Chomsky hierarchy of Languages

Formal Grammar – It is a notation used to specify a language and is made of 4 components – non terminals or variables, terminals, production rules and a start symbol.

∴ A formal grammar is a 4-tuple (V, T, P, S) where

V – finite set of symbols called as non-terminals or variables

T – finite set of symbols called as terminals

P – finite set of rules called production rules

S – is a member of V called start symbol

There are 4 types of grammars, differing in their power. Chomsky classified the grammars to 4 types in terms of productions as

a) type 0 or unrestricted or phase structure grammar
b) type 1 or context-sensitive grammar
c) type 2 or context-free grammar
d) type 3 or regular grammar.

1. Type 0 grammar
   1. Also called phase structure grammar which every production rule is of the form $\alpha \to \beta$ where α & β are in $(V \cup T)^*$ and $\alpha \neq \varepsilon$
   2. Since there is no restriction on what can appear on the left hand side of production rule as well as what

can appear on the right side of production rule, it is capable of specifying the largest no of languages

3) The language specifiable using type 0 grammars are called as recursively enumerable languages

4 These languages are accepted by Turing machines (TM)

5 In this type of grammars, there are no restrictions for productions

eg $A \to BC$
$AB \to C$
$ABC \to CA$

(b) Type 1 grammar

1 Also known as context sensitive in which every production rule is of the form $\alpha \to \beta$ where α & β in $(V \cup T)^+$ and $\alpha \neq \varepsilon$ & $|\alpha| \leq |\beta|$ ie length of the string that can appear on left side of production is less than or equal to the length of the string that can appear on right-hand side of production

2. The no of languages it can specify is < than that of type 0 grammar

2) The language specifiable using type 1 grammar are called as context sensitive languages

3) CSLs are accepted by Linear Bounded Automata (LBA)

eg ~~$V = \{S, A, B\}$~~ $V = \{S, A, B\}$ ~~$T = \{a, b, c\}$~~ $T = \{a, b, c\}$

~~P~~ : ~~$S \to SB$~~ $SB \to SC$ $C \to ab$

$S \to abc \mid aAbc$
$Ab \to bA$
$Ac \to Bbcc$
$bB \to Bb$
$aB \to aa \mid aaA$

(c) Type 2 grammar.

1. Also known as context free grammar (CFG) in which every production is of the form $A \Rightarrow \alpha$ where A is a single non-terminal and $\alpha \in (V \cup T)^*$.
2. The no. of languages it can specify is < than that of type 1 grammar.
3. Language accepted by CFGs are context free languages (CFLs)
4. CFLs are accepted by Push Down Automata (PDA)

$E \rightarrow E+E \mid E*E \mid E \mid id$.

(d) Type 3 grammar.

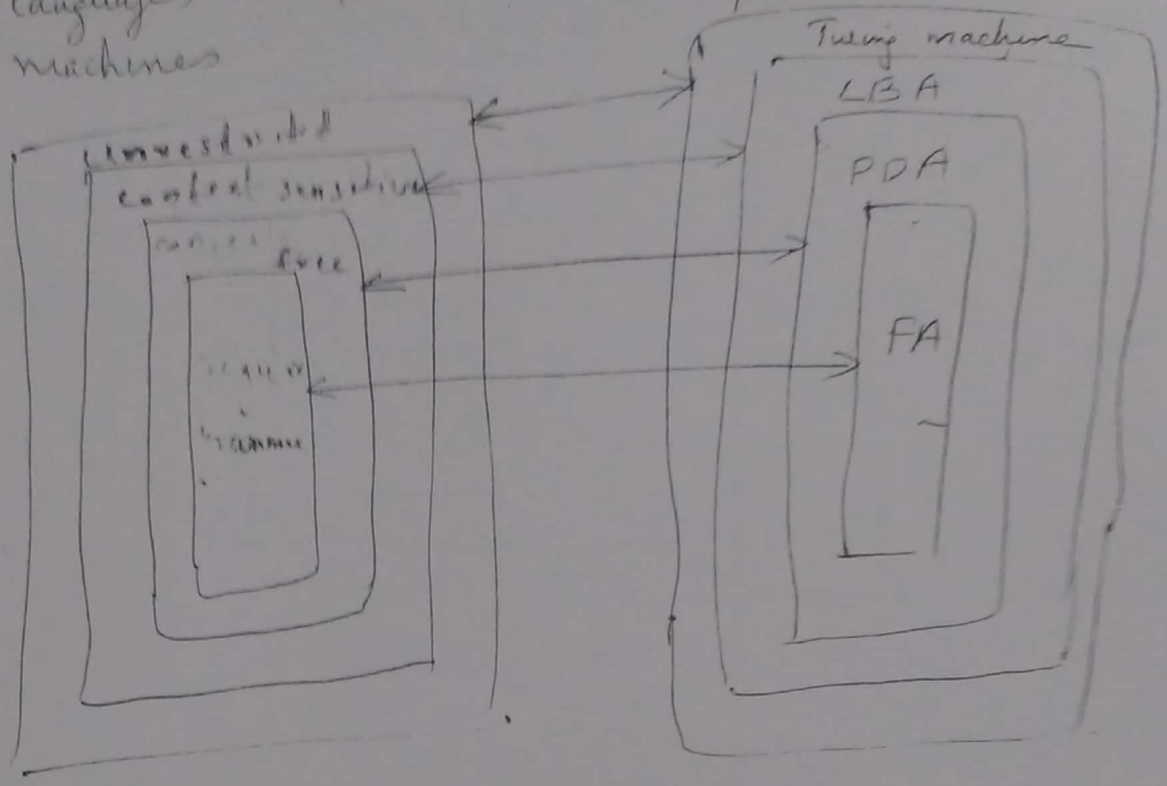
1. Also called regular grammars in which every production is of form $A \rightarrow aB$ or $A \rightarrow a$ where A & B in V and a in T.
2. The no. of languages it can specify is < than that of type 2 grammar.
3. Regular grammars are of 2 types - left linear or right linear.

If prods are of form $A \rightarrow a$ or $A \rightarrow aB$ it is right linear.

If prods are of form $A \rightarrow Ba$ or $A \rightarrow a$ it is left linear.

4. The language accepted by this grammar is called as regular sets or regular languages
5. Regular sets are accepted by Finite Automata (FA)

The following picture shows hierarchy of languages and the relationship with the machines



Relationship between the four classes of languages

Every regular language is a CFL but vice versa is not true. Similarly every CFL not containing ε is a CSL but vice versa not true. And every CSL is a recursively enumerable language but vice versa is not true. ∴ we conclude that type (i) languages properly include the type (i+1) languages except for the case of empty string ε for $i = 0, 1, 2$.

∴ These 4 classes of languages constitute a hierarchy called Chomsky hierarchy

For eg $\{a^n b^n \mid n \geq 0\}$ is proved to be non-regular but has a CFG
$\{a^n b^n c^n \mid n \geq 0\}$ is proved to be non context free but has a CSG

which says that-

1. The regular sets are properly contained in CFLs
2. The CFLs not containing empty string ε are properly contained in CSLs.
3. The CSLs are properly contained in recursively enumerable languages.

These classes form a hierarchy because each is superclass of next (higher number) class.

Type 0 ⊃ Type 1 ⊃ Type 2 ⊃ Type 3

Construction of DFA from a regular grammar G

Given a regular grammar G, a FA accepting L(G) can be obtained as follows:

(1) The no. of states of the automata is equal to no. of non-terminals of the grammar plus one i.e. there will be a state corresponding to every non-terminal of the grammar plus one more state which be the final state of FA. If L(G) contains ε make start-state as final state.

(2) The transitions in the automata is obtained as follows:

For every production $A \to aB$ do
make $\delta(A, a) = B$

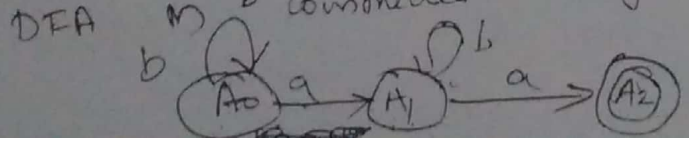
For every prod of form $A \to a$ do
make $\delta(A, a)$ = final state

Let $G = \{(A_0, A_1), \{a, b\}, P, A_0\}$ with

P. $A_0 \to aA_1$ $A_1 \to bA_1$ $A_1 \to a$

$A_0 \to bA_0$.

DFA M constructed as follows



Context free Grammar

A CFG is a 4-tuple denoted by

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

1) V - finite set of symbols called variables or non-terminals
2) T - set of symbols called terminals
3) P - set of productions of form $A \to \alpha$ $\alpha \in (VUT)^*$
4) S is a member of V called as start symbol.

The language generated by a CFG is called context free language (CFL).

eg $G = (\{S\}, \{a, b\}, P, S)$ where

P:

$S \to aSa$

$S \to bSB \mid \varepsilon$.

Derivation

Derivation refers to the replacement of an instance of a non-terminal in a given string by the right-hand side of the production rule, whose left hand side is the non-terminal to be replaced. Derivation produces a new string from the given string and consists of terminals only then derivation not possible

i.e.

For eg $\{a^n b^n\}$.

$\{a^n b^n c^n \mid n \geq$

eg $G = (\{S\}, \{a, b\}, P, S)$ $S \to aSa$
where $P: \{S \to aSa / bSb / \varepsilon\}$ $\to aa$

To derive a string aa $S \to aSa \to aa$ $(S \to \varepsilon)$

A string of terminals and variables (nonterminals), α is called a sentential form if $S \overset{*}{\Rightarrow} \alpha$.

eg $G = (\{E\}, \{+, *, (,)\}, P, E)$

where $P: E \to E+E / E*E / (E) / a$.

The language generated by a CFG G is
$L(G) = \{w \mid w \text{ is in } T^* \text{ and } S \overset{*}{\to} w\}$.

Complete models describe programmable languages by a notation called BNF (Backus-Naur form) which is the CFG notation with minor changes in format

Standard notations

1. Capital letters towards the start of the English alphabet are used to denote non-terminals (A, B, C etc)
2. Lower case letters towards the start of alphabet are used to denote terminals eg a, b, c etc digits are used to represent terminals.
3. S is used to denote start symbol.
4. Lower case letters towards the end of the alphabet of English language like u, v, w etc are used to denote strings of terminals
5. Letters α, β, γ etc are used to denote strings of terminals and non-terminals
6. Capital letters like X, Y, Z used to denote terminals or non-terminals.

Derivation Trees (Parse trees)

While deriving a string w from S, if every derivation is considered to be a step in the tree construction, then we get a graphical display of a string w called derivation tree or parse tree of a string w

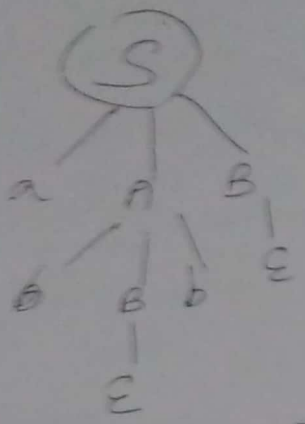
ie. The derivation in a CFG can be represented using trees which are representing derivations are called derivation trees

A parse tree for a CFG $G = (V, T, P, S)$ is a tree which satisfies the following:

1) Every vertex has a label which is a variable or terminal symbol or ε
2) The root has the label S
3) The label of an internal vertex is a variable
4) If x has label A and vertices $x_1, x_2, x_3 \ldots x_k$ are the sons of vertex x, in order from left to right with labels $x_1 x_2 \ldots x_k$ must be a production
5) A vertex x is a leaf if its label is ε, x is the only son of its father.

eg A CFG $G = (\{S, A, B\}, \{a, b\}, P, S)$ where $P: S \to aAB \quad A \to bBb \quad B \to A/\varepsilon$

Draw the parse tree for abb



$S \to aAB$
$\to abBbB$
$\to abbB$
$\to abb$

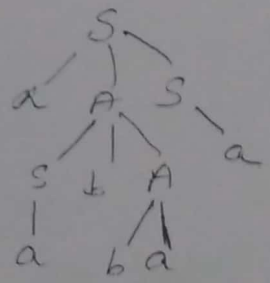
Leftmost and Rightmost derivation

In each step in derivation, a production is applied to the leftmost variable, then the derivation is a leftmost derivation. If the rightmost variable is replaced at each step it is rightmost derivation

Eg: Consider a CFG $G = (\{S, A\}, \{a, b\}, P, S)$ where P: $S \to aAS$
$A \to SbA$
$S \to a$
$A \to ba$,

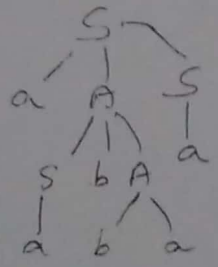
Leftmost derivation for string aabbaa

$S \to aAS$
$\to aSbAS$
$\to aabAS$
$\to aabbaS$
$\to aabbaa$



Rightmost derivation

$S \to aAS$
$\to aAa$
$\to aSbAa$
$\to aSbbaa$
$\to aabbaa$



Ambiguity in CFGs

A CFGs G such that some word has a parse trees is said to be ambiguous if there exists 2 or more leftmost or right-most derivations or parse trees to derive a string from a CFGs then CFGs is ambiguous. A The grammar that generates 2 or more parse tree is called as ambiguous grammar

A CFL for which every CFGs is ambiguous is said to be inherently ambiguous CFL

eg Consider a CFGs with productions

$S \to aSbS / bSaS / S$

To derive a string abab

$S \to aSbS$

$\to abSaSbS \to abaSbS$

$\to ababS \to ababab$

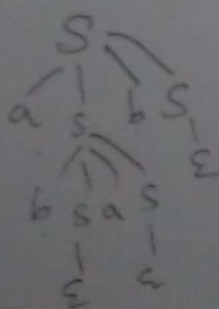
or

$S \to \cancel{bSaS} \Rightarrow aSbS$

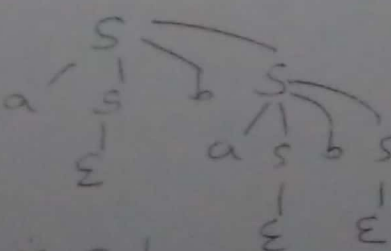
$\to abS \to abaSbS$

$\to ababS$

$\to ababab$



or



The given CFGs is ambiguous as there are 2 parse trees for the CFGs

Design of CFGs

1. Design a CFG to specify all strings over $\{a, b\}$ that are palindromes.

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

$V = \{S\}$ $T = \{a, b\}$

$P: S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$

2. Write a CFG which generates strings having equal no. of a's & b's.

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

$V = \{S\}$ $T = \{a, b\}$

$P: S \to aSbS \mid bSaS \mid \varepsilon$

or $S \to SS$
$S \to aSb$
$S \to bSa$
$S \to \varepsilon$.

3. Write a CFG to generate the language $L = \{a^m b^n \mid m \neq n\}$.

$G = (V, T, P, S)$ where $V = \{S, A, B\}$
$T = \{a, b\}$

$P: S \to aSb \mid aA \mid bB$
$A \to aA \mid \varepsilon$
$B \to bB \mid \varepsilon$

4. Write a CFG for $ww^R \mid w$ is a binary property is that extreme symbols are same
$S \to 0S0 \mid 1S1 \mid \varepsilon$

5. Write a CFG for $wcw^R \mid w$ is a binary
$S \to 0S0 \mid 1S1 \mid c$

6. Design a CFG for $L = \{a^n b^m a\}$

$S \to aSA \mid aAa$
$A \to bA \mid b$.

asA
asb.

01510

$S \to 0SA$
$\to 0S0$
$\to 00A00$

7 Write a CFG for $L = \{a^i b^i \mid i \geq 0\}$

$S \rightarrow aSb \mid \varepsilon$

8 Design a CFG for $L = \{a^i b^j c^j d^i \mid i, j \geq 0\}$

$S \rightarrow S_1 S_2$
$S_1 \rightarrow aS_1b \mid \varepsilon$
$S_2 \rightarrow cS_2d \mid \varepsilon$

9. Design a CFG for $L = \{a^p b^q c^m \mid q = p + m\}$

$S \rightarrow AB$
$A \rightarrow aAb \mid \varepsilon$
$B \rightarrow bBc \mid \varepsilon$

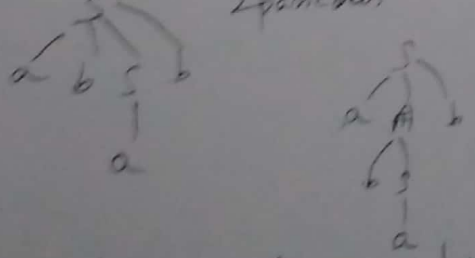
10 a. Design a CFG to generate
$L = \{a^i b^j c^k \mid i + j = k, i \geq 0, j \geq 0\}$

$S \rightarrow aSc \mid S_1$
$S_1 \rightarrow bS_1c \mid \varepsilon$

b Design a CFG for even palindromes
$S \rightarrow aSa \mid bSb \mid \varepsilon$

S.T that the grammar $S \rightarrow a \mid abSb \mid aAb$
$A \rightarrow bS \mid aAAb$ is ambiguous

2 parse trees



so ambiguous

9. $S \rightarrow aB \mid bA$ $A \rightarrow aS \mid bAA \mid a$ $B \rightarrow bS \mid aBB \mid b$
for str aaabbabbba find left most & right most derivation & draw the parse tree

Simplification of CFGs

To simplify CFGs
1. Eliminate useless variables, those variables that do not appear in the derivation of a terminal string from start symbol.
2. Eliminate ε-productions of form $A \to \varepsilon$
3. Eliminate unit prods of form $A \to B$

Eliminating useless ~~~~ Symbols

Let $G = (V, T, P, S)$ be a CFG. A variable $A \in V$ is said to be useful if and only if there is atleast one $w \in L(G)$ such that $S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w$ with x, y in $(V \cup T)^*$.

i) A variable is useful if and only if it occurs in atleast one derivation

Otherwise it is useless. A variable is also useless if there is no way of getting string from it.

A live variable in V defined as
1. If there is a production $A \to x$, $x \in T^*$, then A is live
2. If there is a prod $A \to \alpha$ and every variable in α is live, then A is live.
3. Only var A satisfying rules 1 & 2 are live.

Otherwise the variable is useless

A reachable variable defined as
1. If A appears in a string α for which there's prod $S \to \alpha$, then A is reachable.
2. If B is reachable and A appears in a string α for which there's prod $B \to \alpha$ then A is reachable
3. Only var satisfying rule 1 & 2 are reachable and useful else non-reachable & useless.

Eliminate useless symbols

Q. A CFG with prod $S \to AB/a$
$A \to b$

Here $S \to a$
$A \to b$ but B does not derive a string
But B is not a live variable So Eliminate B

i.e. $S \to a$
$A \to b$

Again A is non reachable as $S \to a$
∴ Reduced grammar is $S \to a$

(2) $S \to aAa$ $A \to Sb/bCC/DaA$
$C \to abb/DD$ $E \to aC$ $D \to aDA$

Here D is not live as a string cannot be derived from D So Eliminate D

i.e. $S \to aAa$
$A \to Sb/bCC$
$C \to abb$
$E \to aC$

To derive a string a S, E not included
So eliminate E

So reduced grammar is $S \to aAa$
$A \to Sb/bCC$
$C \to abb$

(3) $S \to aB/bX$
$A \to BAd/bSX/a$
$B \to aSB/bBX$
$X \to SBD/aBz/ad$

Here B is useless as B is not a live variable
So eliminate B $S \to bX$
$A \to bSX/a$
$X \to ad$

Again A is not involved in deriving a string from S.
So A is non-reachable.
So reduced grammar is $S \to bX$
$X \to ad$.

ie A grammar symbol is useless if it does not satisfy any one of the following conditions
1) $A \overset{*}{\Rightarrow} w$ where w is in T^*.
2) $S \overset{*}{\Rightarrow} \alpha A \beta \overset{*}{\Rightarrow} w$, w is in $L(G)$.

ie A is useless if it does not occur in any terminals and even if it occurs a string of terminals, it does not occur in the derivation sequence of any w in $L(G)$.

(2) Eliminating ε productions and nullable nonterminals

Any production of a CFG of form $A \to \varepsilon$ is called ε-production.

Any variable A for which the derivation $A \overset{*}{\Rightarrow} \varepsilon$ is called nullable

In a given CFG $G = (V, T, P, S)$ a variable defined as
1 if $A \in V$ and $A \to \varepsilon$ is a prod in P, then A is a nullable variable
2 if $A \in V$ and for some $n \geq 1$, $A \to B_1 B_2 \ldots B_n$ is a prod and $B_1, B_2 \ldots B_n$ are nullable, then A is nullable.

Given a CFG G, set V_N of all nullable variables is found using the following steps
1. For all prods $A \to \varepsilon$ put A to V_N.
2. Repeat the following step until no further variables are added to V_N.
For all prods $B \to A_1 A_2 \ldots A_n$ where $A_1 A_2 \ldots A_n$ are in V_N put B to V_N.

Once the set VN has be formed construct P'
for that find all prods in P of form
$A \to x_1 x_2 \ldots x_m$, $m \geq 1$ where
each $x_i \in V \cup T$ for each such prod of P,
we put to P', that productions
as well as those generated by replacing
nullable variables with ε in all possible
combinations for eg if x_i and x_j are both
nullable, then there will be one production
in P' with x_i replaced by ε, one in
which x_j replaced by ε and one in
which x_i and x_j are replaced with ε
If all x_i are nullable then $A \to \varepsilon$ not put to P'.

Eliminate ε-productions

Q. $S \to ABaC$ $A \to BC$ $B \to b/\varepsilon$ $C \to D/\varepsilon$ $D \to d$

Here $B \to \varepsilon$ & $C \to \varepsilon$ so B & C are nullable
$A \to BC \to \varepsilon$ ∴ A is also nullable.
∴ Nullable variables are $\{A, B, C\}$.

∴ $S \to ABaC/ABa/BaC/AaC/aC/Ba/Aa/a$
$A \to BC/B/C$
$B \to b$
$C \to D$
$D \to d$

(2) $S \to a/xb/ayA$
$x \to y/\varepsilon$
$y \to b/x$

Here $x \to \varepsilon$ $y \to x \to \varepsilon$ Nullable variables $\{x, y\}$
$S \to a/xb/b/ayA/aa$
$x \to y$
$y \to b/x$

(3) Eliminating unit- productions

Any production of a CFG of form $A \to B$ where $A, B \in V$ is called a unit- production

eg If $A \to B$ and $B \to a$ is a prod of CFG
$A \to B$ is a unit prod. To eliminate unit- prod replace B by a

$\therefore A \to a \quad B \to a$

To eliminate unit- prods, substitution method used.

Eliminate unit productions

(1) $S \to A \quad A \to B \quad B \to C \quad C \to D \quad D \to a$.

Here S, A, B, C are all unit- variables

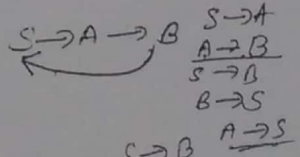
Here $S \to D$ by chain rule

Eliminating unit prods, $S \to a$
$A \to a$
$B \to a$
$C \to a$
$D \to a$.

(2) $S \to A|bb \quad A \to B|b \quad B \to bb|a$.
$B \to S|a$

Here $S \to A$, $A \to B$, $B \to S$ are unit prods. $A \to S$
$S \to B$.

Since $S \to A \quad A \to B \quad S \to A|B \quad S \to b|a$



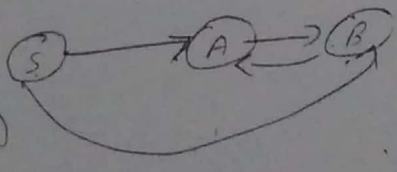
$S \to bb|a|b$
$B \to bb|a|b$
$A \to bb|a|b$

(3) $S \to Aa|B$
$B \to A|bb$
$A \to a|bc|B$

$S \to B \quad B \to A \quad S \to A$ $\quad B \to A \quad A \to a$

Here $S \to B$, $B \to A$, $A \to B$ are unit- productions
$S \to A$ as $S \to B$ & $B \to A$

If we draw a dependency graph



ie $A \to B \quad B \to A \quad S \to B/A$

Resultant grammar

$A \to a|bc|bb$

$B \to a|bc|bb$

$S \to Aa|a|bc|bb$

While reducing CFGs the only point that needs consideration is that removal of one type of productions will introduce prods of another type

So while reducing a CFG,

1. First remove ε-productions
2. Second remove unit-productions
3. Thirdly remove useless productions

Resultant grammar will be a reduced CFG with no ε, unit & useless productions.

Q Simplify the grammar

$S \to AB \quad A \to a \quad B \to C|b \quad C \to D \quad D \to E \quad E \to a$.

1. No ε-prods

$B \to C \quad C \to D \quad D \to E$

$B \to C, D \quad C \to D, E$

2. Eliminate unit-prod

$S \to AB \quad D \to a$

$A \to a \quad E \to a$

$B \to a|b$

$C \to a$

3. Eliminate useless symbols. C, D, E not reachable

$S \to AB \quad A \to a \quad B \to a|b$.

Normal forms for CFGs

While the grammar is in normal form, every prod of a grammar has a specific form. This makes it convenient to design algorithms for working with CFGs to answer certain questions

Two Normal forms

1. Chomsky Normal Form CNF
2. Greibach Normal Form GNF

1. CNF

Any context-free language without ε is generated by a grammar in which all productions are of the form $A \Rightarrow BC$ or $A \Rightarrow a$ where A, B, C are variables and a is a terminal

Before normalizing to CNF, eliminate unit & ε productions.

∴ Convert to CNF $G_1 = (\{S, A, B\}, \{a, b\}, P, S)$ with Rules P:

$S \Rightarrow bA \mid aB$

$A \Rightarrow bAA \mid aS \mid a$

$B \Rightarrow aBB \mid bS \mid b$

No unit or ε prods

$A \Rightarrow a$ & $B \Rightarrow b$ are in CNF

Now replace terminals on right-by variables ie b by C_b and a by C_a

ie

$S \rightarrow C_bA \mid C_aB$

$A \rightarrow C_bAA \mid C_aS \mid a$

$B \rightarrow C_aBB \mid C_bS \mid b$

$C_a \rightarrow a$

$C_b \rightarrow b$.

$\alpha \rightarrow \alpha_1 B \mid \alpha_2 \alpha_1$

Reduction to GNF

1. Eliminate ~~write it~~ null, unit productions and convert the grammar to GNF.
2. Rename all variables as same variable name with different subscripts (eg if S, A, B are variables then those are renamed A_1, A_2, A_3)
3. Choose a prod such that left hand side variable > right hand side starting variable subscript. Then apply lemma 1 or 2 according to the prods
4. Repeat applying lemma 1/2 for all the prods till the grammar comes into GNF.

Q. $G = (\{A_1, A_2, A_3\}, \{a, b\}, P, A_1)$ where

P: $A_1 \to A_2 A_3$

$A_2 \to A_3 A_1 \mid b$

$A_3 \to A_1 A_2 \mid a$

Step 1 Since the rt hand side of prod A_3 start with lower numbered variables begin with prod $A_3 \to A_1 A_2 \mid a$ and substitute $A_2 A_3$ for A_1

ie $A_3 \to \underline{A_2} A_3 A_2 \mid a$ Sub for $A_1 \to A_2 A_3$

we get $A_3 \to \underline{A_3 A_1} \mid A_3 A_2 \mid b A_3 A_2 \mid a$

Applying Lemma 2 to prods

$A_3 \to \underline{A_3 A_1} \mid \underline{A_3 A_2} \mid b A_3 A_2 \mid a$

we get $A_3 \to b A_3 A_2 \mid a \mid b A_3 A_2 B_3 \mid a B_3$

$B_3 \to A_1 A_3 A_2 \mid A_1 A_3 A_2 B_3$

Now prods are

$A_1 \to A_2 A_3$

$A_2 \to A_3 A_1 \mid b$

$A_3 \to b A_3 A_2 B_3 \mid a B_3 \mid b A_3 A_2 \mid a$

$B_3 \to A_1 A_3 A_2 \mid A_1 A_3 A_2 B_3$

Now A_3 in GNF sub A_3 in $A_2 \to A_3 A_1 \mid b$

we get $A_2 \to b A_3 A_2 B_3 A_1 \mid a B_3 A_1 \mid b A_3 A_2 A_1 \mid a A_1 \mid b$

sub A_2 in $A_1 \to A_2 A_3$

$A_1 \to b A_3 A_2 B_3 A_1 A_3 \mid a B_3 A_1 A_3 \mid b A_3 A_2 A_1 A_3 \mid$
$a A_1 A_3 \mid b A_3$

sub A_1 in B_3

$B_3 \to b A_3 A_2 B_3 A_1 A_3 \mid a B_3 A_1 A_3 A_3 A_2 \mid$
$b A_3 A_2 A_1 A_2 A_3 A_2 \mid a A_1 A_3 A_3 A_2 \mid b A_3 A_3 A_2 \mid$
$b A_3 A_2 B_3 A_1 A_3 A_3 A_2 B_3 \mid a B_3 A_1 A_3 A_3 A_2 B_3 \mid$
$b A_3 A_2 A_1 A_3 A_3 A_2 B_3 \mid a A_1 A_3 A_3 A_2 B_3 \mid b A_3 A_3 A_2 B_3$

Now A_1, A_2, A_3 & B_3 are in CNF

(2) $S \to AA \mid 0$

$A \to SS \mid 1$

sub A_1 for S & A_2 for A

$A_1 \to A_2 A_2 \mid 0$

$A_2 \to A_1 A_1 \mid 1$

Sub for A_1,

$A_2 \to A_2 A_2 A_1 \mid 0 A_1 \mid 1$

Applying lemma 2, we get

$A_2 \to 0 A_1 \mid 1 \mid 0 A_1 B_1 \mid 1 B_1$

$B_1 \to A_2 A_1 \mid A_2 A_1 B_1$

sub A_2 in A_1 we get

$A_1 \to 0 A_1 A_2 \mid 1 A_2 \mid 0 A_1 B_1 A_2 \mid 1 B_1 A_2 \mid 0$

$B_1 \to 0 A_1 A_1 \mid 1 A_1 \mid 0 A_1 B_1 A_1 \mid 1 B_1 A_1 \mid 0 A_1 A_1 B_1 \mid 1 A_1 B_1 \mid$
$0 A_1 B_1 A_1 B_1 \mid 1 B_1 A_1 B_1$