FA → LLG

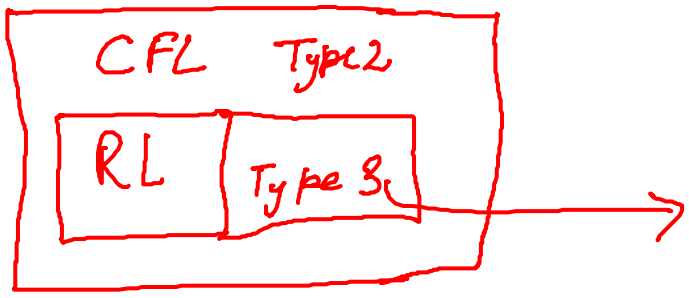
Type 2 $\quad \alpha \rightarrow \beta$

Only one variable

$A \rightarrow \beta$

$\beta \in (V \cup \Sigma)^*$

Context Free Grammar



Linear grammer

Right

$A \rightarrow aB$

Left

$A \rightarrow Ba$

Ex Start with a 'a'

CFG Introduction

- Consider language $\{ 0^n 1^n \mid n \geq 0 \}$, which is nonregular.
- Start variable S with "substitution rules":

$$\begin{cases} S \to 0S1 \\ S \to \varepsilon \end{cases}$$

$\alpha \to \beta$

$\alpha \to$ only one variable

$\beta \in (V \cup \Sigma)^*$

- Rules can **yield** string $0^k 1^k$ by
  - applying rule "$S \to 0S1$" k times,
  - followed by rule "$S \to \varepsilon$" once.
- **Derivation** of string $0^3 1^3$

$$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000S111 \Rightarrow 000\varepsilon111 = 000111$$

CFG Introduction

Definition of CFG

Definition: Context-free grammar (CFG) $G = (V, \Sigma, R, S)$ where

1. V is finite set of **variables** (AKA **nonterminals**)
2. Σ is finite set of **terminals** (with $V \cap \Sigma = \emptyset$)
3. R is finite set of substitution **rules** (AKA **productions**), each of the form

$$L \to X,$$

where

- $L \in V$
- $X \in (V \cup \Sigma)^*$

4. S is **start variable**, where $S \in V$

Language of CFG

Definition: u **derives** v, written $u \overset{*}{\Rightarrow} v$, if

- $u = v$, or
- $\exists\, u_1, u_2, \ldots, u_k$ for some $k \geq 0$ such that

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$$

Remark: "$\overset{*}{\Rightarrow}$" denotes a sequence of ≥ 0 single-step derivations.

Example: With the rules "$A \to B1 \mid D0C$",

$$0AA \overset{*}{\Rightarrow} 0D0CB1$$

Definition: The **language** of CFG $G = (V, \Sigma, R, S)$ is

$$L(G) = \{\, w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w \,\}.$$

Such a language is called **context-free**, and satisfies $L(G) \subseteq \Sigma^*$.

CFG Introduction

Example of CFG

- CFG $G = (V, \Sigma, R, S)$ with
  1. $V = \{S\}$
  2. $\Sigma = \{0, 1\}$
  3. Rules R:

$$S \to 0S \mid \varepsilon$$

- Then $L(G) = \{ 0^n \mid n \geq 0 \}$.
- For example, S derives 0^3

$$S \Rightarrow 0S \Rightarrow 00S \Rightarrow 000S \Rightarrow 000\varepsilon = 000$$

- Note that $\to$ and $\Rightarrow$ are different.
  - $\to$ used in defining rules
  - $\Rightarrow$ used in derivation

(i) $S \Rightarrow \varepsilon$

(ii) $S \Rightarrow 0S \Rightarrow 00S \Rightarrow 00$

(iii) $S \Rightarrow 0S \Rightarrow 00S \Rightarrow 000$

$\{\varepsilon, 0, 00, \ldots\} \Rightarrow 0^*$

Derivation Tree / Parse Tree

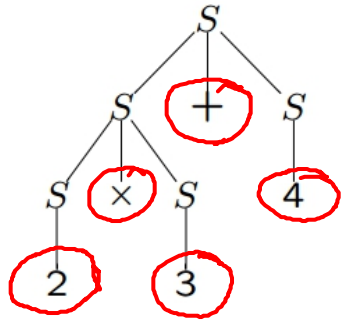
- CFG

$$S \to S + S \mid S - S \mid S \times S \mid S/S \mid (S) \mid -S \mid 0 \mid 1 \mid \cdots \mid 9$$

- Can generate string $2 \times 3 + 4$ using derivation

$$S \Rightarrow S + S \Rightarrow S \times S + S \Rightarrow 2 \times S + S \Rightarrow 2 \times 3 + S \Rightarrow 2 \times 3 + 4$$

  - **Leftmost derivation**: leftmost variable replaced in each step.
- Corresponding **derivation** (or **parse**) **tree**



- **Depth-first** traversal of tree
  - Starting at **root**, walk around tree with left hand always touching tree.
  - string = sequence of **leaves** visited.

Derivation → LMD / RMD

① $L_1 = \{a^n : n \geq 0\}$ ✓

a^*

$S \to aS \mid \varepsilon$

② $L_2 = \{a^n : n \geq 1\}$ ✓

a^+

$S \to aS \mid a$

③ $L_3 = (a+b)^*$ ✓

$L_3 = \{w : w \in \Sigma^*, \Sigma = \{a, b\}\}$

$A \to aA \mid bA \mid \varepsilon$

④ ~~$L_4 = \{a^n b^n : n \geq 1\}$~~

$L_4 = \{a^n b^n : n \geq 1\}$

$S \to AB$

$A \to aA \mid a$

$B \to bB \mid b$

$ww^R \Rightarrow$

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

b b

abbaabba

~~bcb~~ bb

$L_5 \Rightarrow L_5 = \{wcw^R : w \in \Sigma^*\}$ $\Sigma = \{a, b, c\}$

~~bcb~~ ~~abbcabba~~

abbab c baabba ~~or~~ c

b b

$S \to c$

$S \to aSa \mid bSb$

---

ww^R even length palindrome

$w \in \Sigma^*$ → $S \to aSa \mid bSb \mid \varepsilon$

$w \in \Sigma^+$ $S \to aSa \mid bSb \mid a \mid b$

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

$S \to AB$

$A \to aA \mid a$

$B \to bB \mid b$

$L = \{ \varepsilon, a, aa, aaa, b, bb, bbb \}$

$a - b$ $\quad aAb \mid B$

abb

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

$S \to AB$

$A \to aA \mid a$

$B \to bbB \mid bb$

② $L = \{ a^n b^m : n \leq m+3 \}$ $\quad n \geq 0, m \geq 0$

$m = 0, \; n = 3$

$m = 1, \; n = 4$

$m = 2, \; n = 5$

(b) (a)

b^*

$0, 1, 2$

$a^n b^n$

$\Rightarrow S \to aSb \mid ab$

$S \to aSbb \mid abb$

$S \to AB$

$A \to aAb \mid B$

$B \to bB \mid \varepsilon$

$A \to B \mid aAb$

$S \to AB \Rightarrow AbB \Rightarrow AbB \Rightarrow Ab\varepsilon$

$\Rightarrow aAbb$

$\Rightarrow aBbb$

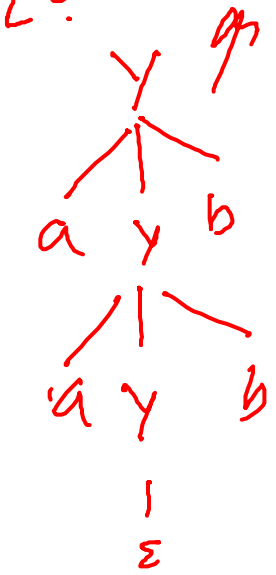
$\Rightarrow abb$

Ex → $S \to XY$
$X \to aX \mid a$
$Y \to aYb \mid \varepsilon$

① Pick last production → lay
② Same process for above production.

① $Y \to aYb \mid \varepsilon$

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



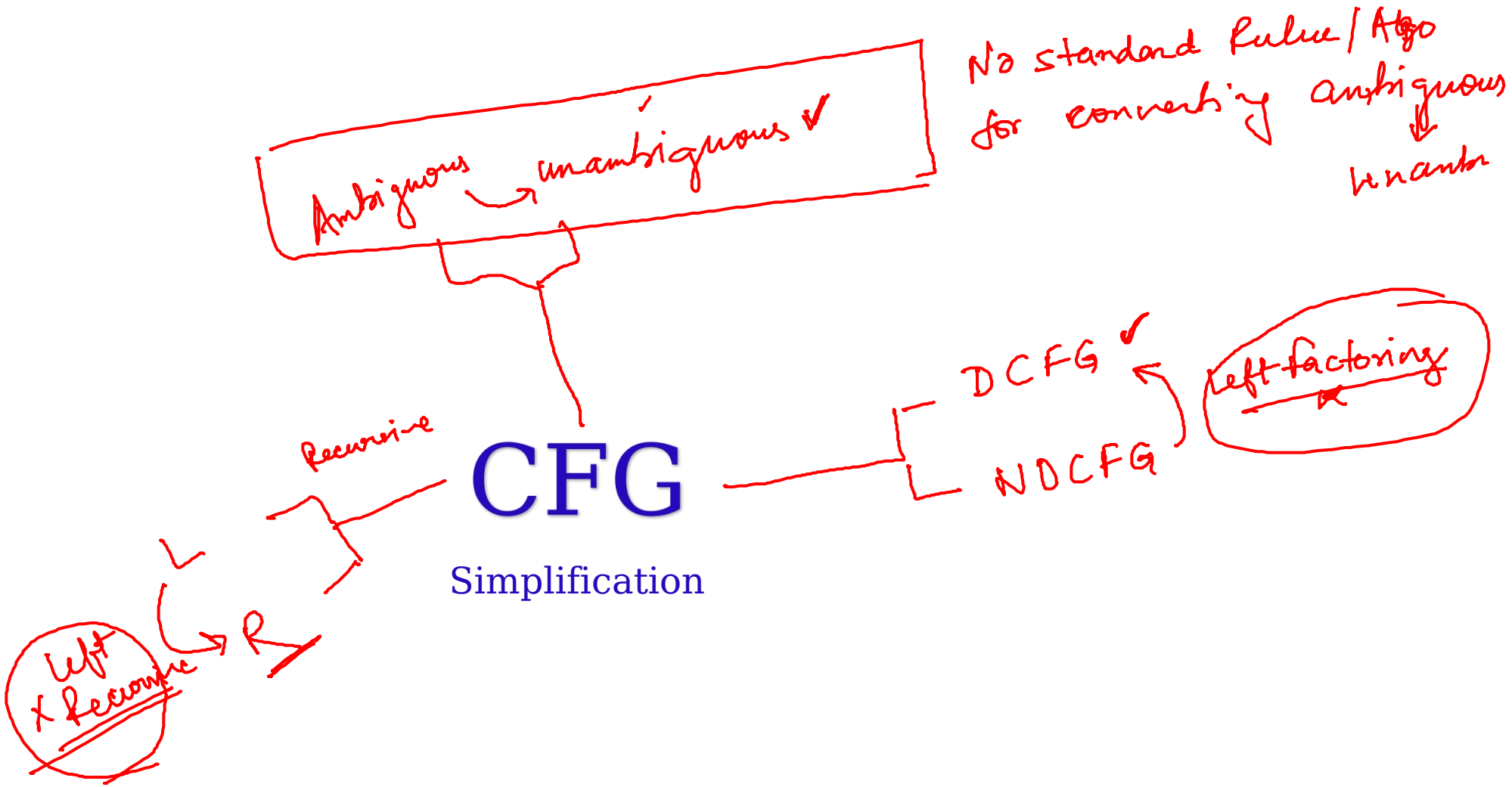
② $X \to aX \mid a$

$\{a^m : m \geq 1\}$

③ $S \to XY$

$\{a^m \cdot a^n b^n : m \geq 1, n \geq 0\}$

$\{a^m (a^n b^n) : m > n, n \geq 0\}$

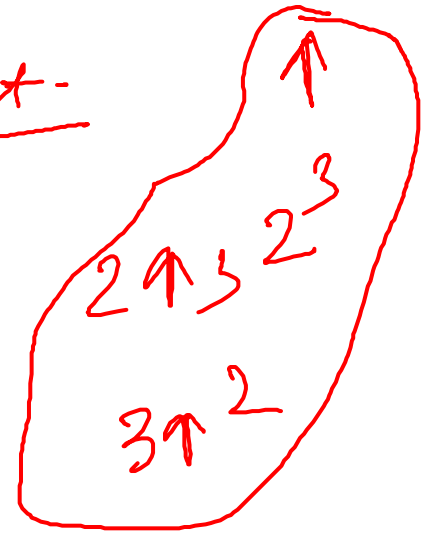

Ambiguous → unambiguous ✓
No standard Rules / Algo for converting ambiguous unambiguous
CFG
Simplification
Recursive
L
R
Left Recursive ✗
DCFG ✓
NDCFG
Left factoring ✗

Ambiguity → CFG is ambiguous if $w \in L(G)$ having more than one parse tree (different LMD or RMD)

then $L(G)$ → inherently ambiguous language:

Cause of Ambiguity ⇒ (I) precedence of operators is not in respected order.

(II) Associativity
— from left to Right.



No standard solution for Ambiguity

⇒ Undecidable

Left Factoring → ① grammar is NDCFG. → DCFG.

$$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3 \mid \cdots \mid \alpha\beta_n$$

$$\Downarrow$$

$$\left\{ \begin{array}{l} A \rightarrow \alpha A' \\ A' \rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid \cdots \end{array} \right\}$$

LL ≠ RL

Left Recursive → Left factoring

Views

Not go

$A \rightarrow A\alpha \mid \beta$ ✓

$\Downarrow$

$A \rightarrow \beta A'$ ✓

$A' \rightarrow \alpha A' \mid \varepsilon$

How to find ambiguity using first & follow?

$L_1 =$

$L_2 =$

$S \rightarrow (L) \mid x$

$L \rightarrow L, S \mid S$ ✓

$\Rightarrow$

$S \rightarrow (L) \mid x$

$L \rightarrow S\,L'$

$L' \rightarrow , S\,L' \mid \varepsilon$

CFG1 =

CFG2 =

Assignment

CFG → find Ambiguity ~~find~~ ~~using~~ without parsetree method

Simplification of CFG

1. to remove ε-production
2. to remove unit production
3. To remove unless unreachable symbol

$A \to Sa \mid b \mid \varepsilon$

$S \to A$

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

$S \to ab$

$S \to aSb$

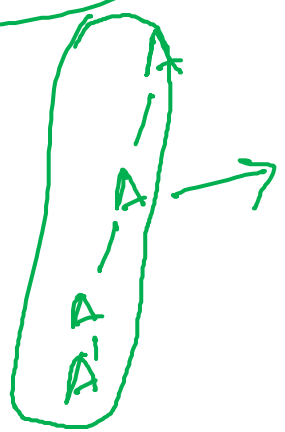
$S \to aSb \mid \varepsilon$

$X \to \varepsilon \to Y$

$A \to B$

$A \to A \mid Ba$

$b \to ba$



1. Remove ε- production $(A \to \varepsilon)$

Before : $B \to xAy$ & $A \to \varepsilon | a$

After : $B \to xAy | xy$, $A \to a$

Ex → $S \to aSb | aAb$

$A \to \varepsilon$ | X

Sol ⇒ $S \to aSb | aAb | ab$ ✓

Remaining write grammars as it

1. ~~visit~~
2. without symbol addon.

Ex → Even length string $((a+b)^*(a+b)^*)$

$S \to AA$

$A \to aA | bA | \varepsilon$ ⇒

$S \to AA | A$

$A \to aA | bA | a | b$

Note : → if start symbol produce ε , cannot eliminate it

2. Remove Unit Production ~~(A → A)~~ (A → B)

Ex → S → Aa | B
B → A | bb
A → a | bc

Sol → find out unit production S → B

S → B → bb
A → a
bc

⇒ S → B → bb | a | bc

{S → Aa | a | bc | bb
B → A | bb
A → a | bc}

⇒ unit production
B → A ⇒ B → a | bc

B → A → a
bc

~~B → B~~

⇓

S → Aa | a | bc | bb
B → a | bc | bb
A → a | bc

Unit p

3. To Remove Useless / unreachable symbol (variable)

$A \to aab$, $A \to aSb$

$S \to aAb$, $A \to ab$ → useful; $B \to \varepsilon$ → useless

- → variable derive string (
- → variable which is not used in derivation of any string is useless symbol
- → variable reached from start state is useful —

Step →

1. find out symbols which generate terminals (Derivability)
2. check remaining variables —
   - useful ⇒ derive some terminals
   - useless → otherwise

— Remove useless symbols & its productions (Reachability)

Ex → ① ε-production
② unit
③ useless | unreachable

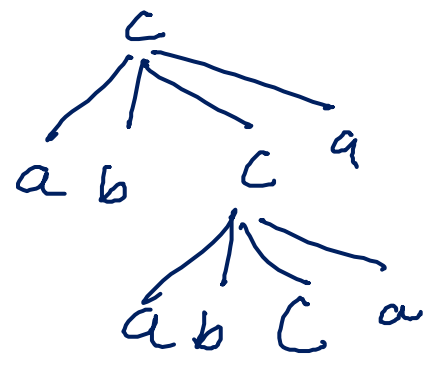
✓S → AB | ~~AC~~
✓A → aAb | bAa | a
✓B → bbA | aaB | AB
~~C → abCa | aDb~~
~~D → bD | aC~~

S → AB |
A → aAb | bAa | a
B → bbA | aaB | AB

① {a, b, A}
② B → bbA → bba
B is usefull
{a, b, A, B}
③ S → AB → abba
S is usefull
{a, b, A, B, S}
④ ⟹ C is not usefull
⑤ D is not usefull

{a, b, A, B}

C → abCa → X Not usefull



C → aDb

Ex →

$\{S \to AB \mid a,\ A \to BC \mid b\} \Longrightarrow$

~~$B \to aB \mid C$~~

~~$C \to aC \mid B$~~

$\{a, b, A, S$

$B \to aB \to aC \to aB \to$ Not terminating
Not generating string

B is useless

$C \to BaC \to aaC \to aaaC \to$ Not generating string

C is useless

$S \to a$ ✓

~~$A \to b$~~ $\Longrightarrow$ A is unreachable symbol so A can remove

↓

final $S \to a$

$S \rightarrow ABC \mid BaB$

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

$B \rightarrow bBb \mid a$

$C \rightarrow aCA \mid ABC$

$\Rightarrow$

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

$S \rightarrow BaB$

$A \rightarrow aA \mid aaa$

$B \rightarrow bBb \mid a$

A is unreachable symbol

$S \rightarrow BaB$

$B \rightarrow bBb \mid a$

$G=(V, \Sigma, P, S)$

(CFG)

Normal forms

Chomsky Normal forms

$V \rightarrow V \cdot V$ or $V \rightarrow T$

$S \rightarrow aAB \Rightarrow S \rightarrow BAB$

$S \rightarrow AAB$

$A' \rightarrow AA$

$A \rightarrow a$, $B \rightarrow b$

$A' B$

Greibach Normal forms

$V \rightarrow T$ or $V \rightarrow T \cdot \alpha$

$\alpha \in V$

$\alpha \rightarrow$ no. of variable

CFG ⟶ CNF { V → T or V → VV }

Steps → ① start variable not allowed in R.H.S of production rule.
New start state S_0, & New rule $S_0 \rightarrow S$

② Remove ε-production ✓

③ Remove unit " ✓

④ {Replace problematic terminals (string)
$A \rightarrow ab \Rightarrow$ { $A \rightarrow T_a T_b$, $T_a \rightarrow a$, $T_b \rightarrow b$ }

⑤ {Sequence of variable → two variable
$A \rightarrow ABCD \Rightarrow A \rightarrow A'B'$, $A' \rightarrow AB$, $B' \rightarrow CD$

Ex → $V \to T$, $V \to V \cdot V$

$S \to XSX \mid aY$

$X \to Y \mid S$

$Y \to b \mid \varepsilon$

CFG → CNF

① New start variable

$S_0 \to S$ ⇒

$S_0 \to S$

$S \to XSX \mid aY$

$X \to Y \mid S$

$Y \to b \mid \varepsilon$

② Remove ε – production ($Y \to \varepsilon$)

$S_0 \to S$

$S \to XSX \mid aY \mid XS \mid SX \mid S$

$X \to Y \mid S$

$Y \to b$

③ Remove unit production

$S_0 \to XSX \mid aY \mid XS \mid SX \mid a$

$S \to XSX \mid aY \mid XS \mid SX \mid a$

$X \to XSX \mid aY \mid XS \mid SX \mid a$

$Y \to b$

④ problematic terminals

$S_0 \to XSX \mid T_aY \mid XS \mid SX \mid a$

$S \to XSX \mid T_aY \mid XS \mid SX \mid a$

$X \to XSX \mid XS \mid SX \mid T_aY \mid b$

$Y \to b$

$T_a \to a$

⑤ Sequence → Variables

$S_0 \to XX_1 \mid T_aY \mid XS \mid SX$

$S \to XX_1 \mid T_aY \mid XS \mid SX$

$X \to XX_1 \mid SX \mid XS \mid T_aY \mid b$

$Y \to b$

$T_a \to a$

$X_1 \to SX$

CFG to GNF

CFG → CNF

&

simplification of CFG

- ε-production
- unit production
- Remove useless / unreachable symbols (variables &tm

CFG $\xrightarrow{\text{into}}$ GNF

$\{A_j, A_i\}$

variable → Terminal variable . α

α → combination of

Rules →

1. Rename variables $A_1, A_2, A_3, \ldots$
2. $A_i \longrightarrow A_j \alpha \quad \forall \; i \leq j$
3. check left recursion, & Remove Left Recursion.
4. Get GNF

$\{ S \to A,\ A_1 \to A_2 \}$ (1)

$A_1 \to A_2$ hold condition

Ex → $\begin{cases} S \to AA \mid a \\ A \to SS \mid b \end{cases}$

GNF → { ~~A~~ Variable → ~~Variable α~~ $T \cdot \alpha$ or $V \to T$ }

Sol → ① Rename variables $S \Rightarrow A_1$, $A \Rightarrow A_2$

$$A_1 \to A_2 A_2 \mid a$$

$$A_2 \to A_1 A_1 \mid b$$

② Verify $A_i \to A_j \; \forall \; i \le j$

$A_1 \to A_2 A_2 \mid a$; hold the condition.

$A_2 \to A_1 A_1 \mid b$, Not true, so modify rules

$A_2 \to A_2 A_2 A_1 \mid aA_1 \mid b$ and again verify. true

⇒ ✓ $A_1 \to A_2 A_2 \mid a$

$A_2 \to A_2 A_2 A_1 \mid aA_1 \mid b$

↓

Left Recursive

$V \longrightarrow T$

$V \longrightarrow T \cdot \alpha$

---

(3) Remove left Recursion —

$A_1 \rightarrow A_2 A_2 / a \rightarrow$ OK

$A_2 \rightarrow A_2 A_2 A_1 / a A_1 / b \rightarrow$ LeftRecursion

Now Remove Left Recursion from production rules

$A_2 \rightarrow a A_1 / b / a A_1 B_2 / b B_2$

$B_2 \rightarrow A_2 A_2 A_1 / A_2 A_2 A_1 B_2$

The Resultant grammar

$A_1 \longrightarrow A_2 A_2 / a$

$A_2 \longrightarrow aA_1 / b / aA_1 B_2 / bB_2$

$B_2 \rightarrow A_2 A_2 A_1 / A_2 A_2 A_1 B_2$

✓

GNF

A final GNF

$A_1 \rightarrow aA_1 / aA_1 A_2 / bA_2 / aA_1 B_2 A_2 / bB_2 A_2$

$A_2 \rightarrow aA_1 / b / aA_1 B_2 / bB_2$

$B_2 \rightarrow aA_1 A_2 A_1 / bA_2 A_1 / aA_1 B_2 A_2 A_1 / bB_2 A_2 A_1 / aA_1 A_2 A_1 B_2 / bA_2 A_1 B_2 / aA_1 B_2 A_2 A_1 B_2 / bB_2 A_2 A_1 B_2$

Ex → CFG → GNF

$S \to XA \mid BB$

$B \to b \mid SB$

$X \to b$

$A \to a$

① Rename variable

$A_1 \to A_2 A_3 \mid A_4 A_4$

$A_4 \to b \mid A_1 A_4$

$A_2 \to b$

$A_3 \to a$

② Verify $A_i \to A_j \quad \forall i \le j$

$A_4 \to A_1 A_4 \mid b$ → Not true

$A_4 \to A_2 A_3 A_4 \mid A_4 A_4 A_4 \mid b$

Again verify $A_i \to A_j, \forall i \le j$

Replace A_2 value

$A_4 \to b A_3 A_4 \mid A_4 A_4 A_4 \mid b$

Resultant grammar

$A_1 \to A_2 A_3 \mid A_4 A_4$

$A_4 \to b A_3 A_4 \mid A_4 A_4 A_4 \mid b$

$A_3 \to a, \quad A_2 \to b$

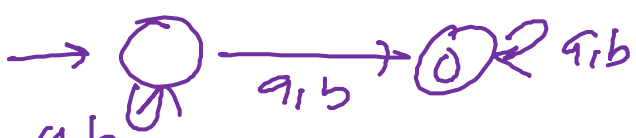
③ check left Recursive

$A_4 \to A_4 A_4 A_4$ → Left Recursive → Remove

$A_4 \to b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$ → GNF

$Z \to A_4A_4Z \mid A_4A_4$

④ Get GNF



$xy^i z$

(1) $(0+1)^*(0+1)(0+1)^*$ ⇒ No. of state in DFA — 2019

xyz

$i = 1$

(2) $\Sigma = \{a, b\}$, $L = \{ x : x = a^{2+3K}$ or $x = b^{10+12K}, K \geq 0 \}$

which one is pumping length

(I) 3K

$K = 0$: a^2, b^{10}

$K = 1$

(II) 9 X

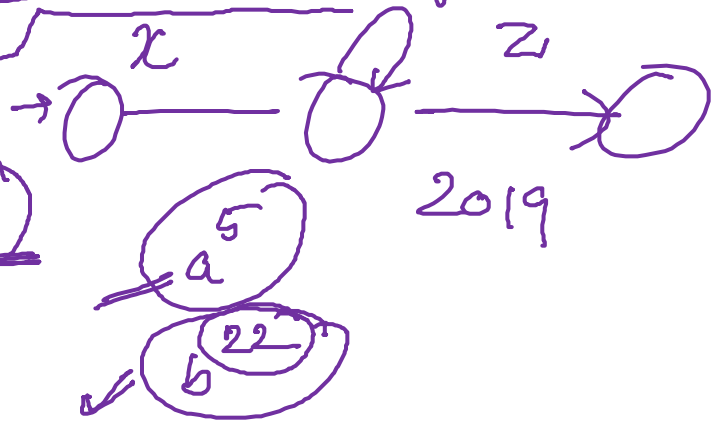
a^5, b^{22}

(III) 5 ✗

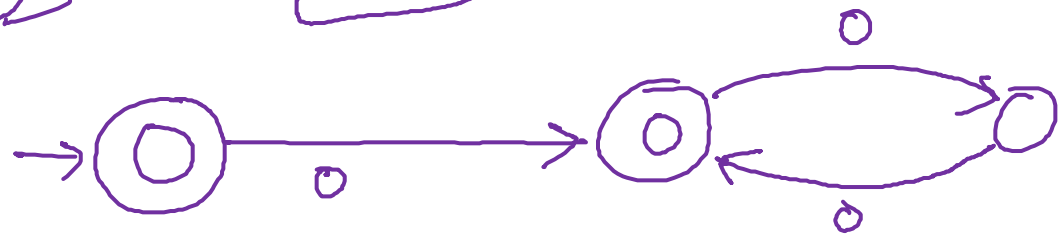
$K^3 = 3$

a^{11}, b^{46}

(IV) 24



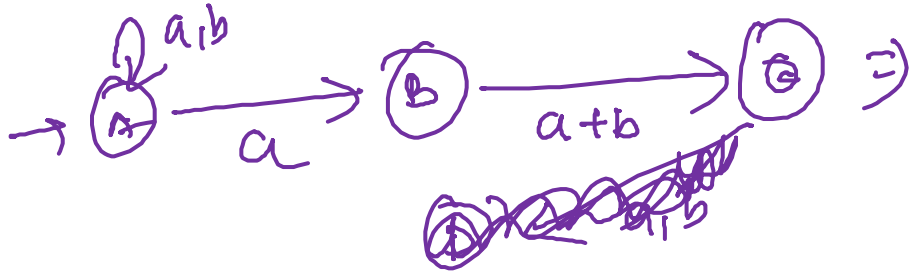
(3)



RE ⇒ $\varepsilon + 0(00)^*$

(1) 2017 $(a+b)^* b(a+b)$ → No of state in DFA

(i) 4, (ii) 5, (iii) 6, (iv) 7



	a	b
A		

(2) $S \to SaS \mid aSb \mid bSa \mid SS \mid \varepsilon$

which one of the following strings is not generated by G.

(a) abab

(b) aaab

(c) abbaa

(d) babba ✓