \mid \epsilon$

NFA



R.E

bb^*a

$b^*a(aa+bb^*a)^*$

Right linear Grammar to LLG

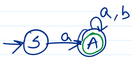
Steps :

1. RLG → FA → FA^R → RLG → LLG

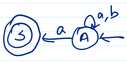
LLG to RLG

1. $S \to aA$
$A \to aA \mid bA \mid \epsilon$

FA



$(FA)^R$



RLG of $(FA)^R$

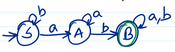
$A \to aA \mid aS \mid bA$
$S \to \epsilon$

LLG

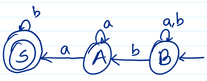
$A \to Aa \mid Ab \mid Sa$
$S \to \epsilon$

2 $S \to bS \mid aA$
$A \to aA \mid bB$
$B \to aB \mid bB \mid \epsilon$

(FA)



$(FA)^R$



RLG of $(FA)^R$

$B \to aB \mid bB \mid bA$
$A \to aA \mid aS$
$S \to bS \mid \epsilon$

LLG

$B \to Ba \mid Bb \mid Ab$
$A \to Aa \mid Sa$
$S \to Sb \mid \epsilon$

Left linear Grammar to RLG

Steps

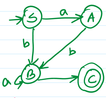
$$\boxed{LLG \rightarrow RLG \rightarrow FA \rightarrow (FA)^R \rightarrow RLG}$$

1. $S \rightarrow Aa \mid Bb$
   $A \rightarrow Bb$
   $B \rightarrow Ba \mid b$

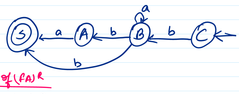
RLG

$S \rightarrow aA \mid bB$
$A \rightarrow bB$
$B \rightarrow aB \mid b$

NFA



$(FA)^R$



RLG of $(FA)^R$

$C \rightarrow bB$
$B \rightarrow aB \mid bC$
$A \rightarrow aS$
$S \rightarrow \epsilon$

Context Free Grammar (CLG)

CFG
↓
CFL (Pumping Lemma for CFL)
↓
PDA (Push Down Automata)

A grammar G is said to be context free if it supports the following function.

$$V \rightarrow (V \cup T)^*$$

Example
① $A \rightarrow aAb \mid b$
② $S \rightarrow aSS \mid \epsilon$
③ $aA \rightarrow aAb \mid ab$ (Not CLG)

Context free language

A language L is said to be context free if there is a context free grammar G that generates the language.

$$L = L(G)$$

Derivation of a string from CFG

1. Left most Derivation (LMD)
2. Right most Derivation (RMD)
3. Derivation Tree

Left most Derivation

A derivation is said to be left most derivation if in each step, the left most variable is the sentential form is replaced.

Q. Derive the string 'aaaab' using LMD from the following grammar.

$S \rightarrow AB$
$A \rightarrow aaA \mid \epsilon$
$B \rightarrow Bb \mid \epsilon$

$S \rightarrow AB$
$S \rightarrow aaAB$ ($\because A \rightarrow aaA$)
$S \rightarrow aaaaAB$ ($\because A \rightarrow aaA$)
$S \rightarrow aaaa\epsilon B$ ($\because A \rightarrow \epsilon$)
$S \rightarrow aaaaBb$
$S \rightarrow aaaaBbb$
$S \rightarrow aaaa\epsilon bb$
$\rightarrow aaaabb$

Right Most Derivation

A derivation is said to be RMD, if in each step the right most variable in the sentential form is replaced.

Q. Same from above LMD.

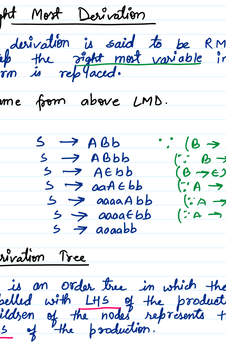
$S \rightarrow ABb$ $\because (B \rightarrow Bb)$
$S \rightarrow ABbb$ ($\because B \rightarrow Bb$)
$S \rightarrow A\epsilon bb$ ($B \rightarrow \epsilon$)
$S \rightarrow aaA\epsilon bb$ ($\because A \rightarrow aaA$)
$S \rightarrow aaaaAbb$ ($\because A \rightarrow aaA$)
$S \rightarrow aaaa\epsilon bb$ ($\because A \rightarrow \epsilon$)
$S \rightarrow aaaabb$

Derivation Tree

It is an order tree in which the nodes are labelled with LHS of the production and the children of the nodes represents the corresponding RHS of the production.

Q. $S \rightarrow AB$
$A \rightarrow aaA \mid \epsilon$
$B \rightarrow Bb \mid \lambda$

string: aaaabb



Unambigous Grammar

A CFG is said to be unambigous if it has exactly one derivation tree or exactly one LMD or RMD.

1. $S \rightarrow AB \mid aaB$
   $A \rightarrow Aa \mid a$
   $B \rightarrow b$

Checking ambigous

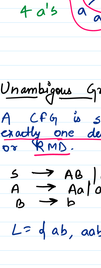
$S \rightarrow aaB$	$S \rightarrow AB$
$S \rightarrow aab$	AaB
	aaB
	aab

$L = \{ab, aab, aaab \text{ --- }\}$

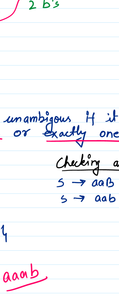
AaB
$AaaB$
$aaaB$
$aaab$
} aaab

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

Checking unambigous using DT



4 levels



3 levels

1. $L = \{ w : w$ starts and ends with same symbol $\}$

over $\Sigma = \{a, b\}$

$L = \{ a(a+b)^*a \;,\; b(a+b)^*b \}$

$S \to aAa \mid bAb$

$A \to aA \mid bA \mid \epsilon$

a, b ✓

babbab ✓

ab ✗

2. $L = \{ w \mid w \in \{a,b\}^*$ and $|w|$ is even $\}$

$L = \{ \epsilon, (aa)^*, (ab)^*, (bb)^*, (ba)^* \}$

$S \to \epsilon \mid aaS \mid abS \mid bbS \mid baS$

3. $L = \{ w \mid w \in \{a,b\}^*$ and $|w|$ is odd $\}$

$L = \{ a \mid b \mid aaS \mid abS \mid bbS \mid baS$

4. $L = \{ a^n b^{n+m} c^m : n, m \geq 0 \}$

In such type of question split this.

$\{ \underbrace{a^n b^n}_{A} \; \underbrace{b^m c^m}_{B} : n, m \geq 0 \}$

AB

$S \to AB$

$A \to aAb \mid \epsilon$

$B \to bBc \mid \epsilon$

aabbbc

5. $L = \{ a^{n+m} b^m c^n : n, m \geq 0 \}$

$L = \{ \underbrace{a^n a^m}_{A} \; \underbrace{b^m c^n}_{B} : n \geq 0 \}$

$n \to 2$
$m \to 3$

aaaaabbbcc

$S \to aSc \mid A$

$A \to aAb \mid \epsilon$

aabc ✓

ab

ab

aSc

aascc

aaascc

aaaAbcc

aaaabcc

$n = 1$ $n \geq 2$

aaaabcc

6

$S \to aSc \mid aAc$

$A \to aAb \mid ab$ → for $m, n \geq 1$

aascc

aaascc

1. 1 & 4
2. consecutive possible.

7. $L = \{ a^n b^m c^k \mid n = m$ or $n \leq k,\; n, m, k \geq 0 \}$

If $n = m$; $\underbrace{a^n b^n}_{A} \underbrace{c^k}_{B}$

If $m \geq k$; $\underbrace{a^n b^k}_{C} \underbrace{c^k}_{D}$

If $m < k$;

$\underbrace{c^k}_{E}$

$S \to AB \mid CD$

$A \to aAb \mid \epsilon$

$B \to cB \mid \epsilon$

$C \to aC \mid \epsilon$

$D \to bDc \mid \epsilon$

$E \to cE \mid \epsilon$

$\underbrace{a^n b^m}_{A} \underbrace{c^k}_{B}$ $n \geq m$

$m < k \mid m \geq k$

$\underbrace{a^n}_{C} \underbrace{b^k c^k}_{D}$

$S \to AB \mid CD$

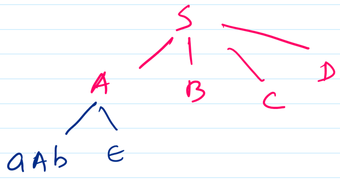
$A \to aAb \mid \epsilon$

$B \to bB \mid \epsilon$

$C \to aC \mid \epsilon$

$D \to bDc \mid \epsilon$

$E \to cE \mid \epsilon$



Q.(i) Define PL for RL & CFL

(ii) Prove

Let L be a infinite CFL. For some positive integer P so that any string $w \in L$, it must satisfy the following properties.

$|w| \geq P$

$w = uvxyz$

with $|vxy| \leq P$

and $|vy| \geq 1$

then for all $i \geq 0$

$uv^ixy^iz \notin L$.

① $|w| \geq P$

② $w = uvwxyz$

③ $|vxy| \leq P$

④ $|vy| \geq 1$

1. $\{L = a^n b^n c^n : n \geq 1\}$. Prove that L is not CFL using pumping lemma.

Let assume that L is CFL.

Let $w = a^P b^P c^P$

Here, $|w| = |a^P b^P c^P|$

$= 3P \geq P$

$\underset{u}{a^{P-1}}\ \underset{v}{a^1}\ \underset{x}{b^{P-2}}\ \underset{y}{b}\ \underset{z}{bc^P}$

ii) $|vxy| = |ab^{P-2}b| = |1+P-2+1|$

$= P \leq P$

(iii) $|vy| = ab = |2| \geq 1$

Let $i = 2$; uv^ixy^iz

$\Rightarrow uv^2xy^2z$

$= \underset{u}{a^{P-1}}\underset{v}{a^2}\underset{x}{b^{P-2}}\underset{y}{b^2}\underset{z}{bc^P}$

$= a^{P+1} b^{P+1} c^P \notin L$

It contradicts our assumption that L is CFL

$\therefore$ L is not CFL.

Proved.

<u>Simplification of CFG</u>

1. Removal of ϵ- production
2. Removal of unique productions
3. Removal of useless productions

<u>Removal of ϵ- production</u>

Any production of the form $A \to \epsilon$ is called ϵ production.

1. $S \to ABaC$
   $A \to Bc$
   $B \to b \mid \epsilon$
   $C \to D \mid \epsilon$
   $D \to d$

<u>Step 1</u>: Find Nullable variables which can generate ϵ

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

<u>Step 2</u>: Remove the ϵ production

$S \to ABaC$ | Remove A: BaC | Remove B: AaC | Remove C: ABa | Remove AB: aC

Remove AC: Ba | Remove BC: Aa | Remove ABC: a

// $A \to Bc \mid B \mid C$ // Doing same as above

$B \to b$

$C \to D$

$D \to d$

After Removal of ϵ production this will be the production.

2. $S \to AaB \mid aaB$
   $A \to \epsilon$
   $B \to bbA \mid \epsilon$

Remove the ϵ- production

$\{A, B\}$

$S \to AaB \mid aB \mid Aa \mid a \mid aa \mid aaB$

$B \to bb \mid bbA$

Here, A is not reachable so directly remove them.

$S \to aB \mid a \mid aa \mid aaB$

$B \to bb$

3. $S \to AB$
   $A \to aAA \mid \epsilon$
   $B \to bBB \mid \epsilon$

$\{A, B, S\}$

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

$A \to aAA \mid aA \mid a$

$B \to bBB \mid bB \mid b$

NOTE: If ϵ belongs to the language L, then S is deriving ϵ. In this case do not remove ϵ from production S.

4. $S \to aSSS \mid bb \mid \lambda$

$S \to aSSS \mid aSS \mid aS \mid bb \mid \epsilon$

$B \to L$

$\boxed{A \to B}$ (Unit prod)

ϵ - production

CFL $\to$ CFG $\to$ GNF

$A \to Ba$

Removal of Unit Production

Any production of the form $A \to B$ where A, B belongs to V is called unit-production.

Format

$$A \to B$$

1. $S \to Aa \mid B$
   $B \to A \mid bb$
   $A \to a \mid bc \mid B$

If grammar contains ϵ-production then first remove it.

$S \to B$
$B \to A$
$A \to B$

Substitute the RHS of the unit productions with the productions of that variable.

$\because A \to a \mid bc$

$\therefore S \to Aa \mid bb \mid a \mid bc$

$B \to a \mid bc \mid bb$

$A \to a \mid bc \mid bb$

B is not reachable because $S \to$ don't have any connection with B so final answer is

$S \to Aa \mid bb \mid a \mid bc$
$A \to a \mid bc \mid bb$

2. $S \to AB$
   $A \to a$
   $B \to C \mid b$
   $C \to D$
   $D \to E$
   $E \to a$

(Unit productions)

$S \to AB$
$A \to a$
$B \to a \mid b$ $\quad \because (E \to a)$
$C \to a$
$D \to a$
$E \to a$

Here, C, D, E are not reachable so final Answer is

$S \to AB$
$A \to a$
$B \to a \mid b$

3. $E \to E+T \mid T$
   $T \to T*F \mid F$
   $F \to (E) \mid a$

$E \to E+T \mid T*F \mid (E) \mid a$
$T \to T*F \mid (E) \mid a$
$F \to (E) \mid a$

Removal of Useless Productions

A variable A is said to be useful if and only if there is at least a string w belongs to the language L such that $S \to xAy \to w$, then that variable is useful.

where x, y belong to $(V \cup T)^+$

A variable that is not useful is called useless. 2 reasons why a variable is called useless

1) Variable cannot be reached from the starting symbol.
2) Variable cannot derive a terminal string.

Steps to remove useless productions

1) Find the useful set, All terminals are useful symbols
2) If any variable produces only terminals then add that variable to the useful set.
3) If any variable produces a combination of useful elements completely then add that variable to the useful symbol.

   Ex: $A \to a$
   $B \to aA$

   Here, Both are useful variable.

4) Find the useless set. Remove all useless productions from the LHS
5) From the RHS, where useless elements, then remove that term.
6) If there is any unreachable variable from the starting production then remove that production.

Ex 1. $S \to AB \mid a$
$A \to Bc \mid b$
$B \to aB \mid C$
$C \to ac \mid B$

Useful set = {a, b, S, A} (step-1)

Useless elements = {B, C} (step-2)

No change in step-3.

Now, Removing production 3 & 4 (step-4)

$S \to AB \mid a$
$A \to Bc \mid b$

(step-5)

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

(step-6)

$S \to a$

2. $S \to aS \mid A \mid C$
   $A \to a$
   $B \to aa$
   $C \to aCb$

Useful set = {a, b, A, B, S}

Useless set = {C}

$S \to aS \mid A$
$A \to a$
$B \to aa$

$S \to aS \mid A$
$A \to aa$ (final Answer)

3. $S \to ABC \mid BaB$
   $A \to aA \mid BaC \mid aaa$
   $B \to bBb \mid a$
   $C \to CA \mid AC$

Useful set = {a, b, A, B, S}

Useless set = {C}

$S \to BaB$

$B \to bBb \mid a$

4. $S \to AB \mid AC$
   $A \to aAb \mid bAa \mid a$
   $B \to bbA \mid aaB \mid AB$
   $C \to abCA \mid aDb$
   $D \to bD \mid aC$

Useful set = {a, b, A, B, S}

Useless = {C, D}

3 productions

1. $S \to ABA \mid aSa \mid bSb$
   $A \to aA \mid aAb \mid B$
   $B \to bB \mid \epsilon$

ϵ - nullables = {S, A, B}

$S \to ABA \mid aSa \mid bSb \mid BA \mid AB \mid AA \mid A \mid B \mid aa \mid bb \mid \epsilon$

$A \to aA \mid aAb \mid B \mid a \mid ab$

$B \to b \mid bB$

Unit Productions

$S \to A$
$S \to B$

$S \to ABA \mid aSa \mid bSb \mid BA \mid AB \mid AA \mid aA \mid aAb \mid ab \mid a \mid b \mid \epsilon \mid aa \mid bb$

CNF

Simplify

1. $S \to aA \mid aBB$
$A \to aaA \mid \epsilon$
$B \to bB \mid bbC$
$C \to B$

Unit Production

2. $S \to a \mid aA \mid B \mid C$
$A \to aB \mid \epsilon$
$B \to Aa$
$C \to CCD$
$D \to ddd$

Useless Production

3. $S \to as \mid aa \mid \epsilon$
$A \to bA$
$B \to AA$

After simplification, what lang. does this grammar generate?

ϵ - Null $\to \{A\}$

$S \to aA \mid aBB \mid a$
$A \to aaA \mid aa$
$B \to bB \mid bbC$
$C \to B$

Unit Production

$S \to aA \mid aBB \mid a$
$A \to aaA \mid aa$
$B \to bB \mid bbC$
$C \to bB \mid bbC$

Useful set

a, b, A, S, B, C

ϵ - null $\to \{A\}$

$S \to a \mid aA \mid B \mid C$
$A \to aB$
$B \to Aa \mid a$
$C \to CCD$
$D \to ddd$

Unit Production

$S \to B$
$S \to C$

$S \to a \mid aA \mid Aa$
$S \to CCD$

Normal Form of CFG

Normalization is performed to standardize the grammar.

By simplification of the grammar the grammar gets minimized but does not get standardized. This is because the RHS of the productions have no specific format.

Normalization is of types

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

CNF

A context free grammar is in CNF if all productions are of the form

$A \to BC$
or, $A \to a$

where A, B, C belongs to V and a belongs to T.

Steps to make grammar into CNF

1. First remove all ϵ- productions
2. Then remove all unit productions

Do not remove useless productions

3. Eliminate the terminals from RHS if they exist with other terminals or variables

Ex: $A \to aB$
$A \to V_1 B$
$V_1 \to a$

4. Eliminate RHS with more than two non-terminals

Ex: $A \to BCD$
$A \to BV_1$
$V_1 \to CD$

5. Repeat step 3 & 4 until RHS of productions contains two non-terminals or one terminal

1. $S \to ABa$
$A \to aab$
$B \to Ac$

$S \to ABV_1$
$V_1 \to a$
$A \to V_1 V_1 V_2$
$V_2 \to b$
$B \to AV_3$
$V_3 \to c$

After step-3

$S \to AX$
$X \to BV_1$
$V_1 \to a$
$A \to V_1 Y$
$Y \to V_1 V_2$
$V_2 \to b$
$B \to AV_3$
$V_3 \to c$

After step 8.

This answer is final and is in CNF.

2. $S \to abAB$
$A \to bAB \mid \epsilon$
$B \to Baa \mid A \mid \epsilon$

Convert this to CNF.

ϵ - null

$\{A, B\}$

$S \to abAB \mid abB \mid abA \mid ab$
$A \to bAB \mid bA \mid bB \mid b$
$B \to Baa \mid A \mid aa$

Unit productions

$\to B \to A$

$S \to abAB \mid abB \mid abA \mid ab$
$A \to bAB \mid bA \mid bB \mid b$
$B \to Baa \mid bAB \mid bA \mid bB \mid b \mid aa$

To CNF

$S \to V_1 V_2 AB \mid V_1 V_2 B \mid V_1 V_2 A \mid V_1 V_2$
$V_1 \to a$
$V_2 \to b$
$A \to V_2 AB \mid V_2 A \mid V_2 B \mid b$

$B \to BV_1 V_1 \mid V_2 AB \mid V_2 A \mid V_2 B \mid b \mid V_1 V_1$

step-3

$S \to V_1 V_7 \mid V_1 V_4 \mid V_1 V_5 \mid V_1 V_2$
$V_3 \to AB$
$V_4 \to V_2 B$
$V_5 \to V_2 A$

$A \to V_2 V_3 \mid V_2 A \mid V_2 B \mid b$
$B \to BV_6 \mid V_2 V_3 \mid V_2 A \mid V_2 B \mid b \mid V_1 V_1$
$V_6 \to V_1 V_1$
$V_7 \to V_2 V_3$

Final Answer.
After repeating step-4 again 1 time.

Greibach Normal Form

A context free grammar is in GNF if all productions are of the form

$$A \rightarrow ax$$

where $a \in T$ and $x \in V^*$ → V derives

Generalized Format

1 variable → 1 terminal then any no. of variable, or

1 variable → 1 terminal.

Example

$S \rightarrow aAB$
$A \rightarrow aA \mid bB \mid b$ → GNF
$B \rightarrow b$

CFG to GNF

How to make a grammar to GNF

CFL → CFG → CNF → Recursion → GNF

Step 1

Convert the grammar to CNF

Step 2

Eliminate left recursion from the grammar.

Step 3

In the RHS if the first symbol is a variable then substitute that variable with productions of that variable.

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

$S \rightarrow aB$
$A \rightarrow a$ → GNF
$B \rightarrow b$

Step-4

In RHS if a terminal is present at any place except the first place then introduce a new production.

Ex: $S \rightarrow aa$

$S \rightarrow aV_1$ → GNF
$V_1 \rightarrow a$

Step-5

Repeat step 3 & 4 until it satisfy the GNF format.

Example 1

$S \rightarrow AB$
$A \rightarrow aA \mid bB \mid b$
$B \rightarrow b$

NOTE: You can only remove ε-production and unit production. It will be easier

$S \rightarrow aAB \mid bBB \mid bB$
$A \rightarrow aA \mid bB \mid b$
$B \rightarrow b$ Final Answer

Example 2

$S \rightarrow absb \mid aa$

$S \rightarrow aV_1SV_1 \mid aV_2$
$V_1 \rightarrow b$
$V_2 \rightarrow a$

Example 3

$S \rightarrow ABb \mid a \mid b$
$A \rightarrow aaA \mid B$
$B \rightarrow bAb$

$S \rightarrow ABb \mid a \mid b$ → Conversion to CNF
$A \rightarrow aaA \mid bAb$
$B \rightarrow bAb$

$S \rightarrow aV_1ABV_1 \mid a \mid b$
$A \rightarrow aV_1A \mid bAV_2$
$B \rightarrow bAV_2$
$V_1 \rightarrow a$
$V_2 \rightarrow b$ Final Answer

1. $S \rightarrow XA \mid BB$
$B \rightarrow b \mid \boxed{BB}$
$X \rightarrow b$
$A \rightarrow a$

$S \rightarrow bA \mid BB$
$B \rightarrow b \mid XAB \mid BBB$
$X \rightarrow b$
$A \rightarrow a$

Removing left recursion $B \rightarrow BBB$

$S \rightarrow bA \mid BB$

$B \rightarrow bC \mid bABC$
$C \rightarrow BBC \mid \epsilon$

$X \rightarrow b$
$A \rightarrow a$

$S \rightarrow bA \mid BB$
$B \rightarrow bC \mid bABC \mid b \mid bAB$
$C \rightarrow BBC \mid BBB$
$X \rightarrow b$
$A \rightarrow a$

$S \rightarrow bA \mid bC \mid bABC \mid bB \mid bAB$
$B \rightarrow bC \mid bABC \mid b \mid bAB$
$C \rightarrow bCBC \mid bABCBC \mid bBC \mid bABBC \mid\mid bCB \mid$
$bABCB \mid bB \mid bABB$

$X \rightarrow b$
$A \rightarrow a$

Q. Convert to GNF.

$S \rightarrow abAB$
$A \rightarrow bAB \mid \epsilon$
$B \rightarrow Baa \mid A \mid \epsilon$

Convert this to CNF.

$S \rightarrow abAB \mid abB \mid abA \mid ab$
$A \rightarrow bAB \mid bA \mid bB \mid b$
$B \rightarrow Baa \mid A \mid aa$

$S \rightarrow abAB \mid abB \mid abA \mid ab$
$A \rightarrow bAB \mid bA \mid bB \mid b$
$B \rightarrow Baa \mid bAB \mid bA \mid bB \mid aa \mid b$

GNF $V_3 \rightarrow b$
$V_1 \rightarrow b$
$S \rightarrow aV_1AB \mid aV_1B \mid aV_1A \mid aV_1$
$A \rightarrow bAB \mid bA \mid bB \mid b$
Recursion $B \rightarrow BV_1V_1 \mid bAB \mid bA \mid bB \mid aV_1 \mid b$
$B \rightarrow bABC \mid bAC \mid bBC \mid aV_1C \mid bC$
$C \rightarrow BV_1C \mid \epsilon$

Left Recursion is removed like Baa to aaZ.

ε - null
$V_1 \rightarrow b$

$S \rightarrow aV_1B \mid aV_1B \mid aV_1A \mid aV_2$
$A \rightarrow bAB \mid bA \mid bB \mid b$
$B \rightarrow bABC \mid bAC \mid bBC \mid aV_1C \mid bC \mid\mid$
$bAB \mid bA \mid bB \mid aV_1 \mid aV_1 \mid b$

Further Process $C \rightarrow BV_1$ (After replacing B)
$C \rightarrow bABCV_1 \mid bACV_1 \mid bBCV_1 \mid aV_1CV_1 \mid bCV_1 \mid\mid$
$bABV_1 \mid bAV_1 \mid bBV_1 \mid aV_1V_1 \mid bV_1$

Formula for substituting Left Recursion

$$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$$

$$A \to \beta_1 Z \mid \beta_2 Z \mid \beta_n Z$$

$$Z \to \alpha_1 Z \mid \alpha_2 Z \mid \cdots \mid \epsilon$$

A PDA is a machine capable of recognizing CNF.

PDA consists of 7 tuples

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

where Q = set of states

Σ = alphabets

δ, Γ = finite set of symbol called the stack alphabet

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

q_0 belongs to the initial state

$Z_0 \in \Gamma$ is the stack start symbol

$F \subseteq Q$ is set of final states

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

Basic Operations of Stack

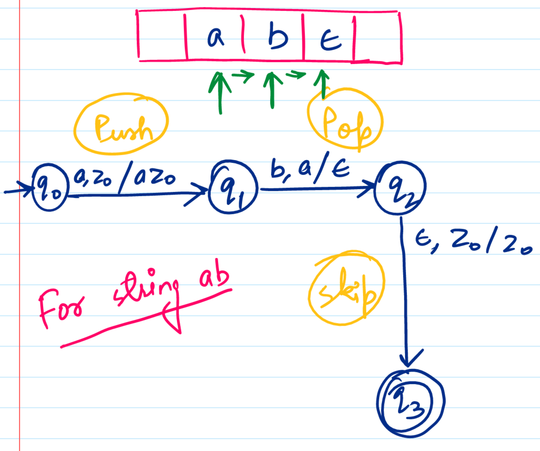
1. Push Operation
2. Pop operation
3. skip Operation

For better understanding read from sir ppt.

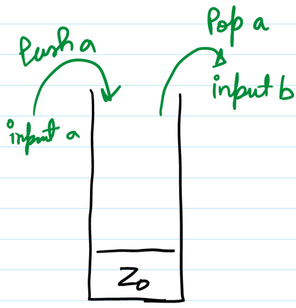
Same stack top and go to different states with some operation, not possible in DPDA but possible in NPDA.

1. Design npda or dpda for $L = \{a^n b^n : n \geq 1\}$

$L = \{ab, aabb, aaabbb, \ldots\}$

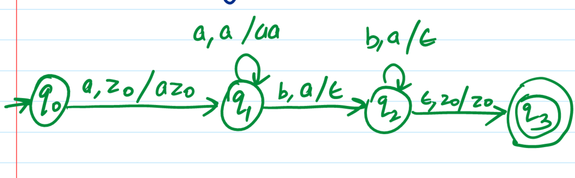


For string ab



For this question only

1. Whenever a comes, push a into the stack.
2. Whenever b comes, pop a from the stack.
3. Whenever ε comes, perform skip operation on the stack and go to the final state.

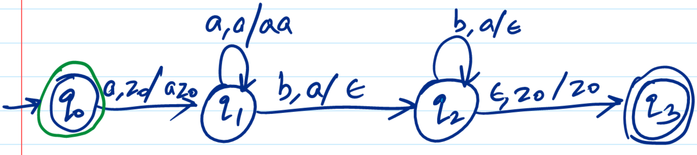


Final Answer

$\boxed{\lambda}$ ✓ $\boxed{\epsilon|a|b|\epsilon}$ ✗

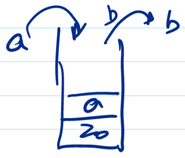
$L = \{a^n b^n : n \geq 0\}$

$L = \{\epsilon, ab, aabb, aaabbb, \ldots\}$



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

$L = \{abb, aabbbb\}$



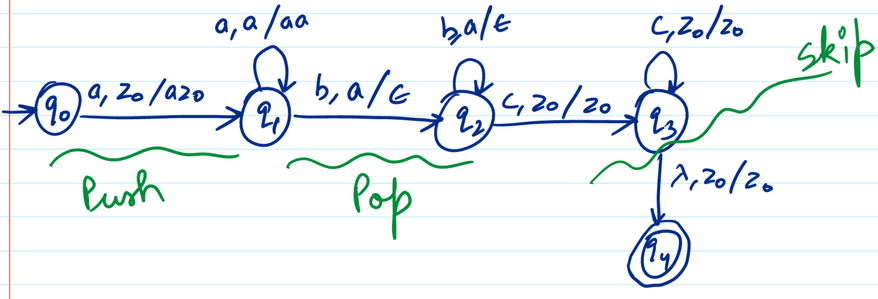
3. $L = \{a^n b^n c^m : n, m \geq 1\}$

$L = \{abc, abcc,$

Push a
Pop a

<u>Comparison</u> → Push, Pop

Non-comparison → skip



Id - Instantaneous Description

Id is an informal location to check how a PDA computes a input string and make a decision that a string is accepted or rejected.

It consist of a triple notation

(q, w, u) where

$q \in Q$ (q is the current state of the control unit)

$w \in \Sigma^*$ (w is the remaining unread part of the input string)

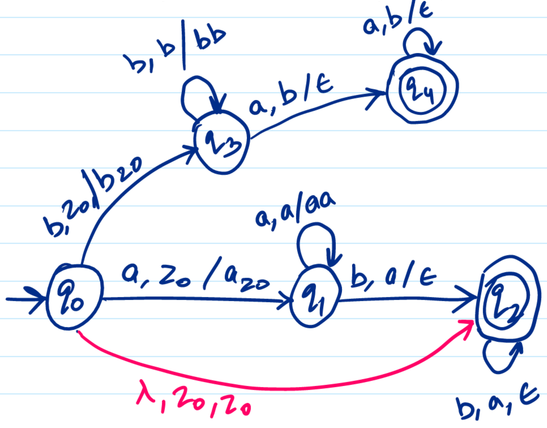
$u \in \Gamma^*$ (u is the current entire stack contents)

A move from one Id to another is denoted by the symbol $\vdash$

$(q_1, aw, bx) \vdash ($

$L = \{ w \in \{a,b\}^* \mid n_a(w) = n_b(w) \}$

c
ab ✓
ba ✓
abab
baba
abba
baab
aabb ✓
bbaa ✓

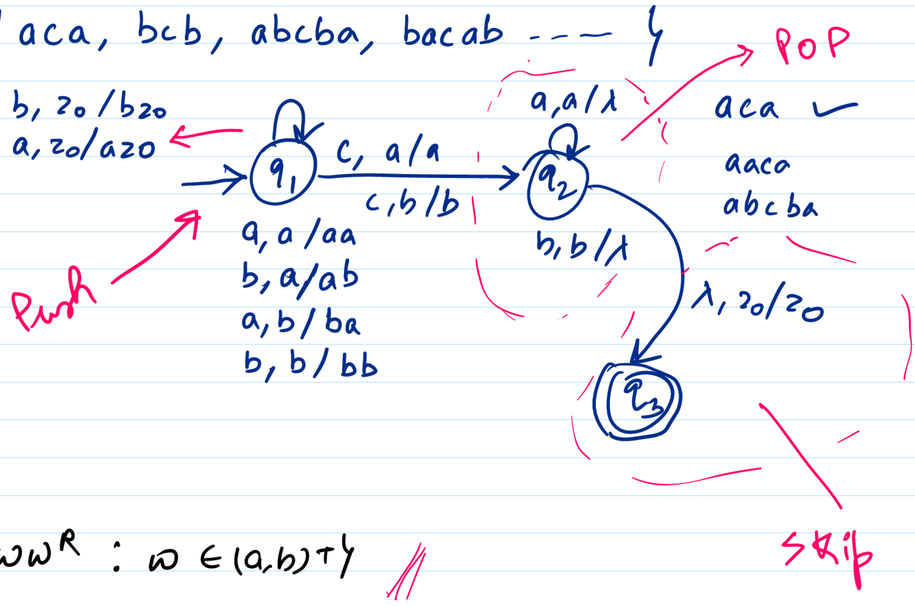


a, b / λ → Pop a
b, a / λ → Pop b

a, z0 / az0
b, z0 / bz0
a, a / aa
b, b / bb

2. $L = \{ wcw^R : w \in (a,b)^+ \}$

$L = \{ aca, bcb, abcba, bacab \text{ --- } \}$



$L = \{ ww^R : w \in (a,b)^+ \}$ //

Put a ε in between like

$\{ w \lambda w^R \}$

Then go with the skip operation.

3. $L = \{ a^n b^m ; n \le m \text{ and } n, m \ge 0 \}$

4. $L = \{ w \in \{a,b\}^+ \mid n_a(w) \ge n_b(w) \}$

Give a sequence ID of your PDA leading to the acceptance of the string baaba

5. $L = \{ a^i b^j c^k : i = j + k \text{ and } i, j, k \ge 0 \}$

Give ID for string 'aabc'

CFG to PDA

CFL → CFG → GNF → PDA

Question

1. Design a npda for the CFG

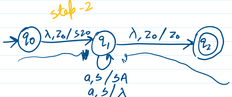
$S \to aSA \mid a$
$A \to bB$
$B \to b$

$S \to aSA$
$S \to a$
$A \to bB$
$B \to b$

} These are in GNF
(If Not then convert in GNF)
→ Step 1

If this is there then PDA will be

Step-2



$a, S / SA$
$a, S / \lambda$
$b, A / B$
$b, B / \lambda$

GNF

NOTE

1. The number of states will be always 3 irrespective of the question.

Initial → Intermediate → final
q_0 q_1 q_2

2. There will always be a loop on the intermediate state

3. Always the first step will be $\lambda, z_0 \mid S z_0$ irrespective of question.
This may change if starting symbol is different.

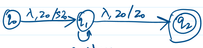
4. Always the transition from intermediate to final state will be $\lambda, z_0 / z_0$ irrespective of the question.

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

$S \to abb \mid ab$

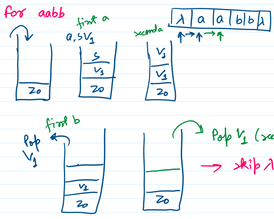
$S \to aSV_1 \mid aV_1$
$V_1 \to b$

} GNF



$a, S \mid S V_1$
$a, S \mid V_1$
$b, V_1 \mid \epsilon$

for aabb



2. $S \to abSb \mid \lambda$

$S \to abSb \mid abb$

$S \to aVSV \mid aVV \mid \epsilon$
$V \to b$
$S \to aVV$

$a, S \mid VSV$
$b, V \mid \epsilon$
$a, S \mid VV$
$\epsilon, S \mid \epsilon$

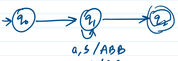
3. $S \to aABB \mid aAA$
$A \to aBB \mid b$
$B \to bBB \mid A$

No ε-production Rule.

Unit Production

$B \to A$

$S \to aABB \mid aAA$
$A \to aBB \mid b$
$B \to bBB \mid aBB \mid b$



$a, S / ABB$
$a, S / AA$
$a, A / BB$
$b, A / \epsilon$
$b, B / BB$
$a, B / BB$
$b, B / \epsilon$

Q. $S \to aABa \mid aAA$
$A \to aBb \mid b$
$B \to bBa \mid A$

No ε production

Unit production

$B \to A$

$S \to aABa \mid aAA$
$A \to aBb \mid b$
$B \to bBa \mid aBb \mid b$

$S \to aABV_1 \mid aAA$

$A \to aBV_1 \mid b$

$B \to bBV_1 \mid aBV_2 \mid b$

$V_1 \to a$

$V_2 \to b$

Q. $S \to aA$
$A \to aABC \mid bB \mid a$
$B \to b$
$C \to c$

Q. $S \to XaaX$
$X \to ax \mid bx$

Q. $S \to aABc \mid aAd$
$A \to aBB \mid b$
$B \to bBd \mid A$

Turing machine was invented in 1936 by **Alan Turing**.

B → means Blank. This is used to denote starting and ending of a string.

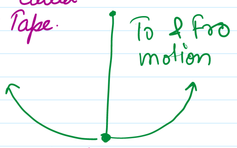
B	B	a	b	a	b	a	b	B	B

This is called Input Tape.

starting of string

Ending of string

Rotate in Between of this



1. Design a Turing Machine that accepts the language

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

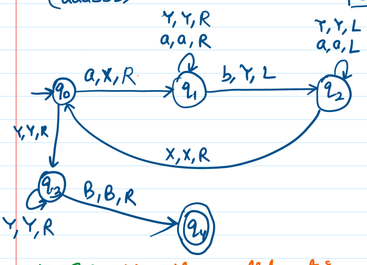
min string (Avoid)

→ start making with a bigger string

→ lets take $n=3$

(aaabbb)

B	B	a	a	a	b	b	b	B	B
		0	1	2	3	4	5	6	



We cannot come directly q_1 to q_4 because aaa will also be accepted.

NOTE: All the valid strings should end at the final state and all the invalid string can end at any other state including initial state.

Here, you don't have to make any deadstate and not any transition for invalid strings.

→ Try to find states of some invalid strings — abb, aabba

2. $L = \{a^n b^n c^m : n, m \geq 0\}$

$n < m$ ✓

$n > m$ ✓

$n = m$ ✓

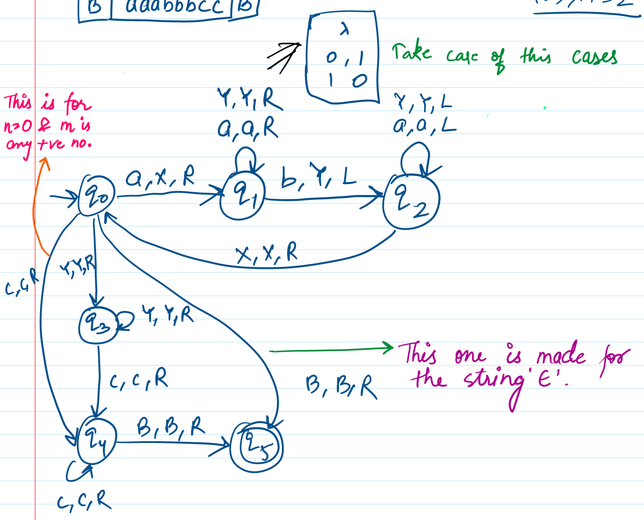
$n=0, m=2$

B	aaabbbcc	B

λ
0, 1
1 0

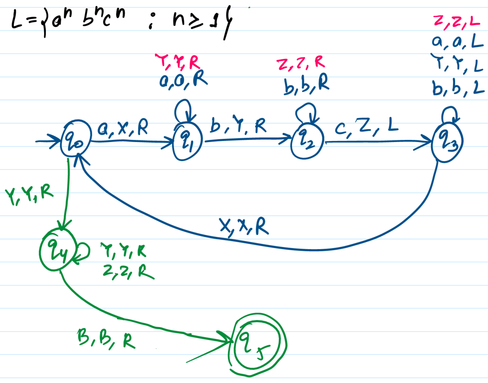
Take care of this cases

This is for $n=0$ & m is any +ve no.



This one is made for the string 'ε'.

3. $L = \{a^n b^n c^n : n \geq 1\}$



Lets take $n=3$

string → aaabbbccc

R1 : XaaYbbZ

R2 : XXaYYbZZ

R3 : XXXYYYZZZ
0 1 2

After coming to the 2nd index of X you have to write transition for Y,Y,R and go to new state and then for Z,Z,R and then B,B,R go to a final state.

4. $L = \{ ww^R : w \in \{a,b\}^* \}$

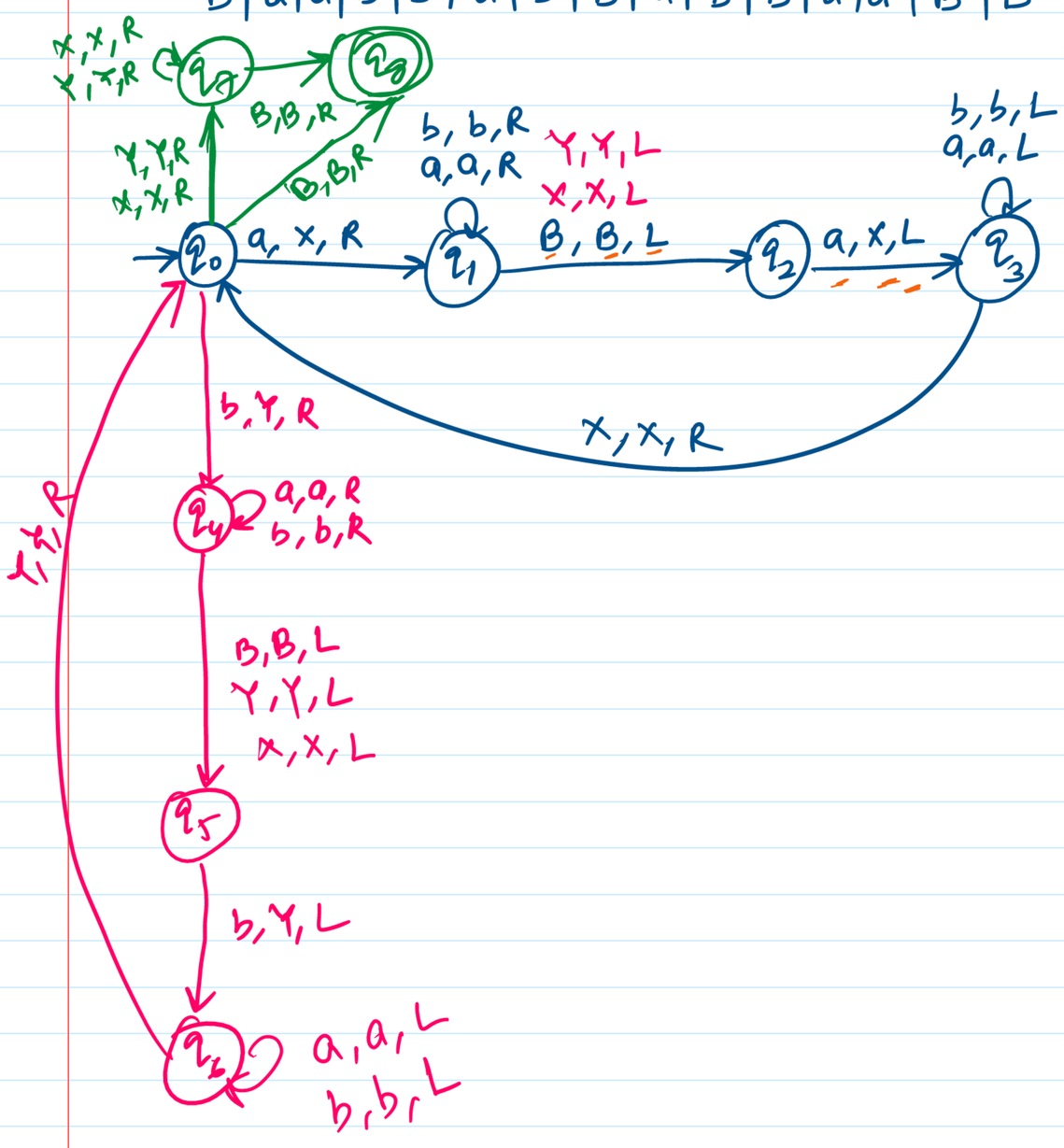
$w = aabbab$

$w^R = babbaa$ (reverse)

$ww^R = aabbabbabbaa$

Input Tape

B | a | a | b | b | a | b | b | a | b | b | a | a | B | B → Combined string



p p^R

a b a a b a B

X b a a b X B

X Y a a b X

X Y a a Y X

X Y a a Y X

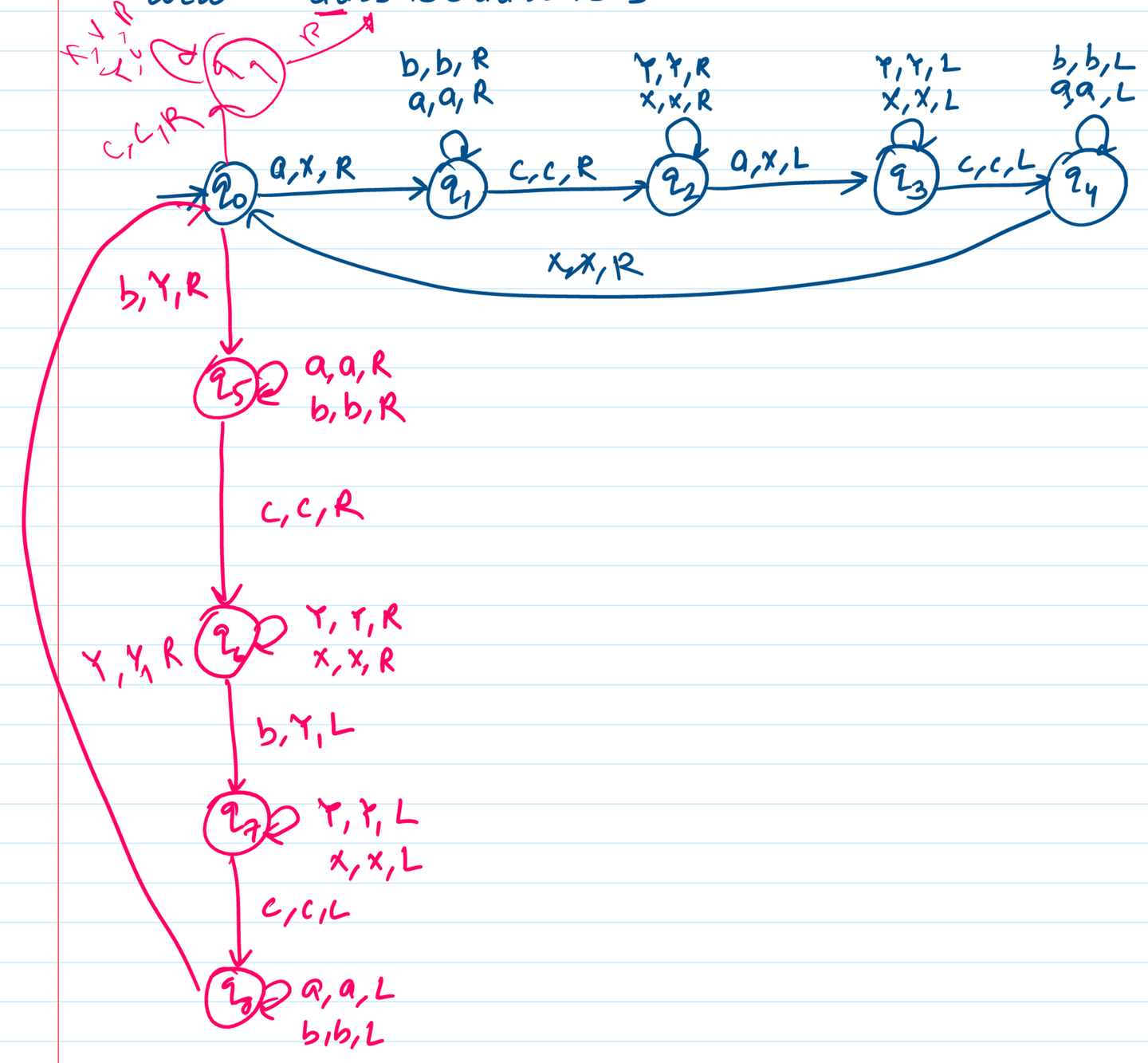
X Y X a Y X

X Y X X Y X

5. $L = wcw : w \in \{a,b\}^*$

Let $w = aabbab$

$wcw = aabbabcaabbab$ B



wcw (PDA)

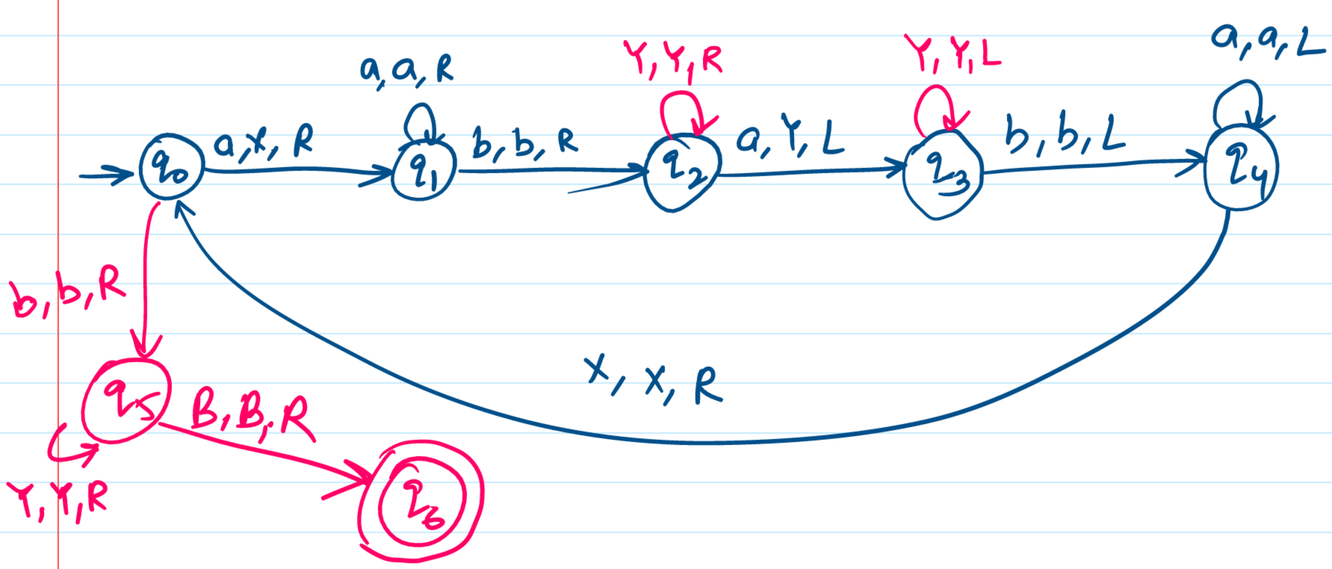
ab a c a b a B

X Y a c a Y X

X Y X c X Y X

6. $L = \{ w \in a^n b a^n : n \geq 1 \}$

Input tape → | a a a b a a a B



Turing Machine contd.

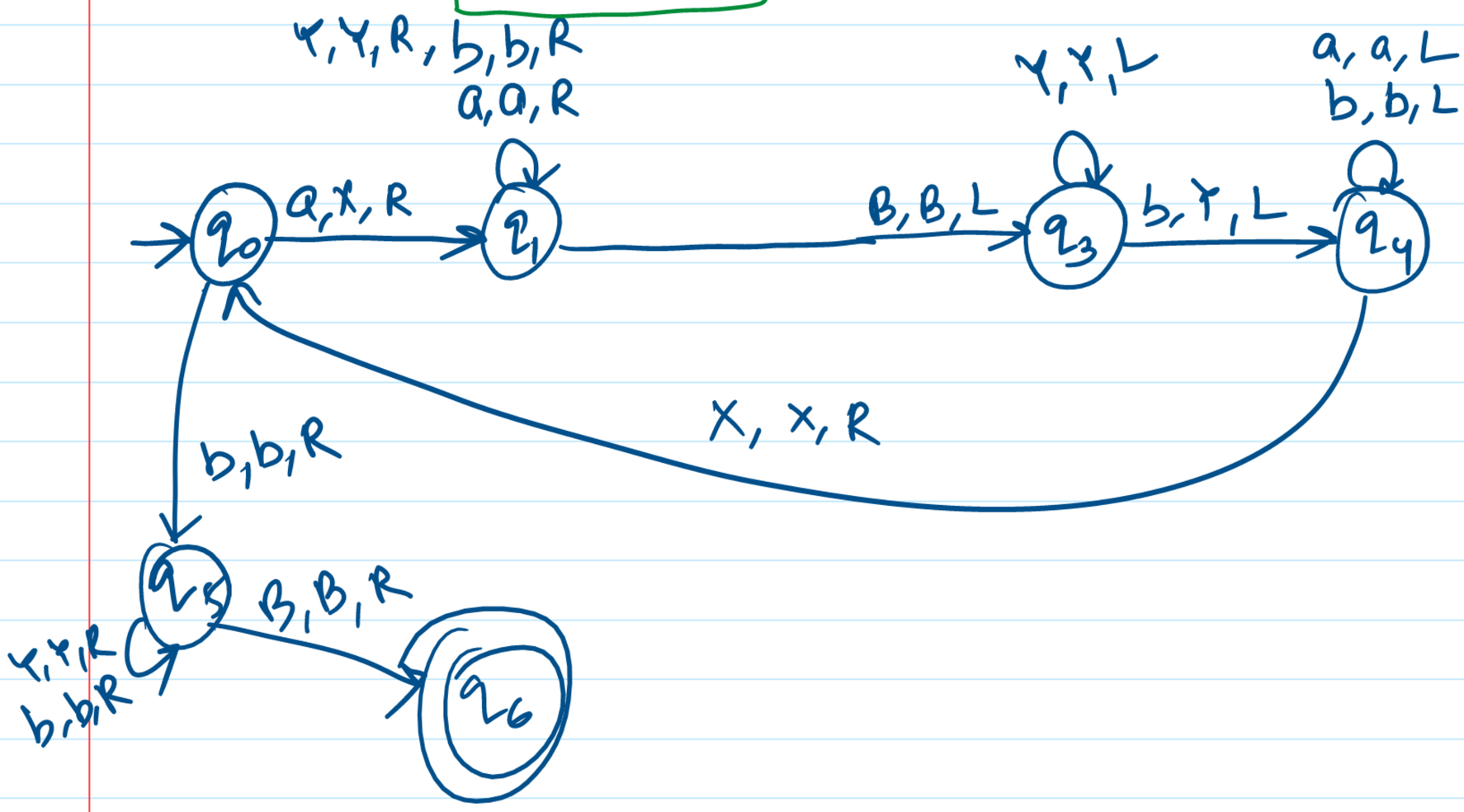
29 April 2022 12:33

$L = \{ a^m b^n \mid m < n \; ; \; m, n \geq 0$

m	n
0	1
1	2
2	3
1	3

Input Tape

$\boxed{aabbbb\,B}$



aabbb

X X Y Y

a a a b b b

X a a b b Y

X X a b Y Y

X X X Y Y Y

aaa bb

aab

X a Y

X X Y