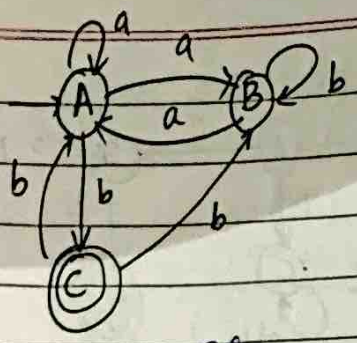
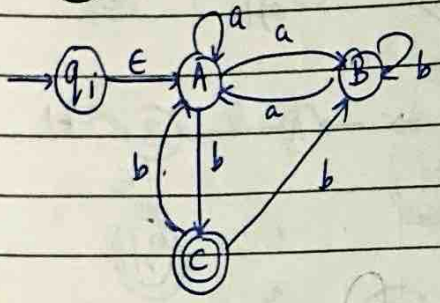
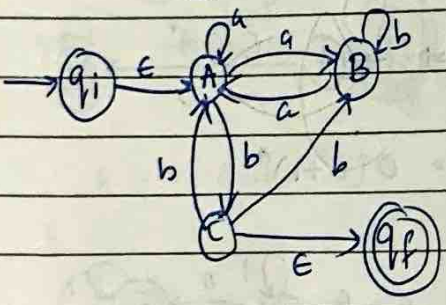
(5)



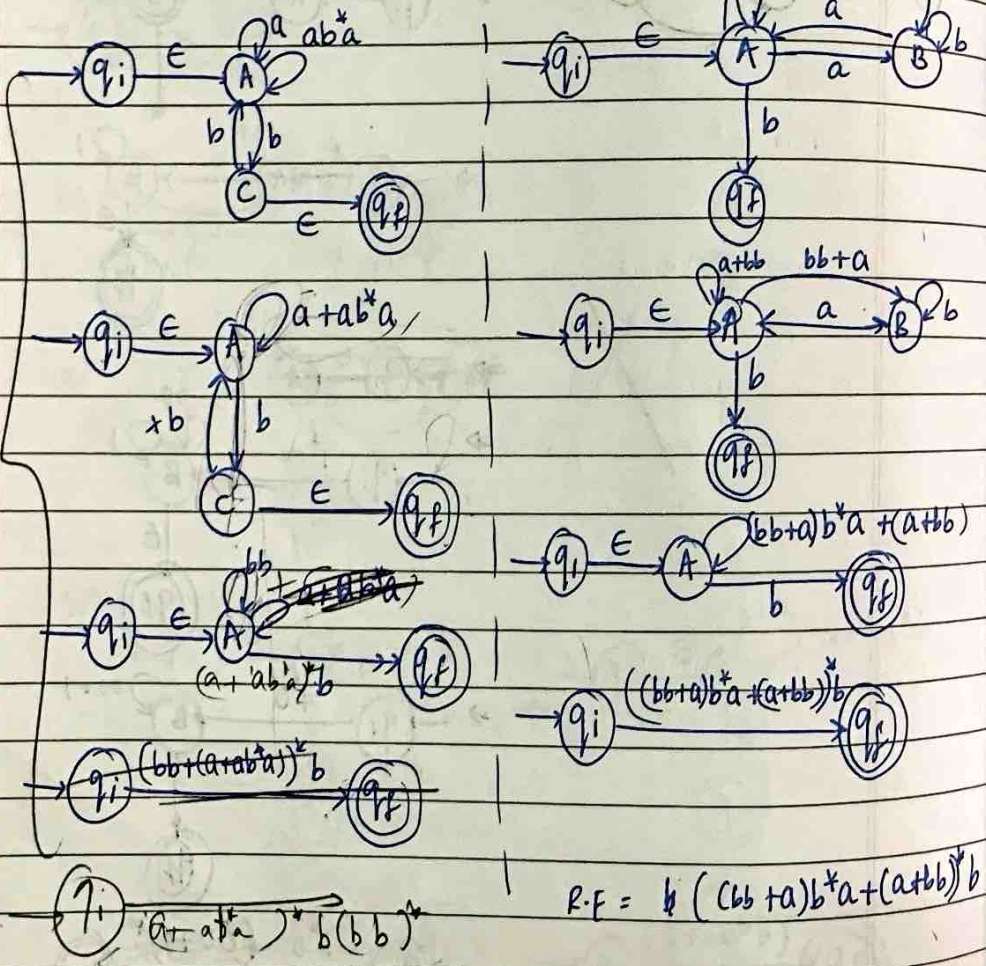
(1)



(2)



(4)



doubt

20/5/24

Identity Rules of Regular expression

1) $\phi + R = R$
   where R is a regular expression
   where ϕ is an empty set
2) $\phi \cdot R = R \cdot \phi = \phi$
3) $\epsilon \cdot R = R \cdot \epsilon = R$ where ϵ is a null string
4) $\epsilon^* = \epsilon$ & $\phi^* = \epsilon$
5) $R + R = R$ eg: $a + a = a$
6) $R^* \cdot R^* = R^*$ eg: $(ab)^*(ab)^* = (ab)^*$
7) $(R^*)^* = R^*$ eg: $(ab)^{**} = (ab)^*$
8) $RR^* = R^*R = R^+$
9) $\epsilon + RR^* = \epsilon + R^*R = \epsilon + R^+ = R^*$
10) $(P+Q)R = PR + QR$ eg: $(a+b)c = ac + bc$
11) $(PQ)^* P = P(QP)^*$
12) $(P+Q)^* = (P^* \cdot Q^*)^* = (P^* + Q^*)^*$

Arden's Theorem

If P and Q are two regular expressions over Σ, and if P doesn't contain ϵ, then the following equation in R given by

$$R = Q + RP$$

has a unique solution i.e.

$$R = QP^*$$

Proof:

$R = Q + RP$

$= Q + QP^*P$ $\because R = QP^*$

$= Q(\epsilon + P^*P)$ $\because \epsilon + R^*R = R^*$

$R = QP^*$

Prove that $R = QP^*$ is the unique solution.

$R = Q + RP$

$= Q + [Q + RP]P$

$= Q + QP + RP^2$

$= Q + QP + (Q+RP)P^2$

$= Q + QP + QP^2 + RP^3$

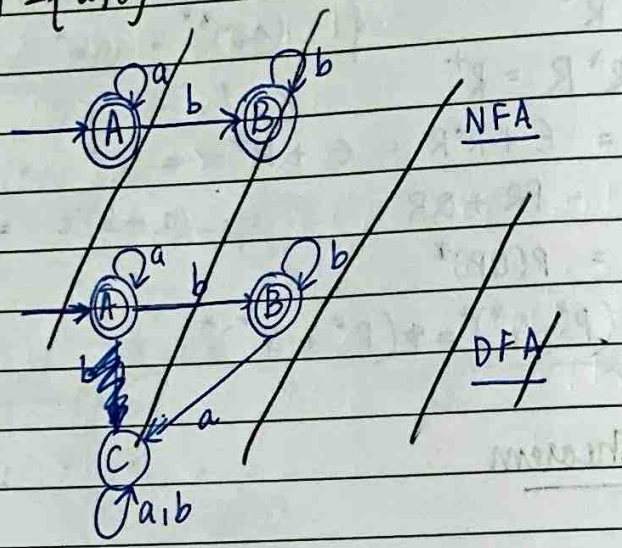
$= Q + QP + QP^2 + \ldots\ldots + QP^n + RP^{n+1}$

$= Q + QP + QP^2 + \ldots + QP^n + QP^* P^{n+1}$

$= Q(\epsilon + P + P^2 + \ldots + P^n + P^* P^{n+1})$

$\boxed{R = QP^*}$

Q] Design a FA for the language $L = \{a^n b^n \mid n \geq 0\}$ over $\Sigma = \{a, b\}$



We cannot design a FA for above language, as there is no enough memory to remember the count value. So it is not a regular language

Regular language

It is regular language iff some finite state machine recognizes it.

eg: $L = \{a^n b^n \mid n \geq 0\}$ is not a regular language.

$L = \{yy \mid y \in (0,1)^*\}$ is not a regular language.

Note: FA has very limited memory

It cannot store or count strings.

21/5/24

Pumping Lemma

It is used to prove that language is not regular.

If substring of a string is repeated many times and if the resultant string is also available in language 'L' then we can say it is regular.

Steps:

1. Consider a language as a regular language.
2. Assume a constant 'C' and select the string 'w' such that $|w| \geq C$ where $|w|$ is length of w
3. Divide 'w' as XYZ such that
   i) $|Y| > 0$
   ii) $|XY| \leq C$
4. Check string XY^iZ belongs to L for $i \geq 0$

(1) Prove that $L = \{a^n b^n \mid n \geq 0\}$ is not a regular language.

Step 1: Assume L is a regular language

Step 2: Let C = 4, w = aabb

Here $|w| \geq C$ i.e $4 \geq 4$.

Step 3: w = aabb

Divide w as XYZ where

X = a $\quad |X| = 1$

Y = ab $\quad |Y| = 2$

Z = b $\quad |Z| = 1$

$\therefore |Y| > 0$ and $|XY| = 3 \leq C$

Step 4: When i = 0, $XY^iZ = XZ$

$= ab \in L$

i = 1 $\quad XYZ = aabb \in L$

i = 2 $\quad XY^2Z = aXYYZ = aababb \notin L$

Hence the given language is not a regular language.

(2) Prove that $L = \{a^i b^j \mid i \leq j\}$ is not a regular language.

$L = \{ab, abb, aabb, aabbb, abbb, \ldots\}$

Step 1: Assume L is a regular language

Step 2: Let $c = 2$ and $w = aabbb$

$\therefore |w| \geq c$.

$5 \geq 2$.

Step 3: Divide w as xyz such that.

$x = a \Rightarrow |x| = 1$

$y = a \Rightarrow |y| = 1$

$z = bbb \Rightarrow |z| = 3$

$\therefore |y| \geq 0 \Rightarrow 1 \geq 0$

$|xy| \leq c \Rightarrow 2 \leq 2$

Step 4: when $i = 0$, $xy^i z = xz = abbb \in L$

$i = 1$, $xyz = aabbb \in L$

$i = 2$, $xy^2z = aaabbb \in L$

$i = 3$, $xy^3z = aaaabbb \notin L$

$\therefore$ The above language is not regular.

(3) Prove that $L = \{a^{2n} \mid n \geq 1\}$ is not a regular language.

$L = \{aa, aaaa, aaaaaa, \ldots\}$

Step 1: Assume L is a regular language

Step 2: Let $c = 3$, and $w = aaaa$

$\therefore |w| \geq c$ i.e $4 \geq 3$

Step 3: Divide w as xyz

$x = a \Rightarrow |x| = 1$

$y = a \Rightarrow |y| = 1$

$z = aa \Rightarrow |z| = 1$

$\therefore |y| > 0 \Rightarrow 1 \geq 0$

$\therefore |xy| \leq c \Rightarrow 2 \leq 3$

Step 4: When $i = 0$, $xy^i z = xz = aaa \notin L$

$\therefore$ The above language is not regular.

④ Prove that $L = \{yy \mid y \in \{0,1\}\}$ is not a regular language.

$L = \{00, 11, 0101, 110110, \ldots\}$

Step 1: Assume L is a regular language

Step 2: Let $c = 3$ and $w = 0101$

$\therefore |w| \geq c \Rightarrow 4 \geq 3$

Step 3: Divide w as XYZ

$X = 0$

$Y = 1$

$Z = 01$

$\therefore |X| \geq 0 \Rightarrow 1 > 0$

$|XY| \leq c \Rightarrow 2 \leq 3$

Step 4: When $i = 0$, $XY^iZ = XZ = 001 \notin L$

$\therefore$ The above language is not regular

Applications of Regular Expressions

[1] Regular expression in UNIX

Notations / meta characters:

1) . → single character

2) [A-Z] → character class for uppercase letters

(A+B+C+ ---- +Z)

· [aeiou] → vowel

(a+e+i+o+u)

3) ? → zero or one

\+ → one or more

* → zero or more

[2] Lexical Analysis

It is the first stage in compiler design.

eg: To identify if it is identify or keyword or value, etc.

```
if: [a-z A-Z][A-Z a-z 0-9]*  return identifier;
    [0-9]+                   return NUM;
    'int' | 'float' | 'char' return KEYWORD;
```

[3] Text Search
o/, like in UNIX text editor
eg: RIT[a-z A-Z 0-9]+
eg: RIT5, RITcs, RITcs03, etc

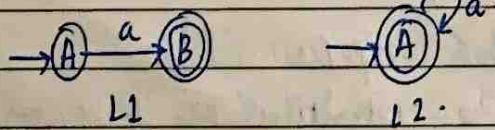
23/5/24 Closure properties of regular language

eg: Set of Natural numbers
$4+5=9 \in N \longrightarrow$ closed
$4*5=20 \in N \longrightarrow$ closed
$4-5=-1 \notin N \longrightarrow$ not closed
$4/5=0.8 \notin N \longrightarrow$ not closed

[1] Union
Set of regular language - {L1, L2, ...}
$L1 \cup L2 \Rightarrow R1 \cup R2 = R3 \longrightarrow L3$.
eg: $R1 = a^*$ $R2 = b^*$
$R3 = R1 + R2 = a^* + b^*$

[2] Concatenation
$L1.L2 \Rightarrow R1.R2 = R3 \longrightarrow L3$
eg: $R1 = a^*$, $R2 = b^*$
$R3 = R1.R2 = a^*b^*$

[3] Kleen Star
$L1 \Rightarrow R1 \Rightarrow R1^* = R2 \longrightarrow L2$.
eg: $R1 = aa$
$R2 = R1^* = (aa)^* a^*$

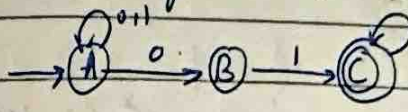


L1 L2.

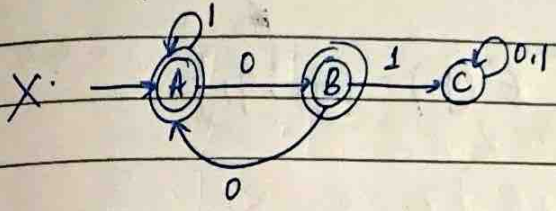
4 Complement

$L1 \rightarrow \overline{L1}$

eg: L1 = { set of strings with 01 as substring }



$\overline{L1}$ - { set of strings without 01 as substring }



i think so

5 Intersection

$L_1 \cap L_2$.

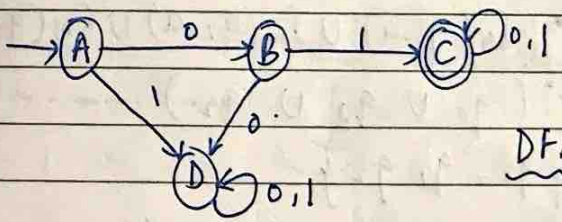
$\Rightarrow \overline{\overline{L_1} + \overline{L_2}}$

$\Rightarrow \overline{\overline{r_1} + \overline{r_2}} \Rightarrow \overline{r_3 + r_4} \Rightarrow \overline{r_5} \Rightarrow L_5$

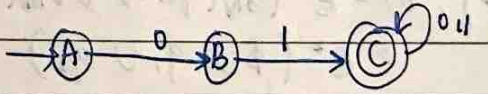
6 Reversal

$L \longrightarrow L^R$

eg: L = { set of strings that begin with 01 }

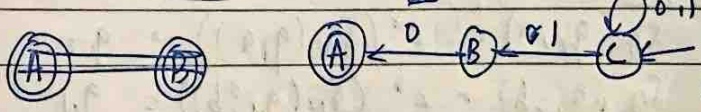


DFA



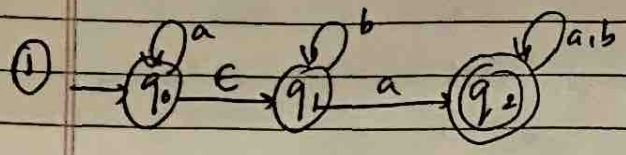
NFA.

intial ~~non final~~ $\leftrightarrow$ final & change directions



reversal.

Convert ϵNFA to DFA



Step 1: Find initial state : ϵ^* (initial state of ϵ NFA) of DFA

$\epsilon^*(q_2) = q_2$ $\quad \epsilon^*(q_0) = \{q_0, q_1\}$

$\epsilon^*(q_1) = q_1$

Step 2: For I/P a, from state $q_0 q_1$

$\delta_D(\{q_0, q_1\}, a) = \epsilon^*(\delta_N(q_0, a) \cup \delta_N(q_1, b))$

$= \{q_0\} \cup \{q_2\}$ $\quad \epsilon^*(q_0 q_2)$

$= \{q_0, q_2\}$ $\quad \{q_0, q_1, q_2\}$

$\delta_D(\{q_0, q_1\}, b) = \epsilon^*(\delta_N(q_0, b) \cup \delta_N(q_1, b))$

$= \epsilon^*(\phi \cup q_1)$

$= \{q_1\}$

Step 3:

$\delta_D(\{q_0, q_1, q_2\}, a) = \epsilon^*(\delta_N(\{q_0, q_1, q_2\}, a)$

$= \epsilon^*(\delta_N(q_0, a) \cup \delta_N(q_1, a) \cup \delta_N(q_2 a))$

$= \epsilon^*(q_0 \cup q_2 \cup q_2)$

$= \{q_0, q_1 q_2\}$

$\delta_D(\{q_0, q_1, q_2\}, b) = \epsilon^*(\delta_N(q_0, b) \cup \delta_N(q_1, b) \cup \delta_N(q_2, b))$

$= \epsilon^*(\phi \cup q_1 \cup q_2)$

$= \{q_1, q_2\}$

Step 4:

$\delta_D(q_1, a) = \epsilon^*(\delta_N(q_1, a)) = q_2$

$\delta_D(q_1, b) = \epsilon^*(\delta_N(q_1, b)) = q_1$

Step 5: $\delta_D(\{q_1,q_2\},a) = \epsilon^*(\delta_N(\{q_1,q_2\},$

$= \epsilon^*(\delta_N(q_1,a) \cup \delta_N(q_2,a))$

$= \epsilon^*(q_2 \cup q_2)$

$= q_2$

$\delta_N(\{q_1,q_2\},b) = \epsilon^*(\delta_N(q_1,b) \cup \delta_N(q_2,b))$

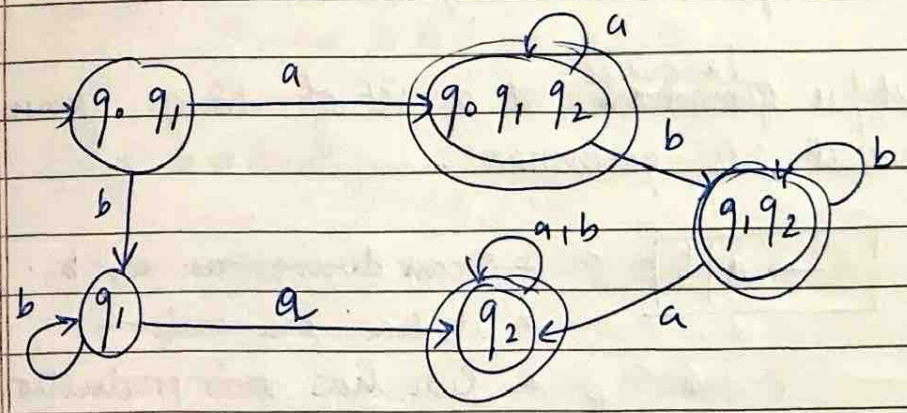
$= \epsilon^*(q_1 \cup q_2)$

$= \{q_1,q_2\}$

Step 6: $\delta_D(q_2,a) = \epsilon^*(\delta_N(q_2a)) = q_2$

$\delta_D(q_2,b) = \epsilon^*(\delta_N(q_2,b)) = q_2$

	a	b
→ $q_0\,q_1$	$q_0q_1q_2$	q_1
q_1	q_2	q_1
$q_0q_1q_2$	$q_0q_1q_2$	q_1q_2
q_1q_2	q_2	q_1q_2
q_2	q_2	q_2



3/6/24

Context Free Grammar

→ set of production rules

It is defined by 4 tuples as $G = \{V, T, P, S\}$

V = Set of variables * uppercase

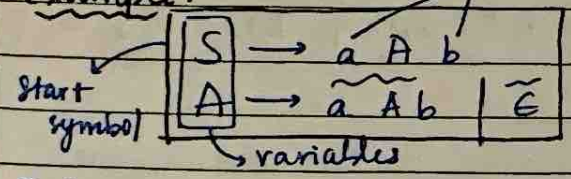
T = Set of terminal symbols * lowercase.

P = Production rules

S = Start symbol

Example:

terminal / I/P symbols



Rule 1: 1 production

Rule 2: 2 productions

LHS → variables

$G = (\{S, A\}, \{a, b\}, \{S \to aAb, A \to aAb \mid \in\}, S)$

(V, T)

Grammar generates strings.

* Context free language is set of strings generated by context free grammar.

① $S \to a \mid b$

* S can derive a or b.
* it has one rule.
* Rule has two productions

$G = (\{S\}, \{a, b\}, \{S \to a \mid b\}, S)$ * | means or.

$L = \{a, b\}$

② $A \to aA \mid \in$ 2 productions

* A is recursive symbol

$L = \{\in, a, aa, aaa, \ldots\} = a^*$

$G = (\{S\}, \{a\}, \{\}, S)$

③ $L = \{a^n b^n \mid n \geq 0\}$

$S \longrightarrow aSb \mid \epsilon$

$S \longrightarrow aSb$

$\longrightarrow aaSbb$

$\longrightarrow aaaSbbb$

$\longrightarrow aaabbb$

The condition for context free grammar: ↗ union

If $A \longrightarrow \alpha$ where $\alpha = \{V \cup T\}$

and $A \in V$

eg: $bS \longrightarrow cSd \mid \epsilon$ is a grammar but not context free. since $bS \notin V$.

Left Most Derivation (LMD)

Here string is obtained by applying production to the left most variable in each step

Example:

① $S \longrightarrow aAS \mid aSS \mid \epsilon$

$A \longrightarrow SbA \mid ba$

String: aabaa

~~$S \longrightarrow aSS$~~

~~$\longrightarrow aaSSS$~~

$\longrightarrow aaa$

$S \longrightarrow aSS$

$\longrightarrow aaASS$

$\longrightarrow aabaSS$

$\longrightarrow aabaaSSS$

$\longrightarrow aabaaSS$

$\longrightarrow aabaaS$

$\longrightarrow aabaa$

Right most Derivation (RMD)

Here string is obtained by applying production to the right most variable in each step

① $S \rightarrow aAS \mid aSS \mid \epsilon$

$A \rightarrow SbA \mid ba$

String: aabaa

RMD:

$S \rightarrow aS\boxed{S}$

$\rightarrow aSaA\boxed{S}$

$\rightarrow aSaAaS\boxed{S}$

$\rightarrow aSaAa\boxed{S}$

$\rightarrow aSa\boxed{A}a$

$\rightarrow a\boxed{S}abaa$

$\rightarrow aabaa$

Ambiguous Grammar

If there exists two LMD or two RMD, then the grammar is said to be ambiguous

Example:

$S \rightarrow S*S$

$\mid S+S$

$\mid a$

$\mid b$

String: $a+a*b$

LMD

$S \rightarrow S+S$

$\rightarrow a+S$

$\rightarrow a+S*S$

$\rightarrow a+a*S$

$\rightarrow a+a*b$

LMD

$S \rightarrow S*S$

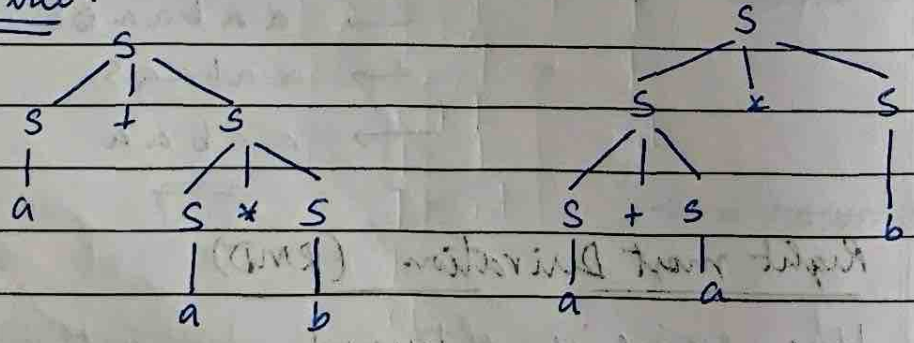
$\rightarrow S+S*S$

$\rightarrow a+S*S$

$\rightarrow a+a*S$

$\rightarrow a+a*b$

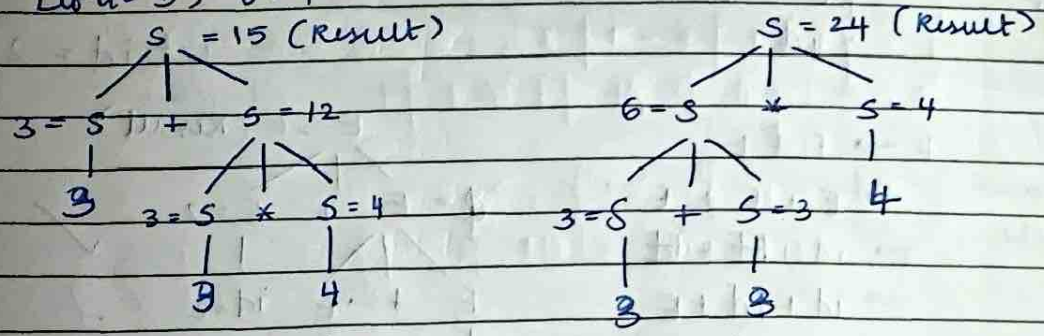
4/6/24 * Parse tree:



(*) It is an ordered rooted tree that graphically represents the semantic information of strings

derived from a context free grammar.

Let a = 3, b = 4



* In second stage of compiler design, it generates a parse tree.
* So the grammar is called ambiguous, because the result ~~can differ based~~ of the expression depends on the parse tree generated. It depends on the parse tree.

Recursive grammar

eg: (1) $A \longrightarrow A\alpha \mid \beta$

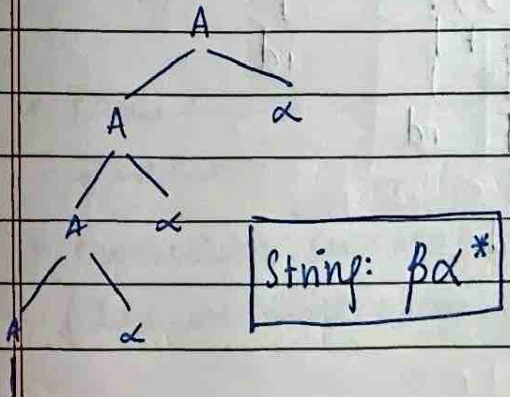
(2) $A \longrightarrow \alpha A \mid \beta$

Left recursion

LR

eg: $A \longrightarrow A\alpha \mid \beta$

* In any production, the first symbol should be the variable



String: $\beta\alpha^*$

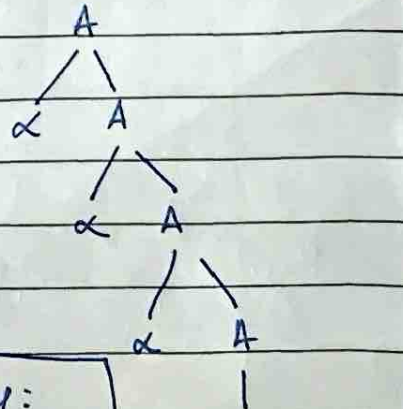
≠

Right recursion

RR

eg: $A \longrightarrow \alpha A \mid \beta$

* In any production, the last symbol should be the variable (recursion)



String: $\alpha^*\beta$

eg:

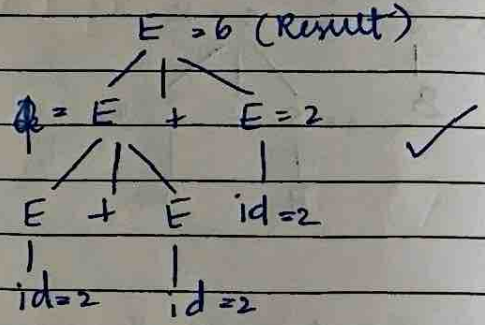
$E \rightarrow E + E$
$\quad | \; E * E$
$\quad | \; id$

String: id + id + id.

Let id = 2

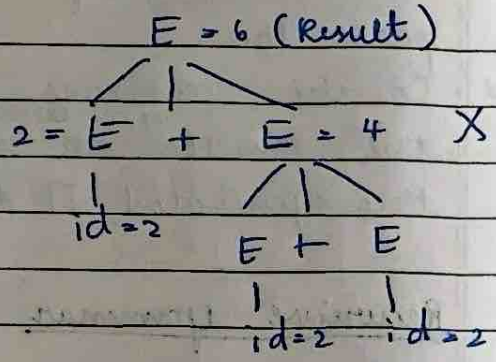
tree grows in the left

$E \rightarrow E + E$
$\rightarrow E + E + E$
$\rightarrow id + E + E$
$\rightarrow id + id + E$
$\rightarrow id + id + id$



tree grows in the right

$E \rightarrow E + E$
$\rightarrow E + E + E$
$\rightarrow E + E + id$
$\rightarrow E + id + id$
$\rightarrow id + id + id$

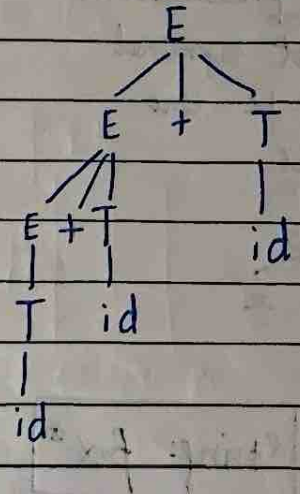


The first one is the correct way of evaluation for addition since it is left associativity.

Left associativity — LR
Right associativity → RR.

Modification:

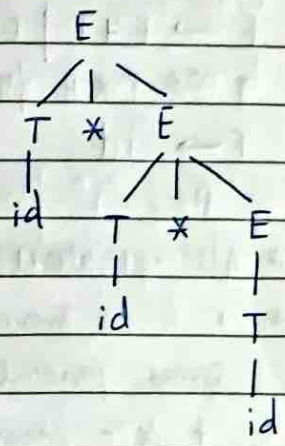
$E \rightarrow E + T \; | \; T$
$T \rightarrow id$



Let us assume that * is right associative.

RR: $E \rightarrow T * E \mid T$

$T \rightarrow id$



eg: $E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

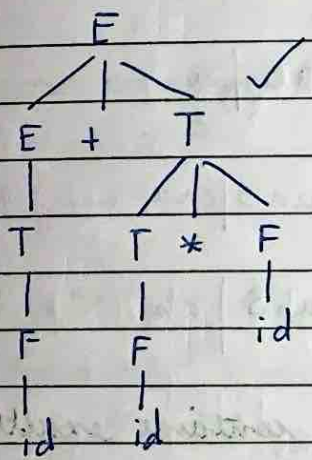
$F \rightarrow id$

$E \rightarrow E+T \mid E*T \mid T$

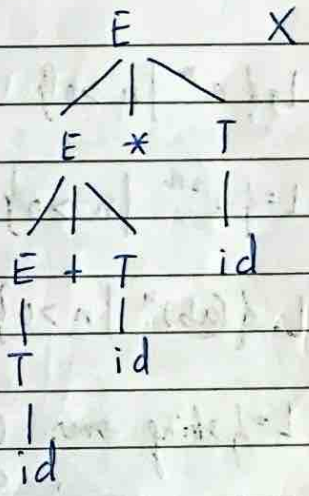
$T \rightarrow id$

may look same but left one sets the priority to * over +

eg: id + id * id



Multiplication first
Addition next



Addition first
Multiplication next

* Productions in same rule have same precedence.
* Production in next rule have more priority (like in first example)

Grammar for arithmetic expression

$E \rightarrow E+T \mid E-T \mid T$
$T \rightarrow T*F \mid T/F \mid T\%F \mid F$
$F \rightarrow id.$

* All operators are left associativity
* + & - have same priority and *, /, % have same priority
* + & - have lower precedence than *, /, %

Write a CFG for each of the following languages.

i) $L = \{a^n b^n \mid n>0\}$

$S \rightarrow aSb \mid ab$

ii) $L = \{a^n \mid n>0\}$ $S \rightarrow aS \mid a$

iii) $L = \{a^{2n} \mid n>0\}$ $S \rightarrow aaS \mid aa$

iv) $L = \{(ab)^n \mid n>0\}$ $S \rightarrow abS \mid ab$

v) L = { string over (a,b) that contains exactly two b's }

~~$S \rightarrow Sbb$~~
~~$S \rightarrow AbbA \mid bb$~~.
$S \rightarrow AbAbA$
$A \rightarrow aA \mid \epsilon$

vi) L = { string over (a,b) that contains at least two b's }

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

vii) $L = \{a^+ b^*\}$

~~$S \to aAbB$~~ ~~$S \to aA$~~

~~$A \to a$~~

$S \to aS \mid aA$

$A \to bA \mid \epsilon$

viii) $L = \{$odd length palindrome over $\{a, b\}\}$

~~$S \to aSb$~~

$L = \{a, b, bab, aba, bbb, aaa, aabaa, ababa, \ldots\}$

$S \to aSa \mid bSb \mid a \mid b$

[OR]

$S \to aSa$
$\mid bSb$
$\mid a$
$\mid b$

ix) $L = \{a^n b^m a^n \mid m, n > 0\}$

~~$S \to aSbSaS \mid aA$~~

$S \to aSa \mid bA$

$A \to bA \mid b$

$S \to aSa \mid aAa$

$A \to bA \mid b$

x) $L = \{a^n b^n c^m \mid m, n > 0\}$

~~$S \to aS$~~ ~~$S \to A$~~

~~$S \to AcS$~~ ⊗

$S \to AB$
$A \to aAb \mid ab$
$B \to cB \mid c$

$S \to aSbA \mid aSb \mid ab$
$A \to cA \mid c$

xi) $L = \{a^n b^n c^m d^m \mid m, n > 0\}$

$S \to AB$

$A \to aAb \mid ab$

$B \to cBd \mid cd$

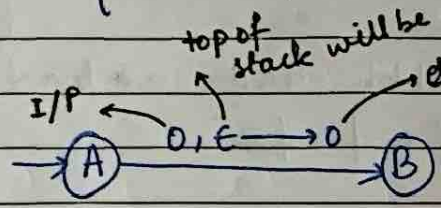
Pushdown Automata (PDA)

↳ to implement context free grammar

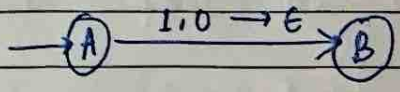
↳ PDA = FSM + stack

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

$L = \{01, 0011, 000111, \dots\}$

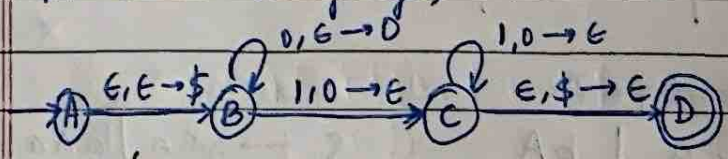


* when I/P is 0, w/o checking top, push 0 into stack



* When I/P = 1, top = 0, then nothing is pushed

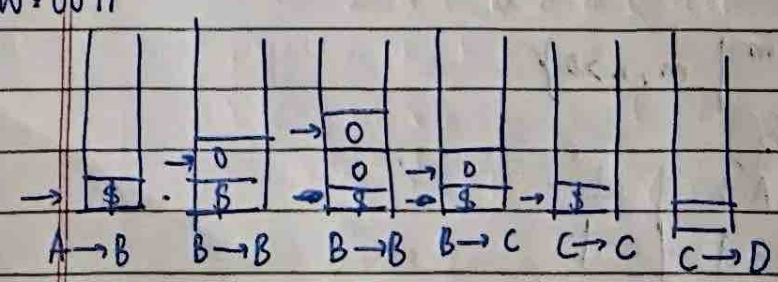
For above language,



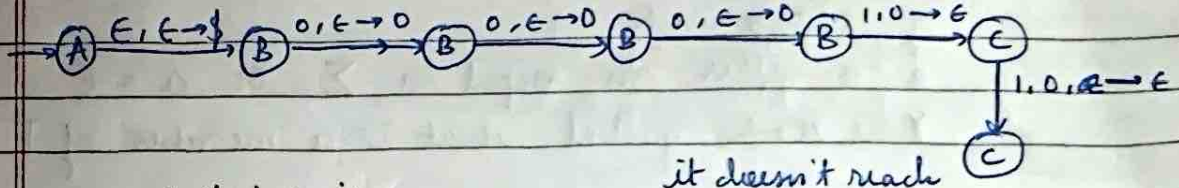
'$' is pushed to know if stack is empty / reached bottom of stack

* When I/P is 0, 0 is pushed
  B → B
* when I/P is 1, top = 0, & nothing is pushed, 0 is popped
  B → C
* when repeat C → C
* When I/P is nothing, $ is popped if it is top.

W = 0011



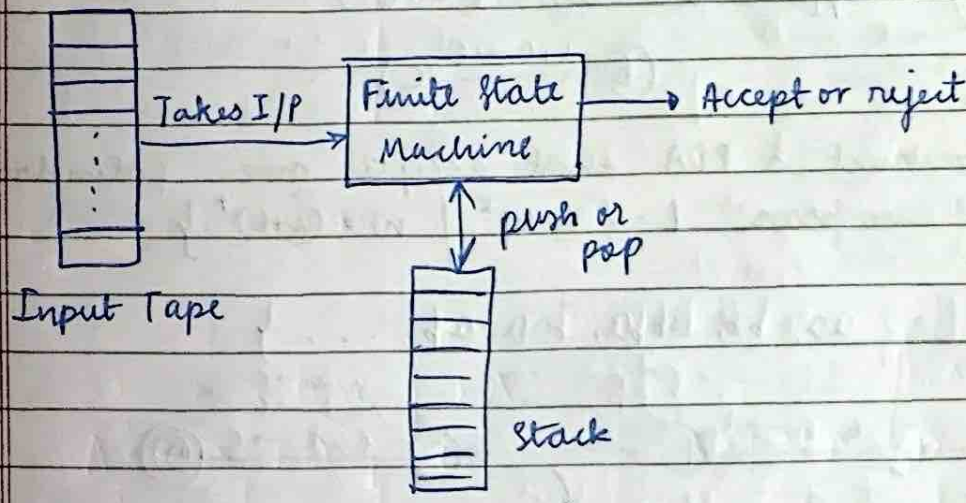
W = 000 11



∴ string is invalid

it doesn't reach final state because the top of stack is 0 when I/P = ϵ.

* PDA has three components:
  1. I/P Tape
  2. Finite control unit
  3. Stack with infinite size.



* PDA is formally defined by 7 tuples:

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

Q = finite set of states
Σ = finite set of input symbols
Γ = finite stack alphabet $(V \cup T)$
δ = transition function
q_0 = start state
$\$$ = start stack symbol
F = set of final states

Transition function $\delta(q, a, X)$

q is state in Q

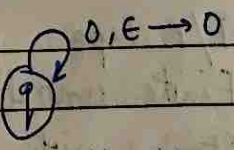
a is either an input in Σ or $a = \epsilon$

X is stack symbol that is a member of Γ

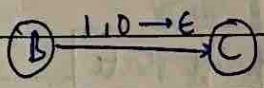
eg: 1) $\delta(q, 0, 0) = (q, \epsilon)$

- 0 → I/P
- 0 → top of stack that is popped
- ϵ → pushed into stack.

2) $\delta(q, 0, \epsilon) = (q, 0)$

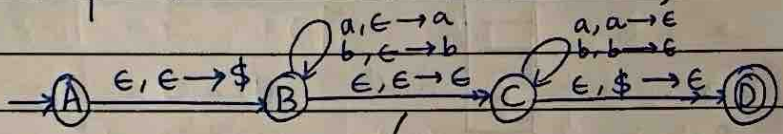


3) ~~$\delta(q_1$~~, $\delta(B, 1, 0) = (C, \epsilon)$



(1) Construct a PDA that accepts given palindromes of the form $L = \{ w w^R \mid w = (a+b)^+ \}$

$L = \{ aa, bb, abba, baaaab, \ldots \}$



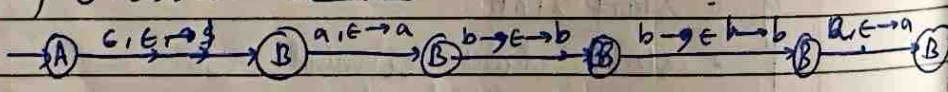
divide string into 2 parts.

It assumes ϵ in all positions and tries all combinations.

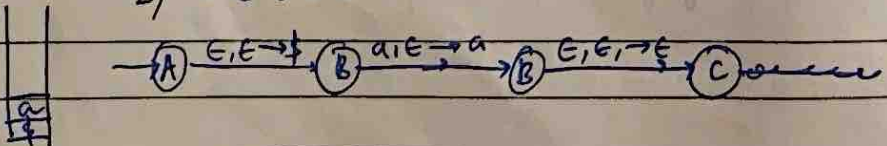
~~ϵ a ϵ ϵ b ϵ ϵ ϵ a ϵ~~.

eg: a b b a.

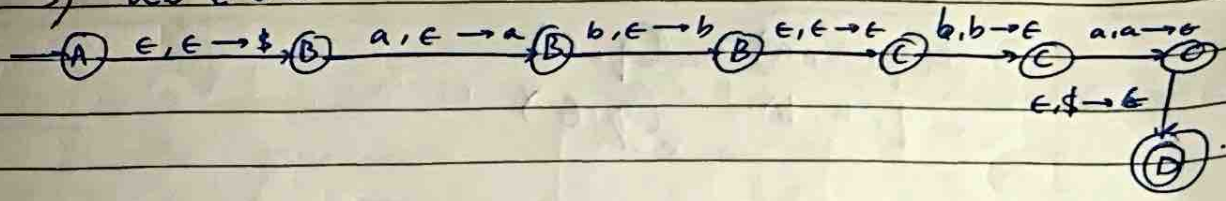
1) ϵ a b b a. ~~will~~



2) a ϵ b b a.



3) ab ∈ ba



Converting CFG to PDA

1) For non terminal, $A \rightarrow \alpha$

$\delta(q, \epsilon, A) = (q, \alpha)$ — push the production

(top of stack to pop)

2) For terminal a,

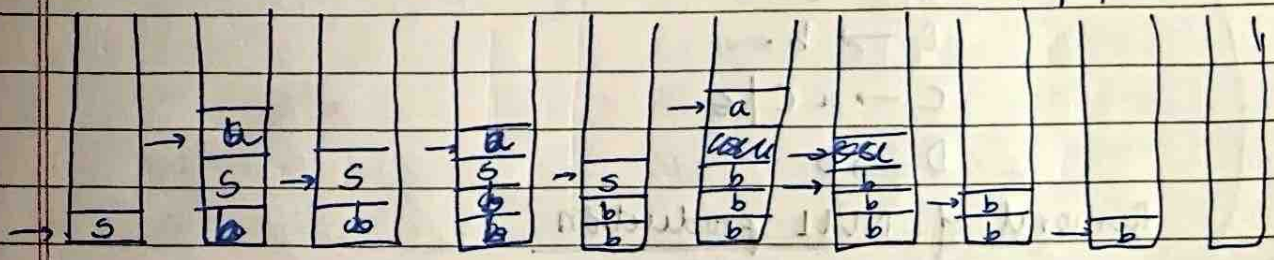
$\delta(q, a, a) = (q, \epsilon)$

eg:

$S \rightarrow aSb \mid ab$

String : aaabbb

* When top is V, push the production into stack
* If top is T, compare with I/P, and pop it



1) Whether top of stack is variable ?
   Pop variable and push a production
2) Whether top of stack is terminal ?
   Compare with I/P, if same, pop out of stack
3) If stack is empty, string is accepted by PDA.

PDA:

Ans:

$P = (q, \{a, b\}, \delta, \{a, b, S\}, S, q_0, \phi)$

Transition function for Non terminal:

$S \rightarrow aSb \quad \delta(q, \epsilon, S) = (q, aSb)$

$S \rightarrow ab \quad \delta(q, \epsilon, S) = (q, ab)$

Transition function for

Terminal

$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

In PDA, input is accepted when we reach final state or when stack is empty.

Removal of unit production

*. If the production rule has only one variable as production, then it is unit production

$A \longrightarrow BC$

$B \longrightarrow D$ * this is unit production.

$C \longrightarrow aC \mid \epsilon$ ↘ $B \longrightarrow D \longrightarrow b$

$D \longrightarrow b$.

After removal:

$A \longrightarrow BC$

$B \longrightarrow b$.

$C \longrightarrow aC \mid \epsilon$

$D \longrightarrow b$

Removal of NULL production

$C \longrightarrow \epsilon$

RHS.

$A \longrightarrow BC$ $\quad C \longrightarrow aC$

$A \longrightarrow B$. (Replace C with ϵ) $\quad C \longrightarrow a$

$A \longrightarrow BC \mid B$

$B \longrightarrow D$ $\Rightarrow$ $B \longrightarrow b$.

$C \longrightarrow aC \mid \epsilon$ $\quad C \longrightarrow aC \mid a$

$D \longrightarrow b$ $\quad D \longrightarrow b$

Removal of UNIT production

Step 1: To remove $A \to B$, add production $A \to x$ to the grammar rule whenever $B \to x$ occurs in the grammar
[$x \in$ terminal, x can be NULL]

Step 2: Delete $A \to B$ from grammar

Step 3: Repeat Step 1-2 until all unit productions are removed.

Remove unit productions problems:

i) $S \to XY$
$X \to a$
$Y \to Z \mid b$
$Z \to M$
$M \to N$
$N \to a$

No. of productions = 7.
No. of unit productions = 3.

I. $S \to XY$
$X \to a$
$Y \to M \mid b$
$M \to N$
$N \to a$

II. $S \to XY$
$X \to a$
$Y \to N \mid b$
$N \to a$

III. $S \to XY$
$X \to a$
$Y \to a \mid b$

(or)

$S \to XY$
$X \to a$
$Y \to Z \mid b$
$Z \to M$
$M \to a$
$N \to a$

To remove $M \to N$, since $N \to a$, $M \to a$.

To remove $Z \to M$, since $M \to a$, $Z \to a$.

$S \to XY$
$X \to a$
$Y \to Z \mid b$
$Z \to a$
$M \to a$
$N \to a$

To remove $Y \to Z$, since $Z \to a$, $Y \to a$.

$S \to XY$
$X \to a$
$Y \to a \mid b$
$Z \to a$
$M \to a$
$N \to a$

Table

PTO.

Here the symbol Z, M, N are not reachable from start symbol, 'S', we can remove productions that are derived from Z, M, N.

$S \rightarrow XY$
$X \rightarrow a$
$Y \rightarrow a \mid b$

Removal of NULL production

Step 1: To remove $A \rightarrow \epsilon$, look for all productions that contain A.

Step 2: Replace each occurrences of A in each of these productions with ϵ.

Step 3: Add the resultant productions to the grammar.

eg: $S \rightarrow ABAC$
$A \rightarrow aA \mid \epsilon$
$B \rightarrow bB \mid \epsilon$
$C \rightarrow c$

No. of productions : 6
No. of NULL productions = 2

For $A \rightarrow \epsilon$,
$S \rightarrow ABAC$ becomes $S \rightarrow BAC$ and $S \rightarrow ABC$
$A \rightarrow aA$ becomes $A \rightarrow a$

For $B \rightarrow \epsilon$,
$S \rightarrow ABAC$ becomes $S \rightarrow AAC$
$B \rightarrow bB$ becomes $B \rightarrow b$

For $A \rightarrow \epsilon$,
$S \rightarrow ABAC$ becomes $S \rightarrow ABC \mid BAC \mid BC$
$A \rightarrow aA$ becomes $A \rightarrow a$

For $B \rightarrow \epsilon$,
$S \rightarrow ABAC \mid BAC \mid BC$ becomes $S \rightarrow AAC \mid AC \mid C$
$B \rightarrow bB$ becomes $B \rightarrow b$

∴ Resultant grammar,

$S \longrightarrow ABAC \mid BAC \mid BC \mid AAC \mid AC \mid C$
$A \longrightarrow aA \mid a$
$B \longrightarrow bB \mid b$
$C \longrightarrow c$

Simplification or Minimization of CFGs:

1. Removal of ϵ production
2. Removal of unit production
3. Removal of useless variables

} To make it efficient and compiler friendly

Removal of useless variable

The variable which are not involved in derivation of any string is useless symbol.

Step1: Select the variables that cannot be reached from start symbol of grammar and remove them along with their all productions.

eg: $S \longrightarrow AB \mid C$, $A \longrightarrow b$, $B \longrightarrow d$, $C \longrightarrow c$ $\Rightarrow$ $S \longrightarrow AB \mid C$, $A \longrightarrow b$, $B \longrightarrow d$

Here we remove C because not reachable from Start symbol S.

Step2: Select variable that are reachable from start symbol but it doesn't derive any terminal. Remove them along with their productions

Eg: $S \longrightarrow AB \mid AC$, $A \longrightarrow a$, $B \longrightarrow bB$, $C \longrightarrow c$ $\Rightarrow$ $S \longrightarrow AC$, $A \longrightarrow a$, $C \longrightarrow c$

Here B is reachable but doesn't derive any terminal

Eg: ① $S \to aAB \mid bA \mid \cancel{\epsilon}$ $\Rightarrow$ $S \to aAB \mid bA$

$A \to aB \mid b$ $\Rightarrow$ $A \to aB \mid b$

$B \to \cancel{\epsilon} \mid d$ $B \to d$

② $S \to aA \mid \cancel{abB}$ $S \to aA$

$A \to aaA \mid \epsilon$ $\Rightarrow$ $A \to aaA \mid \epsilon$

$B \to \cancel{bB} \mid \cancel{bb\epsilon}$ // remove null production

$\cancel{\epsilon \to B}$ $\Rightarrow$ $S \to aA \mid a$

$A \to aaA \mid aa$

Simplify the CFG:

① $S \to A \mid 0C1$

$A \to B \mid 01110$

$C \to \epsilon \mid cD$

* Remove useless variable B and D

$S \to A \mid 0C1$

$A \to 01110$

$C \to \cancel{\epsilon} \epsilon$

* Remove null production

$\cancel{S \to A \mid 0C1 \mid 01}$

$S \to A \mid 01$

$A \to 01110$

* Remove unit production

$S \to 01110 \mid 01$

② $S \to aS \mid aA$

$A \to bA \mid \epsilon$

* Remove null production

$S \to aS \mid aA \mid a$

$A \to bA \mid b$

Write a context free grammar for each following languages:

(1) $L = \{a^n b^m a^n \mid m, n > 0\}$

~~$S \rightarrow aSa \mid B$~~
~~$B \rightarrow bb \mid b$~~

$S \rightarrow aSa \mid B$
$B \rightarrow bB \mid b$

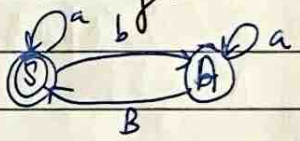
(2) $L = \{a^n b^m c^m d^{2n} \mid m, n \geq 0\}$

$S \rightarrow aSdd \mid A$
$A \rightarrow bAc \mid \epsilon$

~~$S \rightarrow aSdd \mid A \mid bcc$~~
~~$A \rightarrow bAc \mid bc$~~

(3) String over $\{a, b\}$ with even no. of b's

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



(4) String with even no. of a's and b's

$S \rightarrow aA \mid bB \mid \epsilon$
$A \rightarrow aS \mid bC$
$B \rightarrow bS \mid aC$
$C \rightarrow bA \mid aB$

$S \rightarrow aA \mid bB \mid \epsilon$
$A \rightarrow aS$
$B \rightarrow bS$

↳ doesn't accept abab.

(5) Strings with even a's and odd b's

(6) Strings with odd no. of a's and b's

(7) Strings over {a, b, c} that doesn't contain abc as substrings

Chomsky Normal Form (CNF)

In CNF, we have restriction in length of RHS ~~which~~ is ie elements in right side should either be two variables or a terminal.

A CFG is a CNF if the production are in following form:

$$A \longrightarrow a$$
$$A \longrightarrow BC$$

eg:

$S \longrightarrow AB$ ✓
$A \longrightarrow a$ ✓
$B \longrightarrow b$ ✓
[CNF]

$S \longrightarrow A$ ✗
$A \longrightarrow BC$ ✓
$B \longrightarrow b$ ✓
$C \longrightarrow c$ ✓
[not CNF]

Convert CFG $\longrightarrow$ CNF

Step 1: If start symbol S occurs on right side, create a new start symbol S' and new production, $S' \longrightarrow S$.

Step 2: Remove null productions

Step 3: Remove unit production

Step 4: Replace each production i.e $A \longrightarrow B_1 \dots B_n$ where $n > 2$ with

$$A \longrightarrow B_1 C$$
$$C \longrightarrow B_2 \dots B_n$$

Step 5: Repeat this step for all productions having 2 or more variables in RHS.

Step 6: If the right side of any production is of the form $A \longrightarrow aB$ then the production is replaced by

$$A \longrightarrow XB$$
$$X \longrightarrow a$$

Step 7: Remove the useless symbols.

classmate
Date
Page

Q:① $S \rightarrow ASA \mid aB$
$A \rightarrow B \mid S$
$B \rightarrow b \mid \epsilon$

Step 1: change start symbol

$S' \rightarrow S$
$S \rightarrow ASA \mid aB$
$A \rightarrow B \mid S$
$B \rightarrow b \mid \epsilon$

Step 2: remove NULL production

$S' \rightarrow S$
$S \rightarrow ASA \mid aB \mid a$
$A \rightarrow B \mid S \mid \epsilon$
$B \rightarrow b$

$\Rightarrow$

$S' \rightarrow S$
$S \rightarrow ASA \mid aB \mid a \mid S \mid SA \mid AS$
$A \rightarrow B \mid S$
$B \rightarrow b$

Step 3: remove unit production

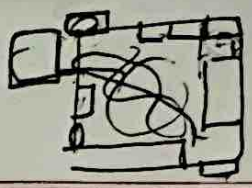
~~$S' \rightarrow S$~~
~~$S \rightarrow ASA \mid aB \mid a$~~
~~$A \rightarrow b$~~

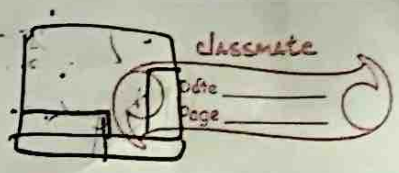
1) Remove $S \rightarrow S$
$S' \rightarrow S$
$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$A \rightarrow B \mid S$
$B \rightarrow b$

2) Remove $S' \rightarrow S$
$S' \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$A \rightarrow B \mid S$
$B \rightarrow b$

3) Remove $A \rightarrow B$
$S' \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$A \rightarrow b \mid S$
$B \rightarrow b$

4) Remove $A \rightarrow S$
$S' \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$
$B \rightarrow b$





Step 4: Remove $S' \to ASA$, $S \to ASA$, $A \to ASA$

~~1) Remove $S' \to ASA$~~

$S' \to AC \mid aB \mid a \mid SA \mid AS$

$S \to AC \mid aB \mid a \mid SA \mid AS$

$A \to b \mid AC \mid aB \mid a \mid SA \mid AS$

$B \to b$

$C \to SA$

Step 5 Remove $S' \to aB$, $S \to aB$, $A \to aB$.

$S' \to AC \mid XB \mid a \mid SA \mid AS$

$S \to AC \mid XB \mid a \mid SA \mid AS$

$A \to b \mid AC \mid XB \mid a \mid SA \mid AS$

$B \to b$

$C \to SA$

$X \to a$.

This is a CNF.

eg: ② $S \to as \mid aA$

$A \to bA \mid \epsilon$

Step 1:

$S' \to S$

$S \to aS \mid aA$

$A \to bA \mid \epsilon$

Step 2:

$S' \to S$

$S \to aS \mid aA \mid a$

$A \to bA \mid b$

Step 3:

$S' \to aS \mid aA \mid a$.

$S \to aS \mid aA \mid a$.

$A \to bA \mid b$.

Step 4: none.

Step 5:

$S' \to aXS \mid XA \mid a$.

$S \to XS \mid XA \mid a$

$A \to YA \mid b$.

$X \to a$

$Y \to b$.

classmate
Date
Page

(3) $S \rightarrow a \mid aA \mid B \mid C$
$A \rightarrow aB \mid \epsilon$
$B \rightarrow aA \mid \epsilon$
$C \rightarrow cCD$
$D \rightarrow ddd$

Step 1: None

Step 2: Remove $A \rightarrow \epsilon$

$S \rightarrow a \mid aA \mid B \mid C$
$A \rightarrow aB$
$B \rightarrow aA \mid a$
$C \rightarrow cCD$
$D \rightarrow ddd$

Remove $B \rightarrow \epsilon$

$S \rightarrow a \mid aA \mid B \mid C \mid \epsilon$
$A \rightarrow aB \mid a$
$B \rightarrow aA \mid a$
$C \rightarrow cCD$
$D \rightarrow ddd$

Remove $S \rightarrow \epsilon$

$S \rightarrow a \mid aA \mid B \mid C$
$A \rightarrow aB \mid a$
$B \rightarrow aA \mid a$
$C \rightarrow cCD$
$D \rightarrow ddd$

Step 3: Remove $S \rightarrow B$, $S \rightarrow C$

$S \rightarrow a \mid aA \mid cCD$
$A \rightarrow aB \mid a$
$B \rightarrow aA \mid a$
$C \rightarrow cCD$
$D \rightarrow ddd$

Step 4: Remove useless variables

$S \rightarrow a \mid aA \mid cCD$
$A \rightarrow aB \mid a$
$B \rightarrow aA \mid a$
~~$C \rightarrow cCD$~~
~~$D \rightarrow ddd$~~

$S \rightarrow a \mid aA$

$A \rightarrow aB \mid a$

$B \rightarrow aA \mid a$

Step 5:

$S \rightarrow a \mid XA$

$A \rightarrow XB \mid a$

$B \rightarrow XA \mid a$

$X \rightarrow a$

2. $S \rightarrow ABC \mid BaB$

$A \rightarrow aA \mid BaC \mid aaa$

$B \rightarrow bBb \mid a \mid D$

$C \rightarrow CA \mid AC$

$D \rightarrow \epsilon$

Step 1: None

Step 2: Remove $D \rightarrow \epsilon$

$S \rightarrow ABC \mid BaB$

$A \rightarrow aA \mid BaC \mid aaa$

$B \rightarrow bBb \mid a \mid \epsilon$

$C \rightarrow CA \mid AC$

Remove $B \rightarrow \epsilon$

$S \rightarrow ABC \mid AC \mid aB \mid Ba \mid BaB \mid a$

$A \rightarrow aA \mid BaC \mid aaa \mid aC$

$B \rightarrow bBb \mid a \mid bb$

$C \rightarrow CA \mid AC$

Step 3: None (no unit production)

Step 4: $S \rightarrow ABC \mid AC \mid aB \mid Ba \mid BaB \mid a$

$A \rightarrow aA \mid BaC \mid aaa \mid aC$

$B \rightarrow bBb \mid a \mid bb$

$C \rightarrow CA \mid AC$

$S \rightarrow aB \mid Ba \mid BaB \mid a$

$A \rightarrow aA \mid aaa$

$b \rightarrow bbb \mid a \mid bb$

Step 5: $S \to XB \mid BX \mid BXB \mid a$
$A \to XA \mid XXX$
$B \to YBY \mid a \mid YY$
$X \to a$
$Y \to b$

(a)

$S \to XB \mid BX \mid a \mid BN$
$A \to XA \mid XZ$
$B \to YM \mid YY \mid a$
$X \to a$
$Y \to b$
$Z \to XX$
$M \to BY$
$N \to XB$

3. $S \to OAO \mid 1B1 \mid BB$
$A \to C$
$B \to S \mid A$
$C \to S \mid \epsilon$

Step 1: $S' \to S$
$S \to OAO \mid 1B1 \mid BB$
$A \to C$
$B \to S \mid A$
$C \to S \mid \epsilon$

Step 2: Remove $C \to \epsilon$
$S' \to S$
$S \to OAO \mid 1B1 \mid BB$
$A \to \epsilon \mid C$
$B \to S \mid A$
$C \to S$

Remove $A \to \epsilon$
$S' \to S$
$S \to OAO \mid 1B1 \mid BB \mid 00$
$B \to S \mid A \mid \epsilon$
$C \to S$
$A \to C$

Remove $B \to \epsilon$

$S' \to S$

$S \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11 \mid B$

$B \to S \mid A$

$C \to S$

$A \to \epsilon$

Step 3: ~~Remove $S' \to \epsilon$~~ Remove unit production

$S' \to S$

$S \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11 \mid B$

$A \to S$

$B \to S$

$C \to S$

$A \to A \mid a \Rightarrow A \to a$

$S' \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$

[$S \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$]

$A \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$

$B \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$

[$C \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$] useless

Step 4

$S' \to 0A0 \mid 1B1 \mid BB \mid \underset{X}{00} \mid \underset{Y}{11}$

$A \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$

$B \to 0A0 \mid 1B1 \mid BB \mid 00 \mid 11$

Step 5:

$S' \to \underset{M}{XAX} \mid \underset{N}{YBY} \mid BB \mid XX \mid YY$

$A \to 0XAX \mid YBY \mid BB \mid XX \mid YY$

$B \to 0XAX \mid YBY \mid BB \mid XX \mid YY$

$X \to 0$

$Y \to 1$

$S' \to XM \mid YN \mid BB \mid XX \mid YY$

$A \to XM \mid YN \mid BB \mid XX \mid YY$

$B \to XM \mid YN \mid BB \mid XX \mid YY$

$X \to 0$

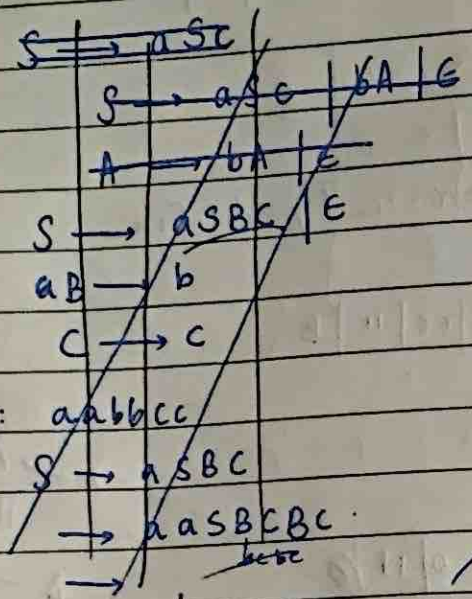
$y \to 1$

$M \to AX$

$N \to BY$

Write a CFG for the language:

① $L = \{a^n b^n c^n \mid n \geq 0\}$ → you can't write a CFG.



$S \rightarrow ABC$
$A \rightarrow aA \mid \epsilon$
$B \rightarrow bB \mid \epsilon$
$C \rightarrow cC \mid \epsilon$

→ you can't write a CFG.

② $L = \{WW \mid W \in (0+1)^*\}$

$S \rightarrow XX$

$X \rightarrow 0X \mid 1X \mid \epsilon$

? not correct

The above two languages are not context free language because there is no context free grammar.