Unit III

DATE: / /
PAGE NO.:

Ch 12 Context-Free Grammars.

Syntax as a method for defining lang. —

Complicated algebraic expressions need to be written in one line of std. symbols.

ex. $\frac{\frac{1}{2}+9}{4+\frac{8}{21}+\frac{5}{3+1/2}}$ = ((1/2)+9)/(4+(8/21)+(5)(3+(1/2))))

↓ easy for us to understand & guess the answer.

↓ needed by comp.$^{r}$ prog. But to understand it, rules are needed.

Also, comp.$^{rs}$ should reject (9)+

Compiler — super prog. does conversion from high-level lang. into a m/c executable lang.

To define arithmetic expr$^{n}$?

Rule 1 : Any No is in the set AE.

2 : If x & y are in AE, then so are

(x) -(x) (x+y) (x-y) (x*y) (x/y) (x**y).

ex. ((3+4)* (6+7)), m/c interpretes this as

3 is in AE, 4 is in AE, (3+4) is in AE.

6 " " 7 " " (6+7) "

((3+4)* (6+7)) "

} part I: expr$^{n}$. prod$^{n}$

Algorithm converts this into,

LOAD 3 in register 1

" 4 " 2

ADD reg$^{r}$. 2 into reg$^{r}$. 1.

LOAD 6 in r3

" 7 r4

ADD to r4

MULTIPLY r1 by r4.

} part II evaluation.

} depends on comp$^{r}$ architecture

Recognizing the struct of a comp$^{r}$. lang. instruction ≈ Rec. struct of a sentence in a human lang.

Syntax.
↓
set of rules (Prod.ns)
↓
Terminals & Non-terms.

substitutions
(Derivation / word Generation)

Grammar is set of rules by which valid sentences in a lang. are constructed.
Determining how a sentence can be formed from the rules of grammar is called parsing the sentence.

Set of rules form a generative grammar. From them & a dictionary (the alphabet) we can generate all the sentences (words) in the lang.

Syntax – rules that do not involve the meaning of words.
Semantics – rules involve meaning of words.
Rules of compr. lang. grammar are all syntactic & not semantic.

X = B+9 → const. should not be out of range.
do not divide by 0, take square root of a -ve No,
do not mix fixed point No. with floating pt Nos. in bad ways.

A law of grammar is a possible substitution. The arrow indicates (⇒) that a substitution was made according to the preceding rules of grammar.

Terminals – words that can't be replaced by anything.
Non-Terminals – " " must " " " other things.

The job of sentence production is not complete until all the nonterminals have been replaced with terminals.

To define arithmatic expression:–

atom → (AE)
AE → (AE+AE)
AE → − | * | /

AE → (AE)
AE → −(AE)
AE → Any-Number

DATE: / /
PAGE NO.:

Apart from start, the only non-terminal is AE.
Terminals are Any-No. & symbols.

✓ To define Any-No,

Any-No → First Digit
First Digit → First D. Other D.
First D → 1 2 3 ... 9
Other D → 0 | 2 ... 9.

To produce No. 106,

Any-No → First D.
→ F. D. Other D
→ F. D. O D other D.
→ 1 0 6

Sequence of applications of the rules that produces the finished string of terminals from the starting symbol is called a derivation or a generation of the word.

The grammatical rules are often referred to as productions. Derivation may or may not be unique. But we may still produce the same finished product.

diff. Deriv. → same words

✓ CFG – invented by linguist Noam Chomsky in 1956.

Def^n – CFG is collection of 3 things.

① An alphabet Σ of letters called terminals from which words of lang. are formed.
② A set of symbols called non-terminals, one of which is the symbol S start.
③ A finite set of productions of the form,
One Nonterminal → finite string of terminals &/or Nonterminals. or E.

At least one production has nonterminal S at LHS.

Non-tes – capital, Term – lowercase

DATE: / /
PAGE NO:

## CFL →

CFL – Context free lang.

The lang. generated by a CFG is the set of all str. of terminals that can be produced from the start symbol S using the productions as substitutions.

ex. let a be a terminal,

Prod.1 $S \to aS$

prod.2 $S \to \Lambda$.

→ If we apply prod.1 n times followed by 1 applic.ⁿ of prod.2, str. $a^n$ is generated.

" 2 alone generates null str.

Thus CFL = $a^*$.

$S \Rightarrow aS$
$\Rightarrow aaS$
$\Rightarrow aa$

} unique way for derivation.

→ : used in statement of prod.ⁿ It means "can be replaced by". $S \to aS$.

⇒ : used between the unfinished stages in the generation of a word. means "can develop into"
$aaS \Rightarrow aaaS$.

Unfinished stages are strings of terminals & non-terminals called as working strings.

ex. term. = a

Prod.1 $S \to SS$
$S \to a$
$S \to \Lambda$.

} CFL $a^*$
many ways to obtain str.

✓ Λ is not a nonterminal ∵ there is no prod.ⁿ of the form $A \to$ something.

Λ " " terminal ∵ it vanished from the finished str.

ex. $\Lambda aa\Lambda = aa$.

Λ is a very special symbol & has its own status.

For non-term. N., $N \to \Lambda$, means whenever we want, N can simply be deleted from any place in a working string.

ex. $S \to X$ CFL: $(a+b)^*$

$S \to Y$

$X \to \wedge$

$Y \to aY$

$Y \to bY$

$Y \to a$

$Y \to b$.

ex. $S \to aS$ CFL: $(a+b)^*$

$S \to bS$ Same lang. is generated if we delete 3 & 4 Prod$^{ns}$.

$S \to a$

$S \to b$

$S \to \wedge$

ex. $S \to XaaX$ CFL: $(a+b)^* aa (a+b)^*$

$X \to aX$

$X \to bX$

$X \to \wedge$.

✓ ex. $S \to XY$ CFL: $(a+b)^* aa (a+b)^*$

$X \to aX$

$X \to bX$

$X \to a$

$Y \to Ya$

$Y \to Yb$.

$Y \to a$

Even-Even- $S \to aaS$

$S \to bbS$

$S \to TSTS$

$T \to ab$

$T \to ba$

$S \to \wedge$

ex. $S \to SS$

$S \to$ BALANCED $S$

$S \to S$ BAL.

$S \to \wedge$

$S \to$ UNBAL $S$ UNBAL.

BAL $\to aa$

BAL $\to bb$

UNBAL $\to ab$

UNBAL $\to ba$

} CFL: EVEN-EVEN.

ex. $S \to aSb$ CFL: non-reg lang. $\{a^n b^n\}$.
$S \to \Lambda$.

ex. $S \to aSa$ CFL: all words produced are in
$S \to bSb$ lang. Palindrome.
$S \to \Lambda$ but all words in Palin.
can't be generated by this gram.

2 flavors of palin. → i) unique central letter.
ii) even length. → Evenpalindrome

To prove that any word in Evenpalindrome can be produced from this grammer,
i) Scan 1st half of the word
ii) When we encounter a, apply prod" 1
iii) " " b " 2.
iv) When we finish 1st half, apply 3.

ex. abbaabba, apply 1 2 2 1 3

ex. $S \Rightarrow aSa$ CFL: entire lang. palindrome.
$S \Rightarrow bSb$
$S \Rightarrow a$
$S \Rightarrow b$
$S \Rightarrow \Lambda$

ex. $S \Rightarrow aSa$ CFL: $\{a^n b a^n\}$.
$S \Rightarrow b$ lang. $\{a^n b a^n b^{n+1}\}$ can't be generated by any CFG.

Give CFL & derive CFG.

ex. $S \to aB$
$S \to bA$
$A \to a$
$A \to aS$
$A \to bAA$
$B \to b$
$B \to bS$ &
$B \to aBB$.

CFL: long. Equal = {ab ba aabb abab abba baab baba ...}

DATE: / /
PAGE NO.:

*Notations – Vertical line | is used for disjunction as

$S \to aS$ becomes $S \to aS \mid \Lambda$.
$S \to \Lambda$

Instead of $\to$, some use $::=$

nonterminals ≈ variables.

$\Lambda = \epsilon$ or $\lambda$. (null str.)

nonterm. in angle brackets – $\langle S \rangle \to \langle X \rangle$
$\langle X \rangle \to a \mid \Lambda$.

nonterm – capital

term – lowercase → do not appear in left of prod?
except $\Lambda$.

✓ Backus Normal (Naur) Form (BNF) –
format for presenting CFG – arrows, ↑, term. & non-term. invented by John Backus & Peter Naur.

Fortran id? (variable) can be 6 alphanumeric char, start with letter

✓ CFG: Identifier → Letter & XXXXX
X → Letter | Digit | Λ
Letter → A | B | ... | Z
Digit → 0 | 1 | ... | 9.

Lang. of all proper instn? can be defined by CFG for C, pascal, Basic & so on

A comp? must determine the grammatical stru. of a comp? lang. Statement. before it can execute the instn?

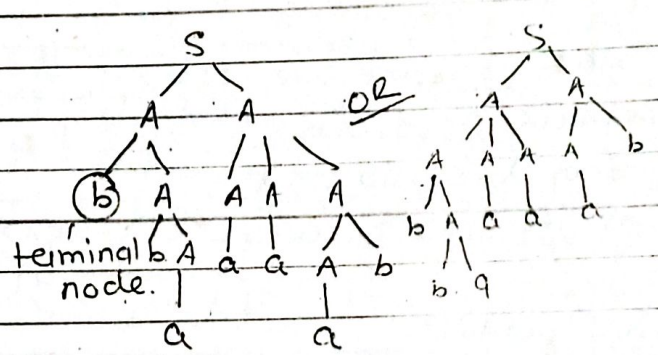
Grammar, $G = (V, T, P, S)$.
V – Non-Term / variables, T – Term., S – Start symbol, P – prod? $\alpha \to \beta$.

# Trees — syntax / parse / generation / prod$^n$ / derivation Trees.

Start with symbol S. Every time we use a prod$^n$ to replace a non-terminal by a string, we draw downward lines from the nonterminal to each char. in the string.

ex. $S \to AA$
$A \to AAA \mid bA \mid Ab \mid a$

$S \to AA$
$A \to bA$ &
$A \to AAA$.




$A \to bA$
$A \to a$
$A \to a$
$A \to Ab$ ... word produced — bbaaaab

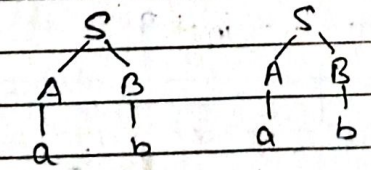
$A \to a$

---

## Ambiguity —

ex. $S \to AB$ 2 derivations to produce ab.
$A \to a$ $S \Rightarrow AB \Rightarrow aB \Rightarrow ab$ or
$B \to b$. $S \Rightarrow AB \Rightarrow Ab \Rightarrow ab$.

But when we draw corresponding syntax trees, 2 derivations are same:




Thus there is no ambiguity of interpretation.

**Def$^n$** A CFG is called ambiguous if for at least one word in the lang. that it generates there are two possible derivations of the world that correspond to different syntax trees.

face

DATE: / /
PAGE NO.:

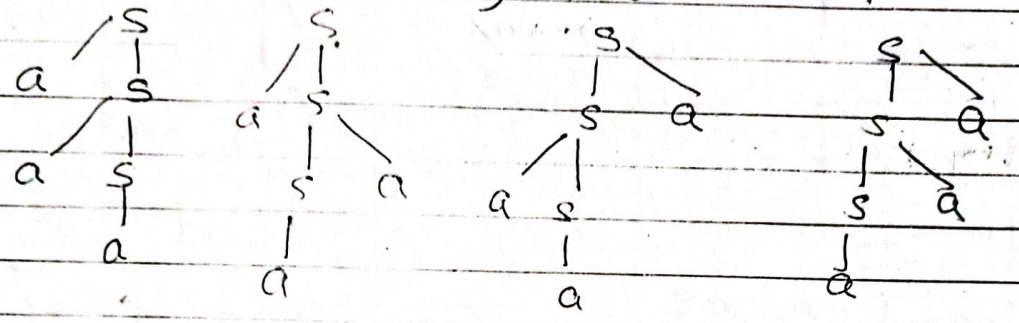
ex. $S \to aSa \mid bSb \mid a \mid b \mid \wedge$

Palindrome → unambiguous.

ex. lang. of all nonnull strings of a's, R.E. $a^+$

$S \to aS \mid Sa \mid a$

$a^3$ can be generated by 4 diff. trees:

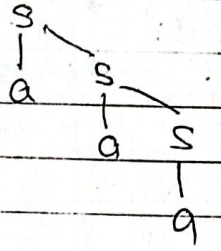


∴ ambiguous. (CFG is ambi. not the lang.).

Same lang. can also be defined by CFG,

$S \to aS \mid a$

$a^3$ has only 1 prod?



---

- Grammatical Format

Regular Grammars.

Reg. lang. — definable by RE.

All reg. lang. can be generated by CFG's & so can some non-reg. lang.

Def^n Semiword — For a given CFG, a semiword is a string of terminals (maybe none) concatenated with exactly one nonterminal (on the right).

(terminal (term) ... (term) (nonterm.)

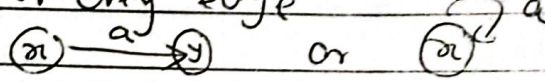
Th^m - Given any FA, there is a CFG that generates exactly the lang. accepted by the FA.

i.e. all reg. lang. are context-free lang.

Steps to create CFG starting from FA:

(1) non terminals in CFG will be all the names of the states in FA with start state S.

(2) For every edge

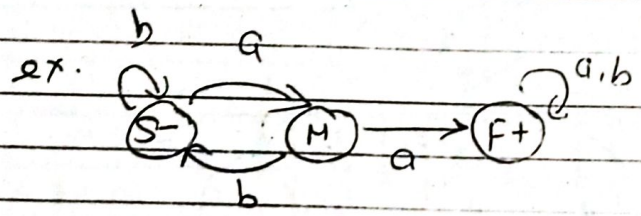



create prod? $X \to aY$ or $X \to ax$

Do same for b-edge.

(3) for every final state X, create prod $X \to \Lambda$.

ex.




CFG: $S \to aM \mid bS$

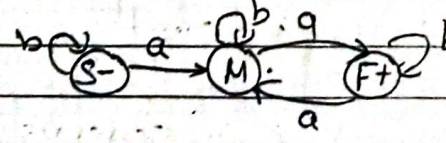
$M \to aF \mid bS$

$F \to aF \mid bF \mid \Lambda$.

word baab is accepted by this FA through full sequence of semipaths:

| | Corresponding CFG prod$^{n}$ |
|---|---|
| S | |
| bS | $S \to bS$ |
| baM | $S \to aM$ |
| baaF | $M \to aF$ |
| baabF | $F \to bF$ |
| baab | $F \to \Lambda$ |

Ex. The lang. of all words with an even No. of a's (with at least some a's) can be accepted by this FA:




$S \to bS \mid aM$

$M \to bM \mid aF$

$F \to aM \mid bF \mid \Lambda$

DATE: / /
PAGE NO.:

Thm – If all the prodns in given CFG fit one of the two forms

Nonterm. → semiword
or " → word.

(word may be Λ), then the lang. generated by this CFG is regular & such grammar is called a regular gram.

Thm – A CFG is called a regular grammar if each of its prodns is of one of the 2 forms

Nonterminal → semiword
or " → word.

Thus all reg. lang. can be generated by reg. gramm. & all reg. gram. generate reg. lang.

It is still possible that a CFG that is not in the form of reg. gram. can generate a reg. lang.

⇒ Steps to build TG from CFG:

CFG: $N_1 \to w_1 N_2$ $\quad N_7 \to w_{10}$
$N_1 \to w_2 N_3$ $\quad \cdots$
$N_2 \to w_3 N_4$
$\cdots$

$N$'s are nonterminals, $w$'s are strings of terminals & the parts $w_y N_z$ are the semiwords used in prodns

Let $N_1 = S$.

Draw a circle for each $N_i$ & one for +.

(S) (N1) (N2) ... (+)

For every prodn rule $N_\alpha \to w_y N_z$

$(N_\alpha) \xrightarrow{w_y} (N_z)$

If $N_\alpha = N_z$, the path is a loop.

For $N_p \to w_q$ $\quad (N_p) \xrightarrow{w_q} (+)$

Every prodn of a word in this CFG:

$S \Rightarrow wN \to wwN \to \cdots wwww$ corresponds to a path in this TG from + to +

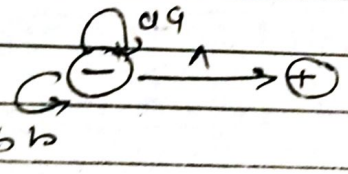
∴ the lang. of this TG is exactly same as that of the CFG. ∴ lang. of CFG is regular.

ex. CFG: $S \to aaS \mid bbS \mid \Lambda$

Only 1 non-term. ∴ 2 states in TG — & +.

Q. $Np \to wq$ is $S \to \Lambda$ ∴ only 1 edge into +, labeled Λ.

$S \to aaS$ & $S \to bbS$ ≅ $N_1 \to wN_1$ ; loops.




by Kleene's th$^m$, any lang. accepted by a TG is regular, ∴ lang. generated by this CFG is regular.

RE : $(aa+bb)^*$.

Page No.:
Date: / /

**Killing ∧ prod^ns :-** ∧ prod.^ns are not necessary in a grammar for a context free lang. that does not contain the word ∧.

**Th^m** If L is a context-free lang. generated by a CFG that includes ∧ prod.^ns, then there is a diff. CFG that has no ∧-prod.^ns that generates either the whole lang. L (if L does not include the word ∧) or else generates the lang. of all the words in L that are not ∧.

→ Purpose of the prod^n N → ∧ is to delete N from the working string.
We delete above prod.^ns & add others.

(1) for all prod.^ns, X → (blah1) N (blah2)
X - any non-term. (even S or N),
blah1 & blah2 - anything (even involving N),

add X → (blah1) (blah2).

[we do not delete X → (blah1) N (blah2), only N → ∧].

(2) For all prod.^ns that involve more than one N on right side, add new prod.^ns that have the same other char. but that have all possible subsets of N's deleted

ex. X → aNbNa

add, X → abNa
X → aNba
X → aba

ex. X → NN

add, X → N
X → ∧.

This new CFG generates exactly the same words as the first grammar with the possible exception of word ∧.

✓ ex. Even Palindrome
$S \to aSa \mid bSb \mid \Lambda$.

remove $S \to \Lambda$, replace it with
$S \to aa$, $S \to bb$

CFG: $S \to aSa \mid bSb \mid aa \mid bb$: also generates Even pallindrome, except word $\Lambda$.

| old CFG | | New CFG | |
|---|---|---|---|
| Derivation: | prod$^n$ used | | |
| $S \Rightarrow aSa$ | $S \to aSa$ | $S \Rightarrow aSa$ | $S \to aSa$ |
| $\Rightarrow aaSaa$ | $S \to aSa$ | $\Rightarrow aaSaa$ | $S \to aSa$ |
| $\Rightarrow aabSbaa$ | $S \to bSb$ | $\Rightarrow aabbaa$ | $S \to bb$. |
| $\Rightarrow aabbaa$ | $S \to \Lambda$. | | |

Problem – rule may create new $\Lambda$ prod$^{ns}$ that can't themselves be removed without again creating more. ex.

$S \to a \mid Xb \mid aYa$
$X \to Y \mid \Lambda$
$Y \to b \mid X$
→ . . . .

To eliminate $X \to \Lambda$, we add,
$S \to b$ & $Y \to \Lambda$.
To elim. $Y \to \Lambda$, we add,
$S \to aa$ & $X \to \Lambda$.

Trick is to eliminate all $\Lambda$-prod$^{ns}$ simultaneously.

- Def$^n$ We call a nonterminal $N$ nullable if
There is a prod$^n$ $N \to \Lambda$ or
" derivation that starts at $N$ & leads to $\Lambda$:
$N \Rightarrow \cdots \Rightarrow \Lambda$.

- Modified Replacement Rule –
1. Delete all $\Lambda$-prod$^{ns}$.

Page No.:
Date: / /

2. Add full prod$^{ns}$: for every prod$^{n}$ $X \to$ old string, add new prod$^{ns}$ of the form $X \to \cdots$, where right side is any modification of old str. that can be formed by deleting all possible subsets of nullable non-terminals. Except that we do not allow $X \to \Lambda$ to be formed even if all the char. in this old str. are nullable.

- ex. CFG: $S \to a \mid Xb \mid aYa$
  $X \to Y \mid \Lambda$
  $Y \to b \mid X$.

X & Y are nullable.
∴ when we delete $X \to \Lambda$, we have to check all prod$^{ns}$ that include X or Y to see for new prod$^{ns}$:

| Old prod$^{ns}$ with nullables | Prod$^{ns}$ Newly formed by Rule. | ∴ New CFG: |
|---|---|---|
| $X \to Y$ | Nothing | $S \to a \mid Xb \mid aYa \mid b \mid aa$ |
| $X \to \Lambda$ | " | $X \to Y$ |
| $Y \to X$ | " | $Y \to b \mid X$. |
| $S \to Xb$ | $S \to b$ | It has no $\Lambda$ prod$^{ns}$ but generates same lang. |
| $S \to aYa$ | $S \to aa$. | |

- ex. lang. $(a+b)^*a$ CFG: $S \to Xa$
  $X \to aX \mid bX \mid \Lambda$.

nullable – X

| Prod$^{ns}$ with nullables | New prod$^{ns}$ | New CFG: |
|---|---|---|
| $S \to Xa$ | $S \to a$ | $S \to Xa \mid a$ |
| $X \to aX$ | $X \to a$ | $X \to aX \mid bX \mid a \mid b$. |
| $X \to bX$ | $X \to b$ | |

∴ $\Lambda$ was not in old CFG, new CFG generates exactly same lang.

- ex. lang. $(a+b)^*bb(a+b)^*$
  inefficient CFG: $S \to XY$
  $X \to Zb$
  $Y \to bW$
  $Z \to AB$
  $W \to Z$
  $A \to aA \mid bA \mid \Lambda$
  $B \to Ba \mid Bb \mid \Lambda$.

X forms word ending in b, Y — starting with b.
∴ S " " with bb.

A, B, Z, W are nullables. Replacements:

| Old | New |
|---|---|
| X → Z b | X → b |
| Y → b W | Y → b |
| Z → AB | Z → A & Z → B. |
| N → Z | Nothing |
| A → a A | A → a |
| A → b A | A → b |
| B → Ba | B → a |
| B → Bb | B → b |

New CFG: generates same lang.

S → XY
X → Zb | b
Y → bW | b
Z → AB | A | B.
W → Z
A → aA | bA | a | b
B → Ba | Bb | a | b.

Killing Unit prod^ns —

Def^n — A prod^n of the form,
Nonterm. → one Nonterm. is unit prod^n.

Th^m. If there is a CFG for lang. L that has no λ-prod^ns then there is also a CFG for L with no λ-prod^ns & no unit prod^ns.

B → str.

⇒ purpose of A → B, is to chang working str.
(blah) A (blah) to (blah) B (blah)
& (blah) str blah.

i.e. we need A → str.

Elimination rule —
for every pair of non-term. A & B, if CFG has A → B or A ⇒ X1 ⇒ X2 ⇒ ... B then introduce new prod^ns:
If nonunit prod^ns from B are —
B → s1 | s2 | ... (str.)
create A → s1 | s2 | ...
Do this for all such pairs of A & B simultaneously

Page No.:
Date: / /

ex. $S \to A|bb$
$A \to B|b$
$B \to S|a$

Unit prod$^{ns}$ & Non-

$S \to A$ $\quad S \to bb$
$A \to B$ $\quad A \to b$
$B \to S$ $\quad B \to a.$

$S \to A$ gives $S \to b$
$S \to A \to B$ " $S \to a$
$A \to B$ " $A \to a$
$A \to B \to S$ " $A \to bb$
$B \to S$ " $B \to bb$
$B \to S \to A$ " $B \to b.$

∴ CFG:
$S \to bb|b|a$
$A \to b|a|bb$
$B \to a|bb|b.$

It generates finite lang..
∵ no nonterm: in any str. produced from S.

CNF (Chomsky Normal Form) -
separate taminal from nonterm.

Th$^{m}$ - L, CFG: there there is another CFG that generates all non Λ words of L, all of whose prod$^{ns}$ are of one of 2 forms:
Nonterm → str of only Nonterm.
" → 1 term.

→ If a, b are term.
add 2 non term A & B & prod$^{ns}$ $A \to a$, $B \to b$.
Prod$^{n}$ involving terminals, replace each a with A & b with B.

ex. CFG: $S \to X_1|X_2aX_2|aSb|b$
$X_1 \to X_2X_2|b$
$X_2 \to aX_2|aaX_1$

CFG$_1$: $S \to X_1$ $\quad X_1 \to X_2X_2$
$S \to X_2AX_2$ $\quad$ ~~$X_1 \to B$~~
$S \to ASB$ $\quad X_2 \to AX_2$
~~$S \to B$~~ $\quad X_2 \to AAX_1$

$A \to a$
$B \to b$

One problem is that we may create unit prod$^{ns}$ where none existed before.

ex. $X \to a$ becomes $X \to A$
$A \to a$.

∴ if any prod$^{n}$ is already in desired forms should be left alone.

ex. $S \to NA$ Here instead of using new nonterm. A, one can used N.
$N \to a|b$.
⇓
$S \to NN$
$N \to a|b$ is not allowed. ∵ bb is not generated by 1st CFG but by 2nd

∴ CFG: $S \to NA$
$N \to a|b$
$A \to a$.

ex. $S \to XY$ generates $aa^*bb^*$,
$X \to XX$ is in desired format.
$Y \to YY$
$X \to a$
$Y \to b$

Def$^{n}$ - If a CFG has only prod$^{ns}$ of form,
Nonterm → str. of exactly 2 Nonterm.
or " → 1 term.
it is said to be in CNF.

Th$^{m}$ - For any context-free-lang. L, the non-$\Lambda$ words of L can be generated by a grammar in which all prod$^{ns}$ are in CNF.

ex. convert $S \to aSa \mid bSb \mid a \mid b \mid aa \mid bb$ [non null palindrome] into CNF.

→ ① Separate term from non-term

$S \to ASA$ $S \to AA$ $S \to a$ $A \to a$
$S \to BSB$ $S \to BB$ $S \to b$ $B \to b$

4

Page No.:
Date: / /

we do not produce needless unit prod^ns $S \to A$ & $S \to B$.

② Introduce R's:

$S \to AR_1$ | $S \to AA$ | $A \to a$
$R_1 \to SA$ | $S \to BB$ | $B \to b$
$S \to BR_2$ | $S \to a$
$R_2 \to SB$ | $S \to b$

— ∞

if we include $S \to \Lambda$, it is palindrome.

Ex. Convert CFG into CNF.

$S \to bA \mid aB$
$A \to bAA \mid aS \mid a$
$B \to aBB \mid bS \mid b$

} lang. equal. OR $S \to aSb \mid bSa \mid ab \mid ba \mid SS$.

Let X & Y be new non-terminal for a & b resp.

∴ CFG: $S \to YA$, $S \to XB$, ★ $A \to YAA$, $A \to XS$, $A \to a$, $B \to XBB$ ★, $B \to YS$, $B \to b$.

$A \to YR_1$, $R_1 \to AA$, $B \to XR_2$, $R_2 \to BB$.

∴ CFG: ~~$S \to YA$~~, $S \to XB$, ~~$A \to YR_1$~~, ~~$R_1 \to AA$~~

$S \to YA \mid XB$
$A \to YR1 \mid XS \mid a$
$B \to XR_2 \mid YS \mid b$.
$X \to a$
$Y \to b$
$R_1 \to AA$
$R_2 \to BB$.

ex. $S \to aaaaS \mid aaaa$

lang. $a^{4n}$, $n = 1,2,3,\ldots = \{a^4\ a^8 \ldots\}$.

→ $S \to AAAAS \mid a$

$S \to AAAA$

$A \to a$

$S \to AR_1$

$R_1 \to AR_2$

$R_2 \to AR_3$

$R_3 \to AS$

$S \to AR_4$

$R_4 \to AR_5$

$R_5 \to AA$

$A \to a$.

OR $S \to SS \mid aaaa$

CNF →

$S \to SS \mid XX$

$X \to AA$

$A \to a$

Leftmost derivations –

Standardize the form of derivations.

Def^n Leftmost nonterminal in a working str. is the 1st non-term. that we encounter when we scan the str. from left to right.

ex. $abNbaXYa$ – leftmost nonterm. is N.

Def^n – If a word is generated by a CFG by a certain derivation & at each step in the deri.^n a rule of prod.^n is applied to the leftmost non-term. in the working str., then it is called leftmost derivation.

ex. CFG: $S \to aSX \mid b$

$X \to Xb \mid a$.

foll. is leftmost derivation :

$S \Rightarrow aSX$

$\Rightarrow aaSXX$

$\Rightarrow aabXX$

$\Rightarrow aabXbX$

$\Rightarrow aababX$

$\Rightarrow aababa$

→ canonical deriv^n.

Rightmost der^n –

$S \Rightarrow aSX$

$\Rightarrow aSa$

$\Rightarrow aaSXa$

$\Rightarrow aaSXba$

$\Rightarrow aaSaba$

$\Rightarrow aababa$.

Page No.:
Date: / /

ex. CFG: $S \to XY$
$X \to XX \mid a$
$Y \to YY \mid b$.

Generate the word aaabb using diff leftmost derivations.

| 1 | 2 |
|---|---|
| $S \Rightarrow XY$ | $S \Rightarrow XY$ |
| $\Rightarrow XXY$ | $\Rightarrow XXY$ |
| $\Rightarrow aXY$ | $\Rightarrow XXXY$ |
| $\Rightarrow aXXY$ | $\Rightarrow aXXY$ |
| $\Rightarrow aaXY$ | $\Rightarrow aaXY$ |
| $\Rightarrow aaaY$ | $\Rightarrow aaaY$ |
| $\Rightarrow aaaYY$ | $\Rightarrow aaaYY$ |
| $\Rightarrow aaabY$ | $\Rightarrow aaabY$ |
| $\Rightarrow aaabb$. | $\Rightarrow aaabb$. |

Thm - Any word that can be generated by a given CFG by some derivation also has a leftmost derivation.

~~Rightmost Deriv$^n$ - (Canonical Deriv$^n$).~~

Removal of Useless Symbols -

ex. $G = (\{S, A\}, \{1, 0\}, P, S)$.
$P$: $S \to 10 \mid 0S1 \mid 100 \mid A \mid SS$.

A - useless remove $S \to A$.

ex. $G = (\{S, A, B\}, \{a\}, P, S)$
$P$: $S \to AB \mid a$
$A \to a$.
B. is useless.
delete $S \to AB$
$\therefore P$: $S \to a$
$A \to a$
A useless remove $A \to a$
$\therefore P$: $S \to a$.

## Greibach Normal Form (GNF)

Every CFL without $\epsilon$ can be generated by gram. where every prod$^n$. is of form $A \to a\alpha$

$\alpha$ - zero or more non-terminals.

ex. G: $S \to ABA \mid AB \mid BA \mid AA \mid A \mid B$.
$A \to aA \mid a$
$B \to bB \mid b$.

Replace leading A by $A \to aA$, $A \to a$ & B by $B \to bB$, $B \to b$.

GNF:

$S \to aABA \mid aBA \mid aAB \mid aB \mid bBA \mid bA \mid aAA \mid aA \mid aA \mid a \mid bB \mid b$.
$A \to aA \mid a$
$B \to bB \mid b$.

## Chomsky Hierarchy -

Depending on restrictions on production, 4 classes:

Type 0 (unrestricted Grammar)
" 1 (context sensitive " )
" 2 (context free " )
" 3 (regular " )

1) Type 0 - no restriction.

$\alpha \to \beta$, $\alpha \neq \epsilon$, $\alpha, \beta \in (V \cup T)^*$

$\alpha$ & $\beta$ can be any combinations of any no. of terminals & non-term.

ex.
P: $A \to AB$
$AB \to BC$
$B \to a$.

2) Context sensitive lang. (CSG)

prod$^n$ - $xAy \to xvy$.

A - non-term, v - term / non-term, $x, v, y \in (V \cup T)^*$, x, y can be null

~~$A \to y$~~ $A \to v$, x & y provide the context in which rule may be applied.

LHS & RHS of rule may be surrounded by content of T & V.

lang $\{a^n b^n c^n d^n : n \geq 1\}$.
even more letters

$S \to abc \mid aAbC$,
$Ab \to bA$
$AC \to BbCC$
$bB \to Bb$
$aB \to aa \mid aaA$.

$CB \to CZ$
$CZ \to WZ$
$WZ \to WC$
$WC \to BC$
$aB \to ab$.
$bB \to bb$
$bC \to bC$
$cC \to cc$.

3. Context-Free Gram.
$A \to \alpha$
$A \in V$, $\alpha \in (V \cup T)^*$
LHS is one V.

4. Regular Gramar-
1. LHS of each prod$^n$ — only one non-term.
2. RHS — at most 1 non-term. — left most
left-linear,
right most - right linear gramma.

Left lin. → ~~$A \to BW$~~ $A \to Bw$ if $A \to wB$ : right lin.
$A \to \epsilon$
$A \to w$

w - str. of term., A, B - ~~term~~ non-term.

Left lin → $G = (\{S, B, C\}, \{a, b\}, P, S)$
P: $S \to Ca \mid Bb$
$C \to Bb$
$B \to Balb$.

Rt. lin → $G = (\{S, A\}, \{0, 1, \epsilon\}, P, S)$
P: $S \to 0A$
$A \to 0A \mid \epsilon$.

Reg. Gram. generates reg. lang.

$(V \cup T)^*$
$\epsilon$ null

## Equivalence of Right-linear & Left-Linear Grammars –

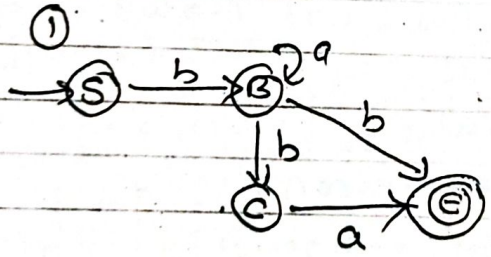
For every right-lin. G., there exists an equt. left-lin. Gr.
Lang. is regular if it is either right-lin or left-lin.

Steps to convert –
i) Represent given grammar (left/Rt.) by a TG with vertices labelled by symbols from $(V \cup \{\epsilon\})$ & transition " " $(T \cup \{\epsilon\})$.

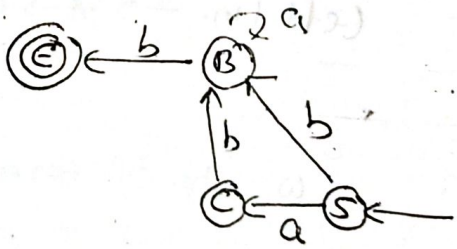
ii) Interchange pos$^{n}$ of initial & final states.
iii) Reverse dir$^{n}$ of all transitions keeping pos$^{n}$ of all intermediate states unchanged.
iv) Rewrite the gram. from this TG, in the reqd. fashion.

ex. Convert to left-lin. Gram.
$S \to bB,\ B \to bC,\ B \to aB,\ C \to a,\ B \to b.$

(1)

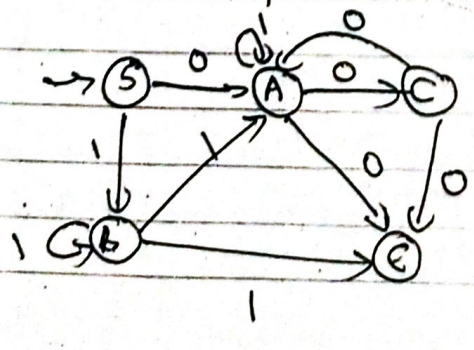


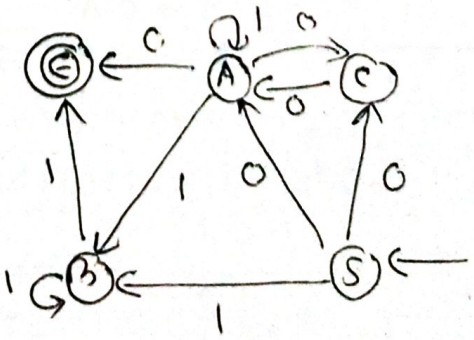

(2)




(3) $S \to Ca \mid Bb$
$C \to Bb$
$B \to Ba \mid b$.

ex. $S \to 0A \mid 1B,\ A \to 0C \mid 1A \mid 0,\ B \to 1B \mid 1A \mid 1,$
$C \to 0 \mid 0A$.

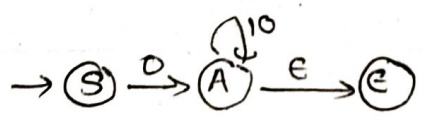


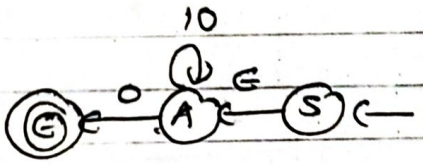



Page No.:
Date: / /

$S \to B1 / A0 / C0$
$B \to B1 | 1$
$A \to A1 | B1 | C0 | 0$
$C \to A0$

ex. $S \to 0A$
$A \to 0A | \epsilon$

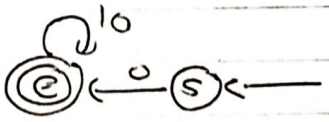



$S \to A\epsilon$, $A \to A10 | 0$

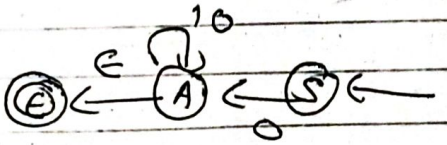
i.e. $S \to A$
$A \to A10 | 0$

ex. Convert to right lin. G.
$S \to S10 | 0$

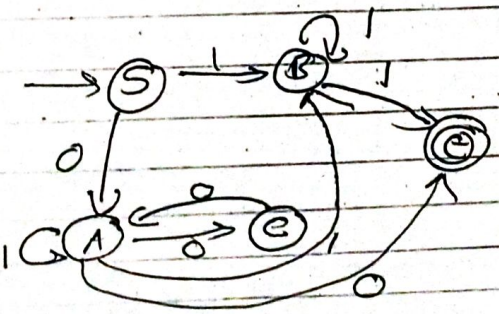

$S \to 0\epsilon$
$\epsilon \to 10\epsilon$
not allowed

∴ modify given G,
$S \to A$
$A \to A10 | 0$


$S \to 0A$
$A \to 10A | \epsilon$

ex. $S \to B1 | A0 | C0$
$A \to C0 | A1 | B1 | 0$
$B \to B1 | 1$
$C \to A0$

$S \rightarrow 0A \mid 1B$

$A \rightarrow 1A \mid 0C \mid 0$

$B \rightarrow 1B \mid 1A \mid 1$

$C \rightarrow 0 \mid 0A$.