Date: Day:

When we apply 0 on (1, 2, 4) it should all go to one state, similarly on applying 1 on (1, 2, 4).

Pumping lemma: (property of regular language)

(can't be used to prove language is regular.)

One of method to check language is regular or not

$a^n b^n$, $n \geq 0$.

- If we don't know n beforehand we can't create finite automata for it

↳ Non regular language

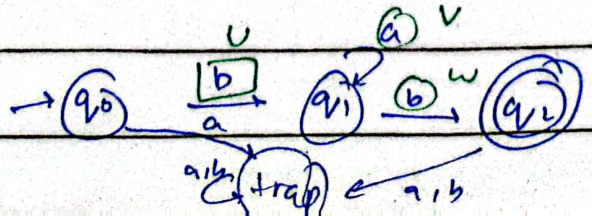
→ a^2b^2, $a^{10}b^{10}$ have finite automata

We can't do this because we have not any additional memory

$a^n b a^n$ → don't have finite automata.

L → Regular → FA → n states

if $x \in L$ and $|x| \geq n$. then corresponding FA has loop.



length.	x	loop.
$\|x\| = 2$	bb	0
$\|x\| = 3$	bab	1
$\|x\| = 4$	baab	2

Note: of length

If a string greater than n state is accepted, it means loop must exist.

before loop — loop — after loop

$x = u\ v\ w$

So, (i) $|v| \neq 0 \rightarrow |v| \geq 1$. length of loop, it can be a indirect loop

(ii) $|uv| \leq n$ (length of string) • uv can't be 0, bcz min length of v is 1

iii) for all $i \geq 0$ $uv^i w \in L$.

↓ pumping length of v^i, should result in getting string, belonging to L. (power i times)

→ $L_1 = \{ x \mid x \in \Sigma^* \text{ & } x = a^m b^n \text{ where } m \geq 0 \}$

Prove by contradiction

Let L is regular ⇒ DFA with n states.

MUST steps 1/2

Every division of a instance should contradict, if any one does not, we exclude instance

Date: But we can check on different "i" at each division Day: _____.

$x = a^n b^n$ $x = a^3 b^3$ part of language, but don't know if length is greater than n.

$\rightarrow |x| = n+n = 2n > n$

$\rightarrow uv = a^n, w = b^n$, $uv = a^{n/2}, w = a^{n/2} b^n$

$uv = a^4, w = a^{n-4} b$.

$uv = a, w = a^{n-1} b^n$.

All can be cases

Instance of palindrome could be a'an, a'n b a', etc

* Division which we can't conclude can not be taken as instance

$uv = a^n, w = b^n$.

$v = a^k$ where $k \geq 1$

$u = a^{n-k}, v = a^k, w = b^n$.

Should be part of language $uv^i w \in L$ for all $i \geq 0$

$(a^{n-k})(a)^{ik})(b^n)$ Find i which contradicts

let i=2

$= (a^{n-k}) a^k a^n = a^n b^n$.

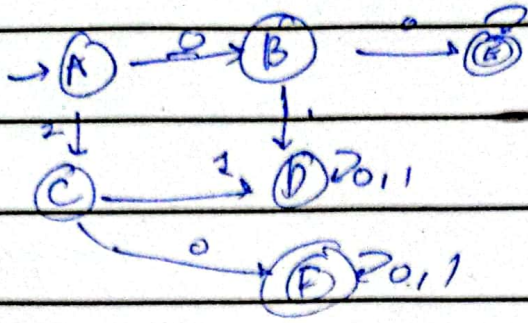
let i = 2.

$= (a^{n-k})(a^{2k})(b^n)$

$= a^{n+k} b^n$. $\therefore a^{n+k} b^n \notin L \Rightarrow$ Assumption was wrong

not accepted by $\rightarrow$ L is non-regular

Method 2 for minimization:



1) Create transition table.
2) If a state is not reachable from initial remove it.
3) Form two groups one of non-final state, one of final state.
4) Check each state with other state in both groups and check their transition goes to same group for both 0 & 1 they can stay together else separate them.
5) Repeat the step for newly created groups. Do this until no new group is created.

Que:-

Every state is reachable from initial

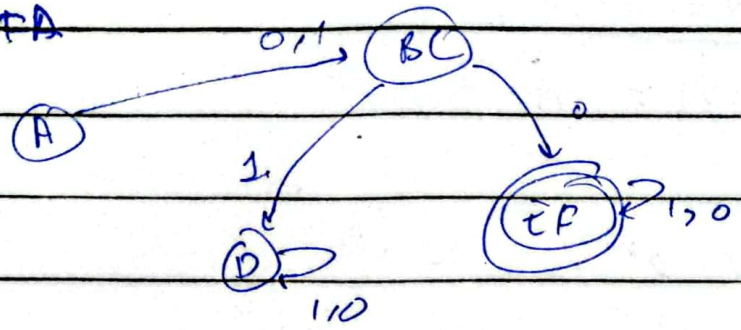
	0	1
A	B	C
B	E	D
C	F	D
D	D	D
E	E	E
F	F	F

Date: _____ Day: _____

- {ABCD} {EF}
- {AD} {BC} {EF}
- {A} {D} {BC} {EF}

→ First check A with B, C, D. Then B with C, D and then C with D. If conflict occurs separate them.

Min. DFA



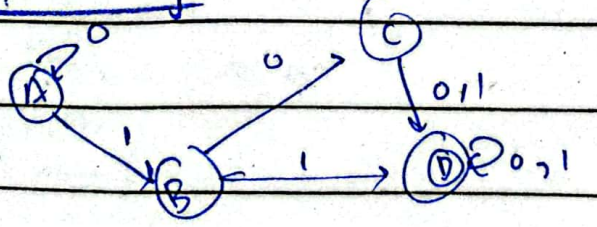
Another example.

Suppose B, A was separated from A because they go to separate. Now, we also check B and C if they go to diff state we separate again.

{a b c} → {a} {bc}.

{a} {b} {c}

Sample eg:



	0	1
A	A	B
B	D	C
C	D	D
D	D	D

Sol: {ABC} {D}

→ {A} {BC} {D}

→ {A} {B} {C} {D}

Let $i = n+1$.

$= n + (n+1-1)|V|$

$= n + n|V| = n(1 + |V|)$. CL.

(n: prime; $(1+|V|)$: Any no.)

= prime * any no = not prime

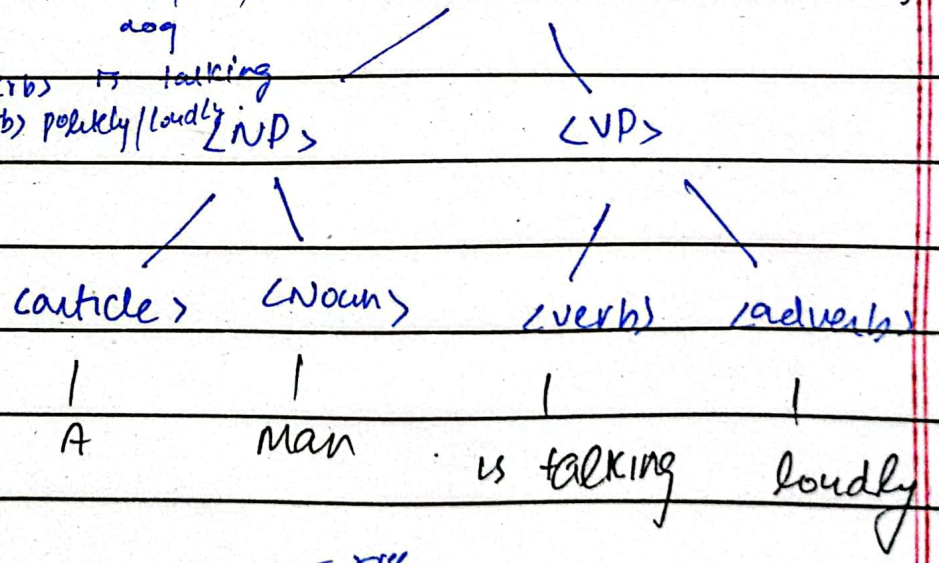
Context Free Language:

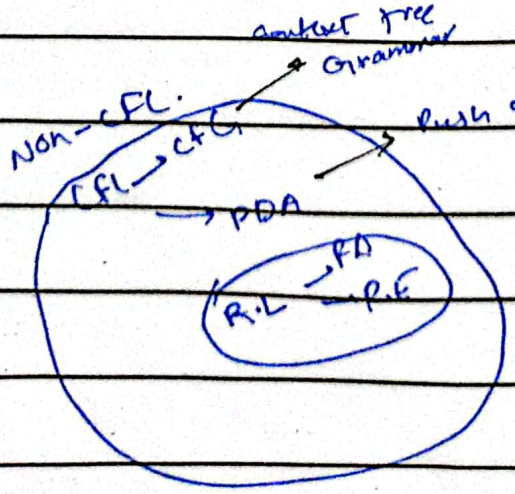
Variables (V) → Sentence, NP, VP, Article, Noun, Verb, Adverb

1. <Sentence> → <NP><VP>
2. <NP> → <article><Noun>
3. <VP> → <verb> / <adverb>
4. <article> → a / the
5. <Noun> → man/cat/dog
6. <verb> → is talking
7. <adverb> → politely/loudly

each is a production

Here S → starting variable





- Every R.L is CFL. — also + some accept non-regular language

PDA — extra memory available

CFG

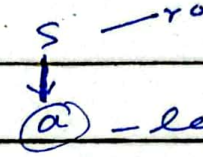
Date: _____ → Draw tree for each of asked string. Multiple can also exist for one.
Day: _____ • Tree exists for each string

CFG (S, V, Σ, P)

S — starting variable
V — set of variables
Σ — terminal / Input Alphabet set
P — set of production.

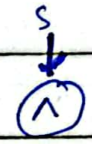
① $L_1 = \{ x \mid x \in \Sigma^* \text{ & } x = a \}$

$S \to a$



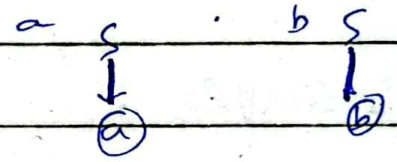
② $L_2 = \{ x \mid x \in \Sigma^* \text{ & } x = \Lambda \}$

$S \to \Lambda$



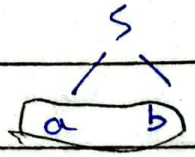
③ $L_3 = \{ x \mid x \in \Sigma^* \text{ & } x = a \text{ or } b \}$

$S \to a \mid b$



④ $L_4 = \{ x \mid x \in \Sigma^* \text{ & } x = ab \}$

$S \to ab$

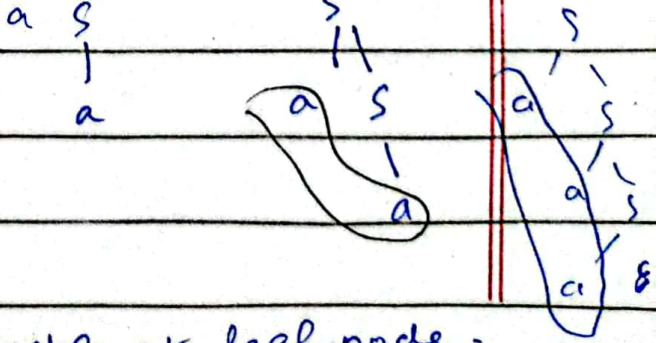


S can can produce a · b

⑤ $L_5 = \{ x \mid x \in \Sigma^* \text{ & } x = a^+ \}$

these two can produce all element of the language

$\begin{cases} S \to a \\ S \to aS \end{cases}$



• There should n't be a variable at leaf node.

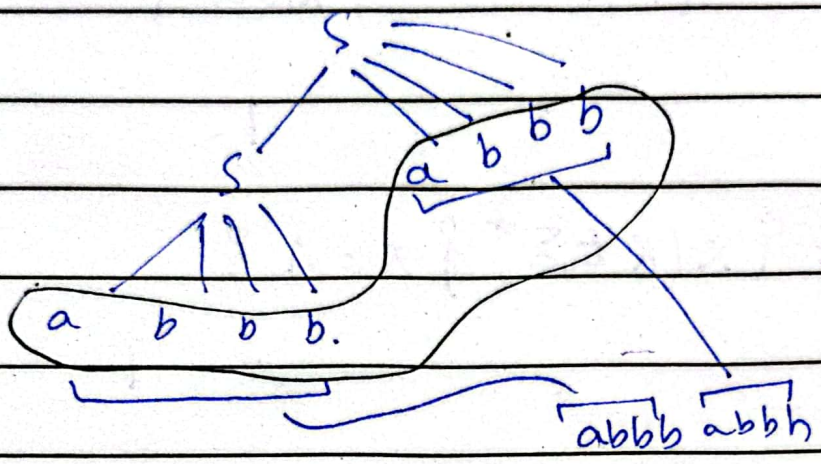
→ Read from left to right

$L_0 = \{abbb\}^*$

- $S \rightarrow \lambda$
- $S \rightarrow abbb$
- $S \rightarrow abbbS$ — right recursion

or

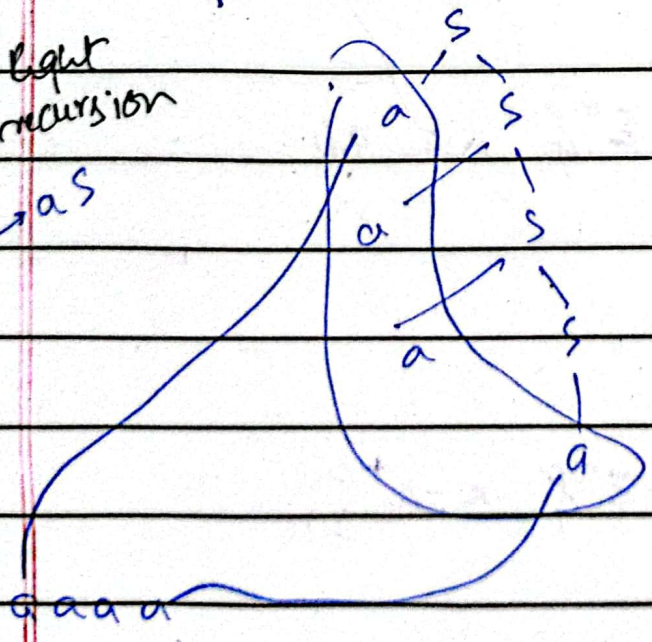
$S \rightarrow Sabbb$ — left recursion



Example :

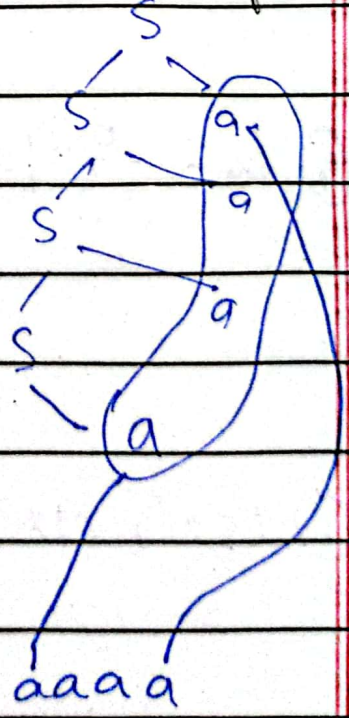
$S \rightarrow aS$

Right recursion



$S \rightarrow Sa$

Left recursion:



∴ we only need to write Grammar. & we are not told to draw tree.

Date: _____ Day: _____

$L_6 = \{ x / x \in \Sigma^* \ \& \ x = (a+b)^* \}$.

$S \to \Lambda$

$S \to a / b$

$S \to aS / bS$

$S \to \Lambda$

$S \to aS / bS$.

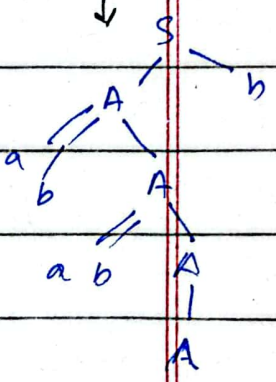
$L_7 = (ab)^* b$.

$S \to Ab$

$A \to \Lambda$

$A \to abA$.

ab abb



$L_8 = (ab + bb)^*$

$S \to \Lambda$

$A \to ab$

$B \to bb$

$S \to AS / BS$

L_9 ends with ab. — $(a+b)^* ab$.

$S \to A\,ab$

$A \to \Lambda / aA / bA$.

Date

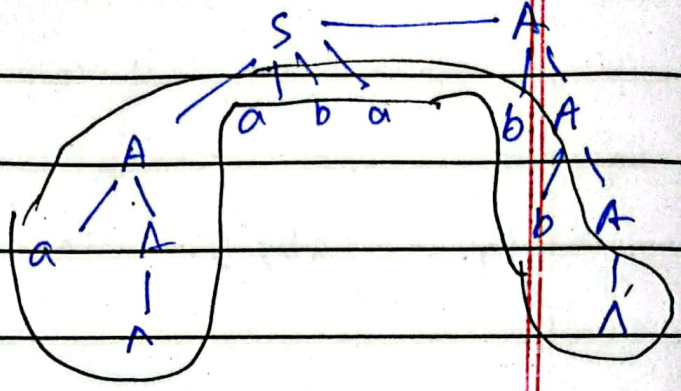
Page

L6 has aba $(a+b)^* aba (a+b)^*$.

$S \rightarrow AabaA$

$A \rightarrow \Lambda | aA | bA$

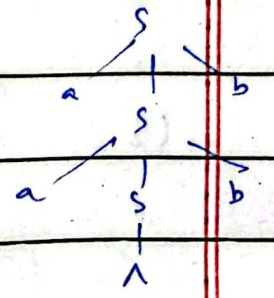
aababb



$X = a^n b^n$,

$S \rightarrow \Lambda$

$S \rightarrow aSb$



~~S → aSb~~

~~S → aa~~

~~A → aA~~

~~B → bB~~

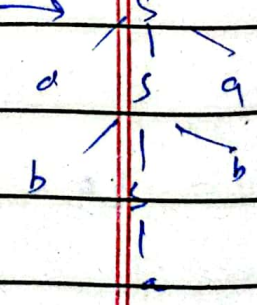
x □ palindrome ababa →

$S \rightarrow \Lambda | a | b$

$S \rightarrow aSa | bSb$

~~S → aSb~~

~~E → aEb~~.



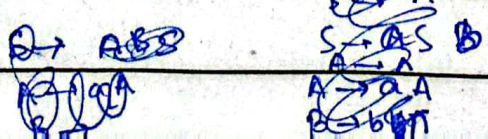
$S \rightarrow AB$

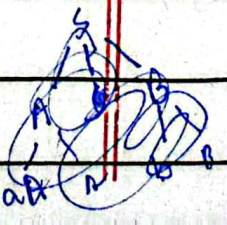
$A \rightarrow aAb | \Lambda$

$B \rightarrow bB | b$

$a^i b^{i+m}$, $m \geq 1$

$X = a^i b^k$, $k > i$





L_1 → CFL

$G_1(S_1, V_1, \Sigma_1, P_1)$,

L_2 → CFL

$G_2(S_2, V_2, \Sigma_2, P_2)$

(1) • $L_3 = L_1 \cup L_2$.

CFL ($G_3(S_3, V_3, \Sigma_3, P_3)$

this complete is a production

$S_3 \rightarrow S_1 \mid S_2$

$V_3 \rightarrow V_1 \cup V_2 \cup S_3$

$\Sigma_3 \rightarrow \Sigma_1 \cup \Sigma_2$

$P_3 \rightarrow \{P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 \mid S_2\}\}$

Check ↓ Ambigous Grammar

(1) $S \rightarrow aS \mid a \mid \wedge$

(2) $S \rightarrow aS \mid Sa \mid a$

(3) $S \rightarrow aSb \mid aSbb \mid \wedge$

(4) $S \rightarrow aSb \mid abS \mid \wedge$

(5) $S \rightarrow S+S \mid S \cdot S \mid S^* S$

$S \rightarrow a \mid b$

(2) $L_4 = L_1 L_2$.

$G_4(S_4, V_4, \Sigma_4, P_4)$

$S_4 \rightarrow S_1 S_2$.

$V_4 \rightarrow V_1 \cup V_2 \cup S_4$.

$\Sigma_4 \rightarrow \Sigma_1 \cup \Sigma_2$

$P_4 \rightarrow (P_1 \cup P_2 \cup \{S_4 \rightarrow S_1 S_2\})$.

(3) $L_5 = L_1^*$

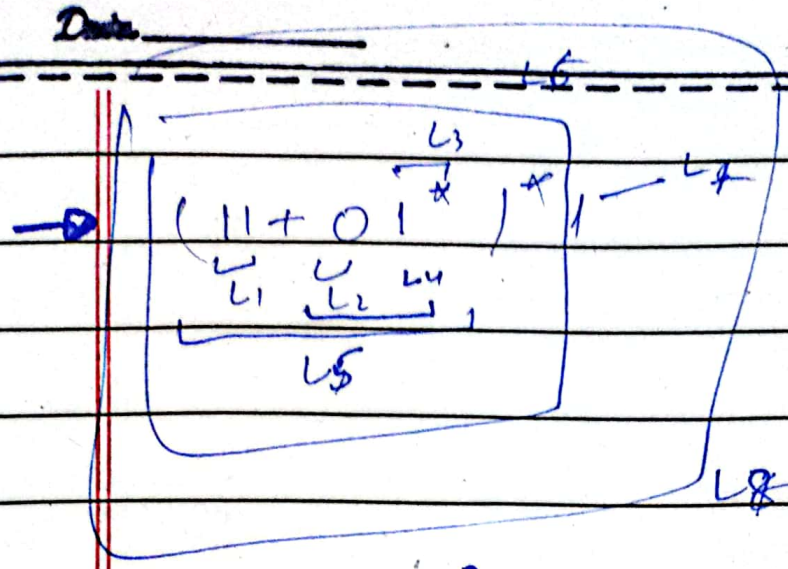
$G_5(S_5, V_5, \Sigma_5, P_5)$.

$S_5 \rightarrow S_1 S_5 \mid \wedge$.

$V_5 \rightarrow V_1 \cup S_5$

$\Sigma_5 \rightarrow \Sigma_1$

$P_5 \rightarrow P_1 \cup \{S_5 \rightarrow S_1 S_5 \mid \wedge\}$.



Generating Grammar from regular expression.

$S_1 \rightarrow 11.$

$S_2 \rightarrow 0$

$S_3 \rightarrow 1S_3 | \Lambda.$

$S_4 \rightarrow S_2 S_3$

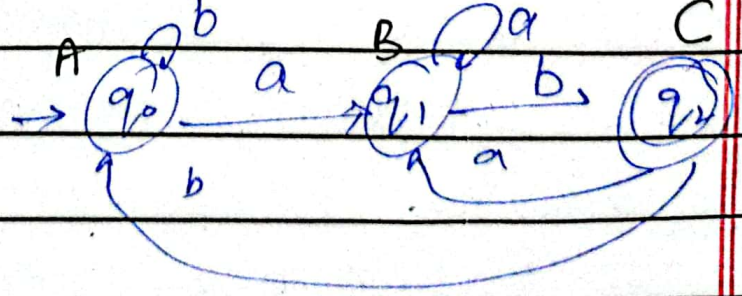
$S_5 \rightarrow S_1 | S_4.$

$S_6 \rightarrow S_5 S_6 | \Lambda..$

$S_7 \rightarrow 1$

$S_8 \rightarrow S_6 S_7.$

Generating grammar from FA



If FA is DFA. Rename all states

Date: _____ Day: _____

if we get a on A we go to B

$A \to aB \mid bA$

$B \to aB \mid bC$

$C \to aB \mid bA$.

(i) $B \to b$ or $C \to \Lambda$ ← To stop recursion (going to C final state every time we get b on B)

Imp: also

* If another state was going to final state we will also do (i) for that state

Regular Grammar:

Variable → terminal Variable $X \to aX_1$

Variable → terminal. $X \to a$.

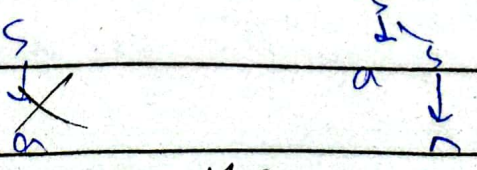
Ambigous Grammar: → After removing ambiguity language should not be changed.

↳ If a string has multiple trees.

* we remove ambiguity. After removing, we should get one tree of string.

(1) • $S \to aS \mid a \mid \Lambda \to aS \mid \Lambda$. non-ambigous.

Ambigous



we remove this

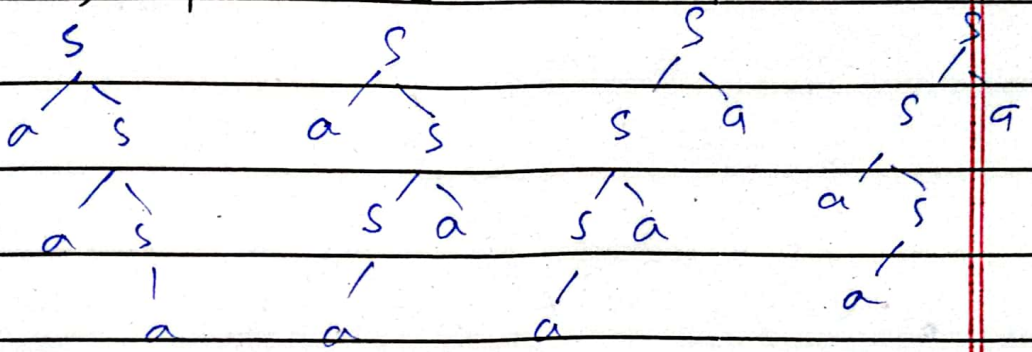
Grammar which is both right, left recursive is always ambiguous.
others can still be ambiguous.

Date _____ Page _____

(2) • $S \to as \mid sa \mid a$

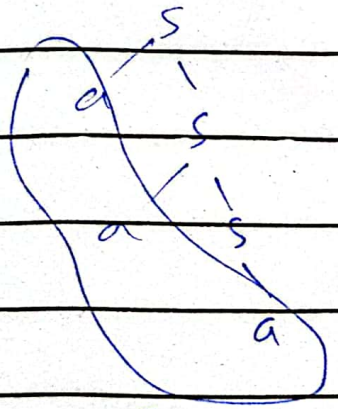
aaa ∈ string

→ 4 possible trees



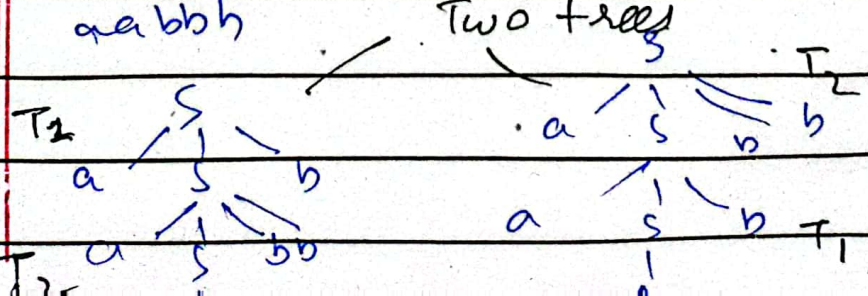
✗ if we remove sa, we will get only one tree

$S \to as \mid a$



(3) $S \to \overset{T_1}{\boxed{aSb}} \mid \overset{T_2}{\boxed{aSbb}} \mid \wedge$

aabbb — Two trees



Date: _____ Day: _____

→ We should not have a way, to intermix T_1, T_2 → we should not be able to first produce T_1, then T_2 then T_1. We should force it, to not come back to produce T_1 after T_2.

T_1

- $S \rightarrow aSb \mid A$ ← First produce equal no of a's and b's then move on to producing two b against one a.

  $A \rightarrow aAbb \mid \wedge$.

  T_2

Now, T_1 can't be produced after T_2

(4) $S \rightarrow asb \mid abS \mid \wedge$

→ Transition, which we don't write, and come is rejected implicitly in both DPDA, NPDA

Date _____ Page _____

Push Down Automata

- We can draw both DFA & NFA for PDA

Language

Regular — Non-regular

Regular → (DFA + Regular)

Non-regular → CFL → Non-CFL

(CFG, PDA)

↳ FA + memory

↳ stack

- we can push/pop in stack.
- If we have $a^n b^n$, we can check given string with help of stack.
  keep pushing a's in stack. when first b arrives start popping a's. If stack becomes empty when no of b's are exhausted it mean string is acceptable.

PDA — 7 tuple $(Q, q_0, \Sigma, \Delta, \delta, \Gamma, Z_0)$

top element

stack alphabet

- for FA $\delta : Q \times \Sigma \to Q$
- for PDA $\delta : Q \times (\Sigma \cup \{\wedge\}) \times \Gamma \to Q \times \Gamma^*$

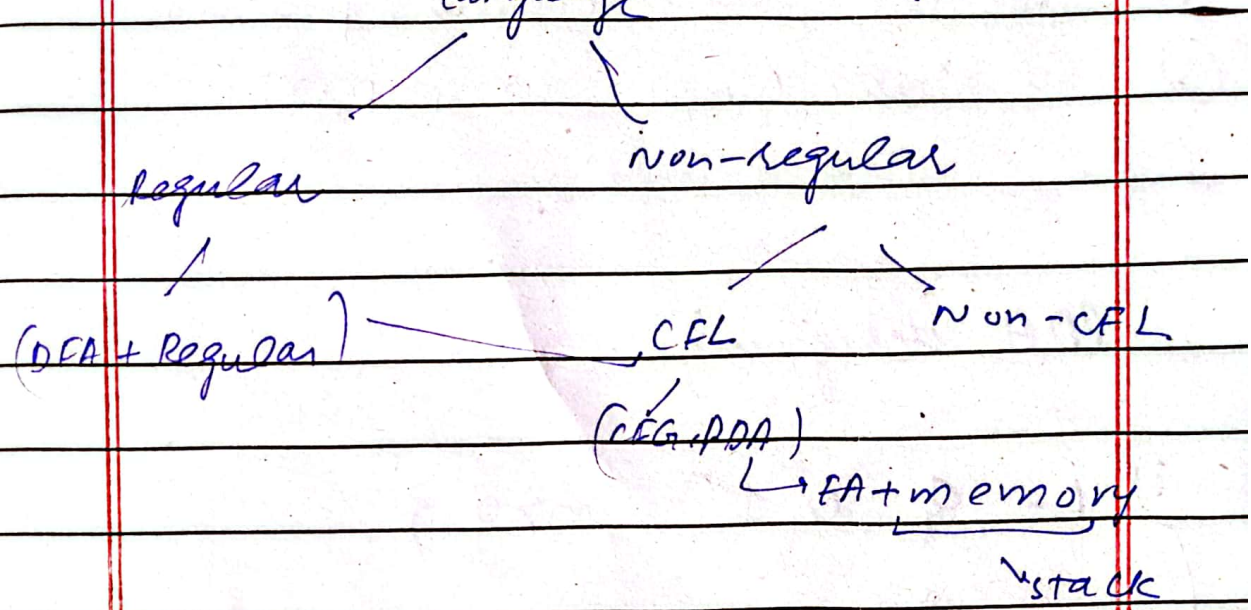
↓ pushed in stack

→ Transition, which we don't write, and come is rejected implicitly in both DPDA, NPDA

Date _____ Page _____

Push Down Automata

- We can draw both DFA & NFA for PDA



- We can push/pop in stack.
- If we have $a^n b^n$, we can check given string with help of stack, when first b keep pushing a's in stack. when first b arrives start popping a's. If stack becomes empty when no of b's are exhausted it mean string is acceptable.

PDA — 7 tuple $(Q, q_0, \Sigma, \Delta, \delta, \Gamma, Z_0)$

(Z_0: top element; Γ: stack alphabet; Δ: pushed in stack)

- for FA $\delta : Q \times \Sigma \to Q$
- for PDA $\delta : Q \times (\Sigma \cup \{\wedge\}) \times \Gamma \to Q \times \Gamma^*$

Date: _____ DPDA → $\delta(q, \wedge, x) \neq \phi$ then $\delta(q, a, x) = \phi$ Day: _____

① if γ, γ^* equal, no change occured

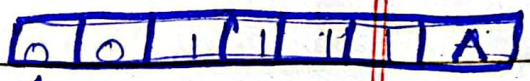
② if γ, γ^* different, we pushed γ

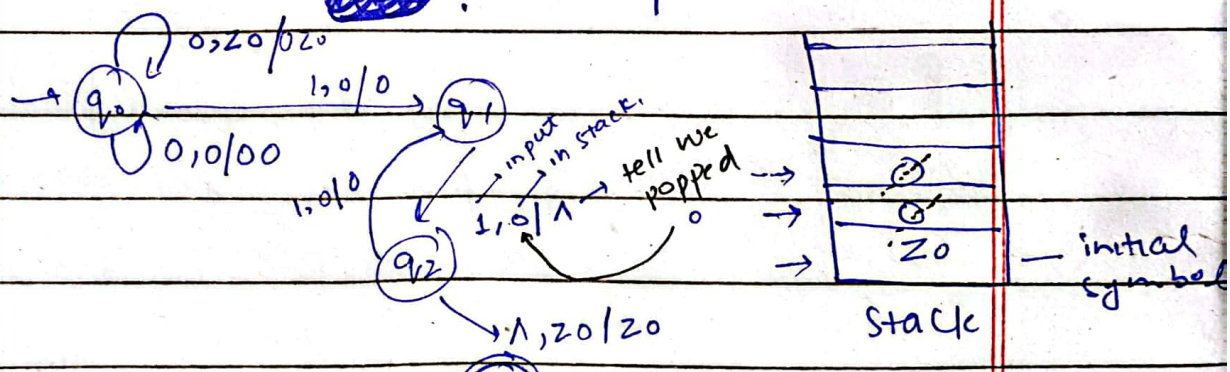
③ if γ^* is $\wedge$, we poped γ

PDA designing:

i) $L = \{ 0^m 1^{2n}, n>0 \}$ 011, 00111, 000111111

$(Q, \Sigma, \Gamma, \delta, q_0, z_0, A)$ 0,1 ⊗z0





① If we get 0 and have z_0 in stack $(0, z_0)$, pop z_0 then push $0, z_0$ in stack. (both)

② Then we have 0 in stack and 0 as input, pop 0 then simply push 0,0 too. (both) (top element 0 in this step)

③ Then we get 1 as input, we go to q_1 state.

④ We get 1 again, we should pop one zero

⑤ Then Step 3,4 repeated again.

⑥ Atlast we get $\wedge$, in stack we had z_0 we go to final state

Transition functions

$q_0, 0, z_0 \to 0z_0$ ($r \to 0$, $r \to 00$)

$q_0, 0, 0 \to 00$ — pushed

$q_0, 1, 0 \to q_1, 0$ — No change occured

$q_1, 1, 0 \to q_2, \Lambda$ — poped.

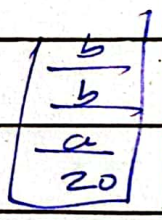
$q_2, 1, 0 \to q, 10$ — No change

$q_2, \Lambda, z_0 \to q_f, z_0$

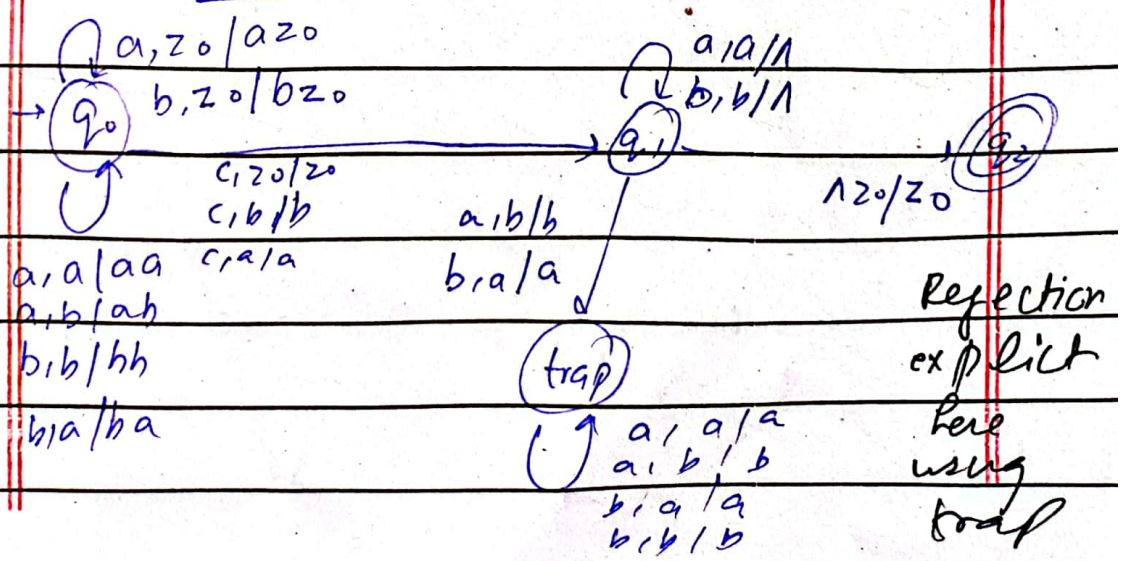
ii) $L = \{ x | xcx^r , x^r$ reverse of $x \}$

let string abbc bba.

Push a, b, b till c arrives then started popping and comparing, we get to know abbc bba is palindrome



DFA.



Date: _____ Day: _____

Now, $(q_0, bacab, z_0)$

remaining input — instack

$(q_0, bacab, bz_0) \rightarrow (q_0, cab, abz_0)$

$\rightarrow (q_1, ab, abz_0) \rightarrow (q_1, b, bz_0) \rightarrow (q_1, \wedge, z_0) \rightarrow (q_2, \wedge, z_0)$

accepted

(q_0, acb, z_0)

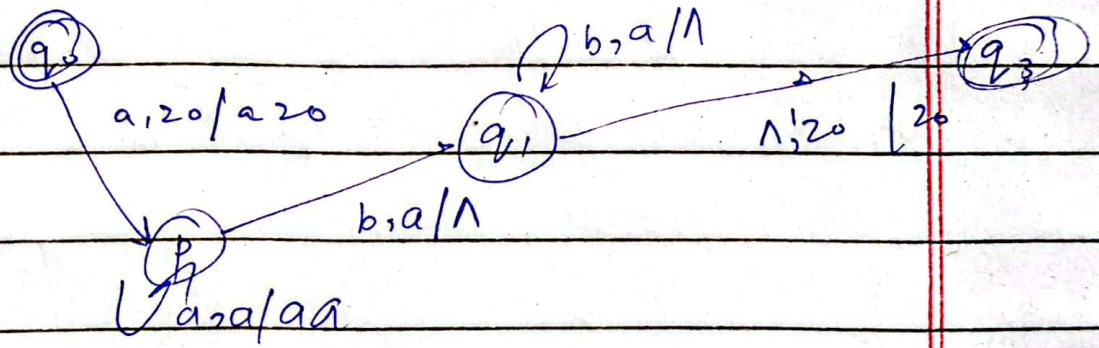
$(q_0, acb, az_0) \rightarrow (q_1, cb, az_0)$

$\rightarrow (trap, \wedge, az_0)$ reject

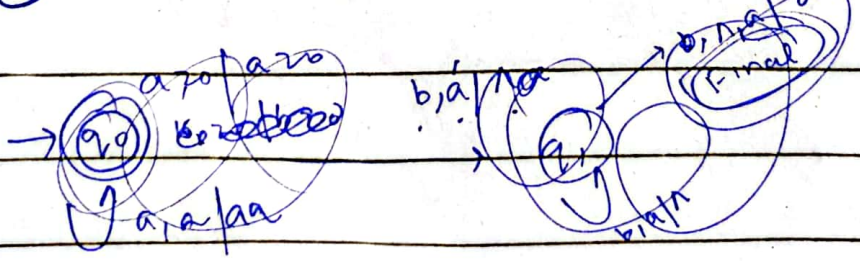
input exhausted, but Elements are still in stack.

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

Ans:



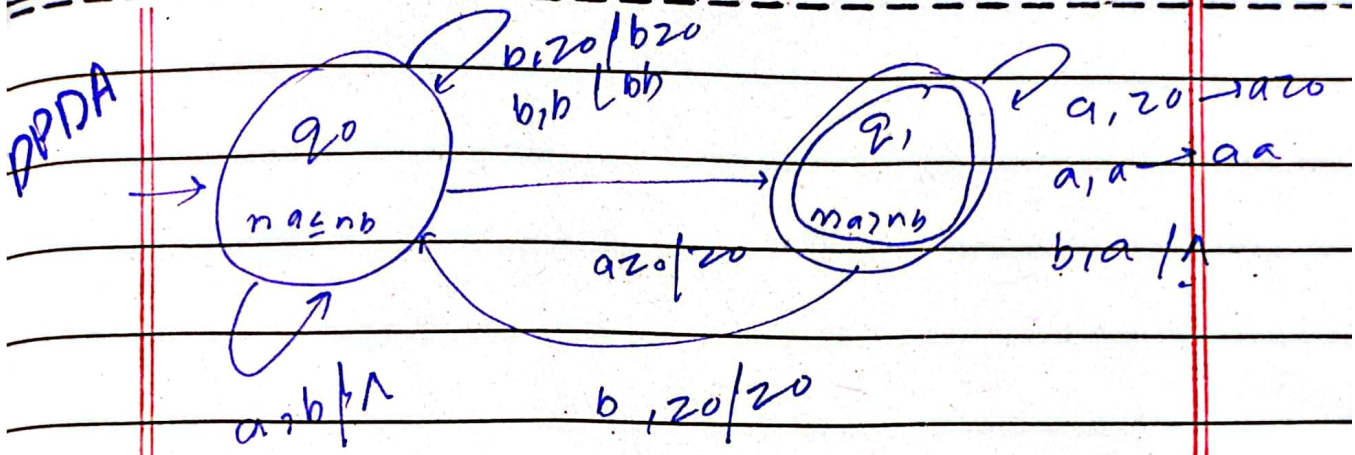
(1) $n_a \geq n_b$



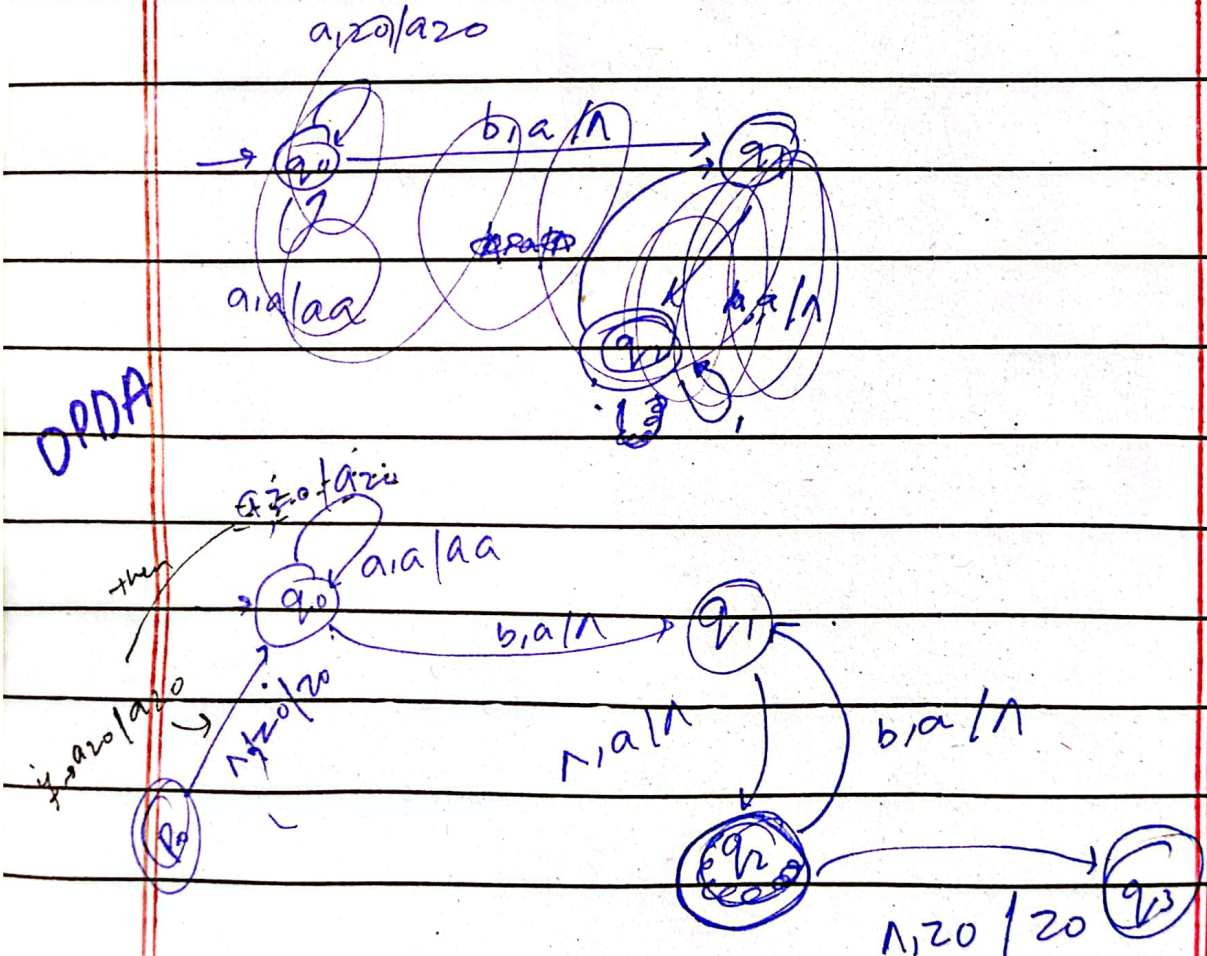
Date

Page

DPDA



(2) $a^i b^j$, $j = 2i$, $i, j \geq 0$



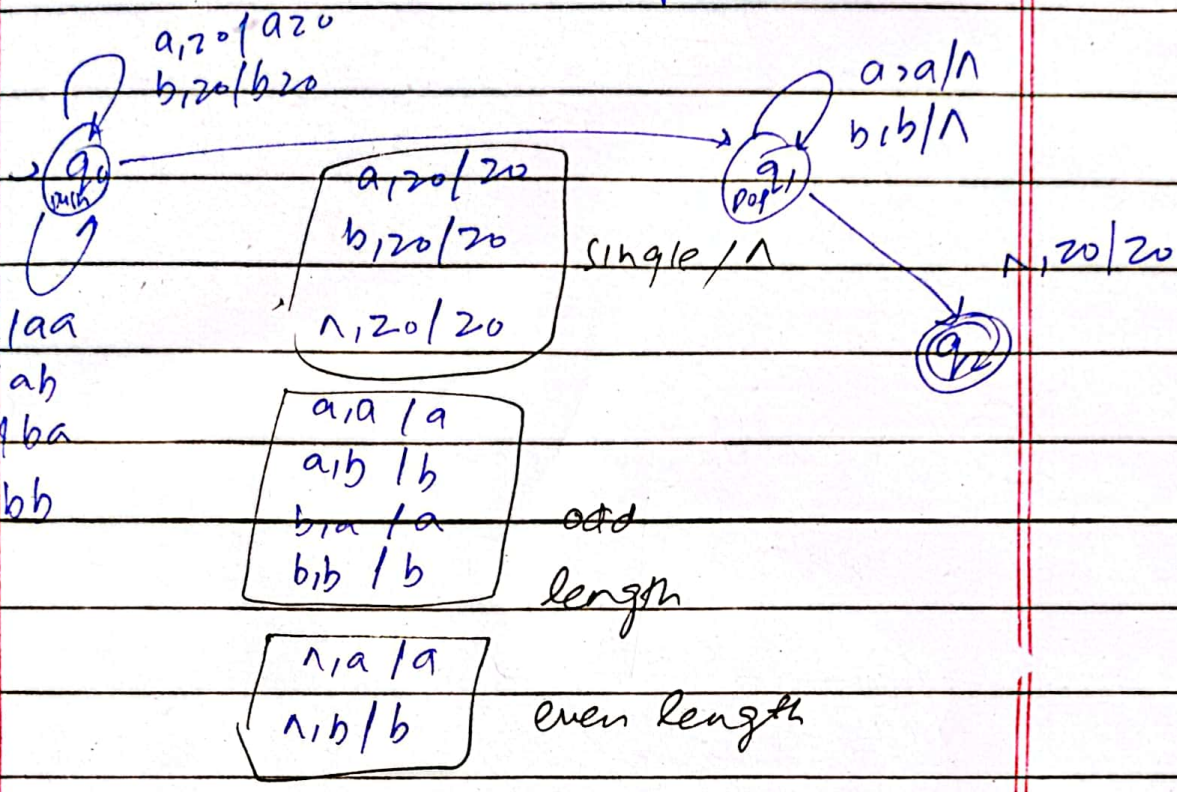
(3) Palindrome.

NDPDA

We can't create DPDA for this

Date: _____ push ab | ba pop. Day: _____

→ ab a ba → pop
push ↓ no change

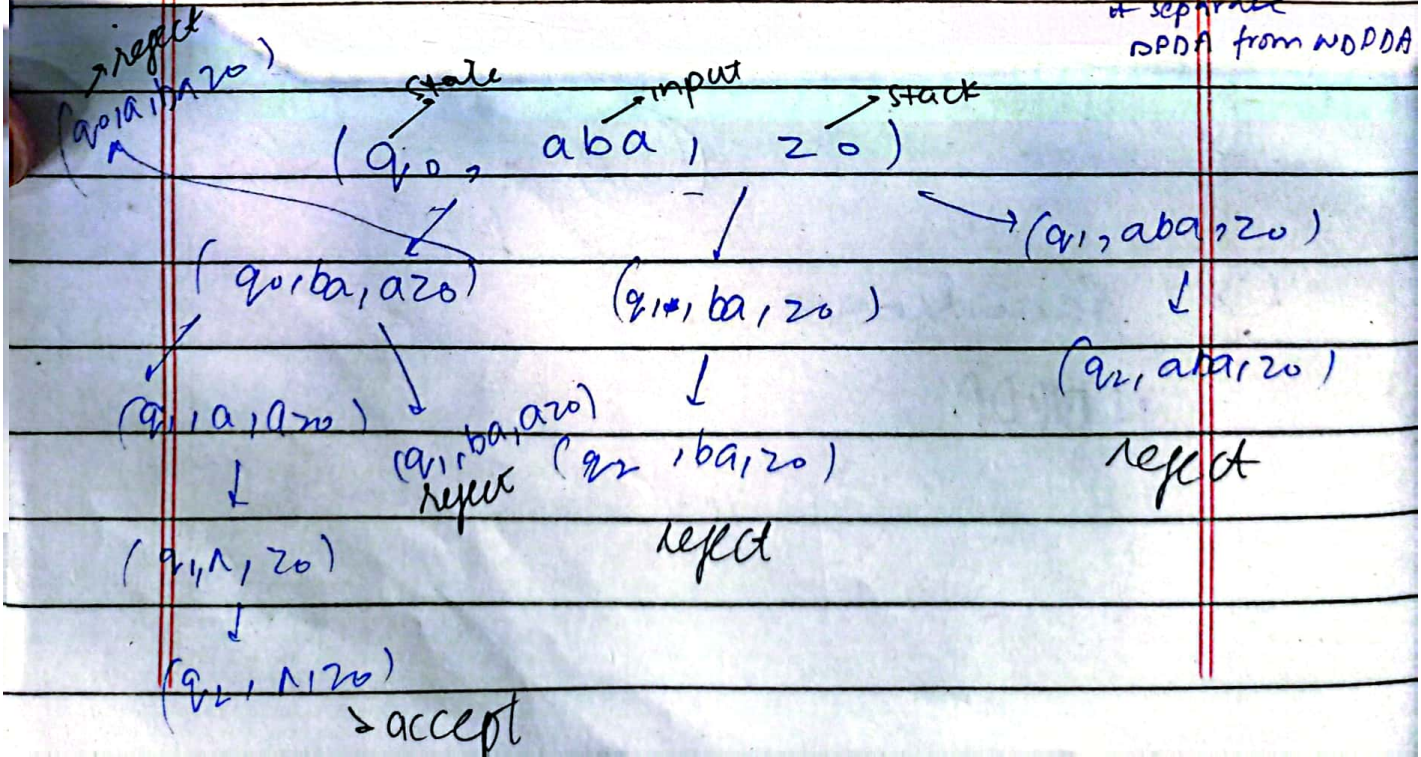


In DPDA, if top of stack is same, initial state of transition is same, then we can't do two transitions on it, such that in one transition we consume input and in one we don't

a,Z0/Z0
∧,Z0/Z0
} transitions from q_0 to q_1

make this PDA NDPDA

this is what a separate DPDA from NDPDA

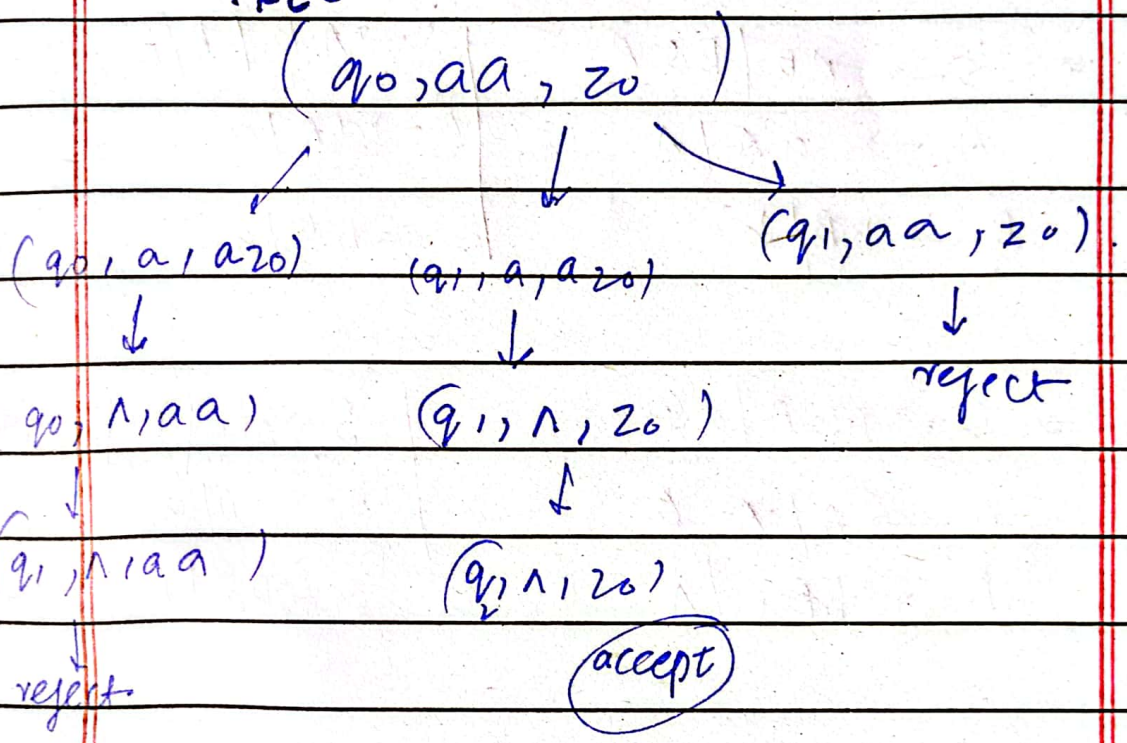


Date _____ Day _____

NOTE:

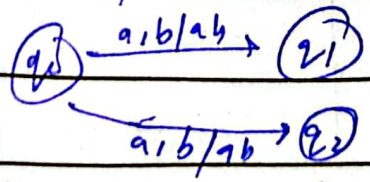
- Your PDA tree should reject a non-accepting string from all branches And accept a accepting string from one branch.

TREE

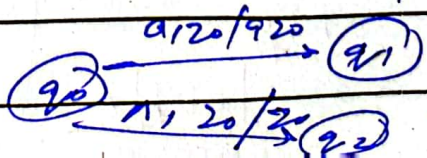


NDPDA done by:-

(1) If two or more edges labeled with same input & same stack symbol



(2) when a stated have 2 edges with same stack symbol and one input symbol is ∧. → transitions



Date: _____ Day: _____

variable → 2 variable
variable → terminal

CNF:

Remove — useless production
② ∧ - production
③ unit - production.
④ convert right side of string of variables
⑤ Chop right to form production of form

not

→ If null acceptable, there should be no production containing null
→ If acceptable, then it should only be in starting variable, removed from other productions

Null - Removed

- $S \to AB \mid BSA$
  $A \to aA \mid \wedge$
  $B \to bB \mid b$

  $S \to AB \mid B \mid BSA \mid BS$
  $A \to aA \mid a$
  $B \to bB \mid b$

- $S \to ABA \mid aSB$
  $A \to aA \mid \wedge$
  $B \to bBa \mid \wedge$.

  Replace ∧ one by one only each variable which contained null

  $S \to ABA \mid BA \mid AB \mid AA \mid A \mid aSB \mid aS$
  $A \to aA \mid a$
  $B \to bBa \mid ba$

~~$A \to aAA$~~

- $S \to CAB$
  $A \to AaA \mid \wedge$
  $C \to C$
  $B \to bB \mid b$

  $S \to CAB \mid CB$
  $A \to AaA \mid aA \mid Aa \mid a$
  $C \to C$
  $B \to bB \mid b$

- $S \to ACD$
  $A \to aAb \mid \wedge$
  $D \to aDa \mid bDb \mid A$
  $C \to aC \mid a \mid \wedge$

  $S \to \wedge$
  $S \to ACD \mid CD \mid AC \mid AD \mid A \mid C \mid D$
  $A \to aAb \mid ab$
  $D \to aDa \mid bDb \mid aa \mid bb$
  $C \to aC \mid a$

$S \to CAB$	$S \to CAB \mid CA \mid CB \mid C$
$A \to AaA \mid \Lambda$	$A \to AaA \mid aA \mid Aa \mid a$
$C \to C$	$C \to C$
$B \to bB \mid b \mid A$	$B \to bB \mid b \mid A \mid \Lambda$

A to be replaced by, null one time, one time it will remain same

$B \to bB \mid b \mid A$

→ Useless production.

We need to see, if we get a useless production. after removing Λ.

such as $B \to bB \mid B$ → useless

→ Unit production.

variable → Single variable

$B \to A$ → replace by what A is producing

$B \to AaA \mid aA \mid Aa \mid a$

$B \to bB \mid b$.

$S \to ACD$

$A \to an \mid b \mid \Lambda$

$D \to aDa \mid bDb \mid \Lambda$

$C \to aC \mid a \mid \Lambda$.

Date: _____ Day: _____

① NO useless

② S → ACD | CD | AD | AC | D | C | A | ∧

A → aAb | ab

D → aDa | bDb | aa | bb ;

C → aCa

③ Unit production in starting variable

D A

S → ACD | CD | AD | AC | aDa | bDb | aa | bb | aC | a | aAb | ab | ∧

C

④ Convert right side to string of variables, or single terminals. - single a, b in production

variables { Xa → a, Xb → b } input alphabet set

Xb D Xb |

S → ACD | CD | AD | AC | XaDXa | XaXa | XbXb |

Xac | a | XaAXb | XaXb | ∧

↓ it was single terminal

A → XaAXa | XaXb

D → XaDXa | XbDXb | XaXa | XbXb

C → XaC | a

Date _____ Page _____

(5) Produce two variables for terminal

Productions, which are not fulfilling

Step 5 chop ~~the~~ on right

- $S \rightarrow ACD$

  $T_1 \rightarrow CD$

  $S \rightarrow AT_1$ — Now in CNF

- $S \rightarrow X_a D X_a$

  $T_2 \rightarrow X_a D$

  $S \rightarrow T_2 X_a$

- $S \rightarrow X_b D X_b$

  $T_3 \rightarrow X_b D$

  $S \rightarrow T_3 X_b$.

- $S \rightarrow X_a A X_b$

  $T_4 \rightarrow AX_b$

  $S \rightarrow X_a T_4$

- $X_a \rightarrow a$
- $X_b \rightarrow b$

All productions ^ NF now)

→ Can, reuse productions.

Date: Day:

Parser: better than drawing tree, bcz in tree we need to draw tree for each string, or parser, we write code which will generate table respective to string.

Bottom-up parser CYK (Cocke Young Kasami)

Algorithm :-

CFG should be in CNF

CFG → Parser

X →

CNF :-

S → XY

X → XA | a | b

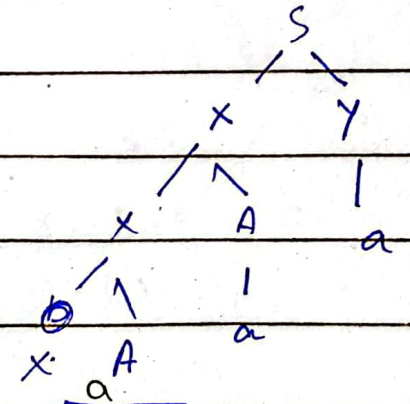
Y → AY | a

A → a

starts with a, b | ends with a

babaa ∈ L(CFG).

can't produce



			a	X,Y,A (1,5)
		b	X,Y,A (4,4)	S,X,Y (4,5)
	a	X (3,3)	S,X (3,4)	S,X (3,5)
b	X,Y,A (2,2)	Ø (2,3)	Ø (2,4)	Ø (2,5)
X	S,X	Ø	Ø	Ø
(1,1)	(1,2)	(1,3)	(1,4)	(1,5)

→ Can take upper or lower half including diagonal.

Discard other half