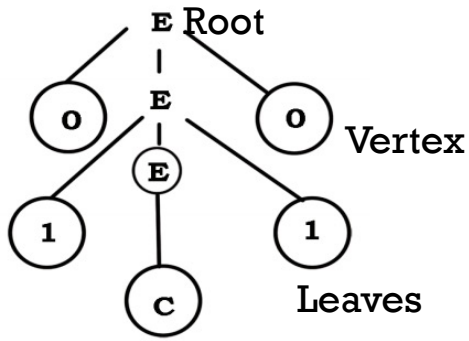
CFG AND CNF

COMP 4200 – Formal Language

PARSE OR DERIVATION TREE

- Parse tree is an ordered rooted tree that graphically presents the semantic information of string derived from CFG.
- Any one production or derivation derived from tree format.
- It is the graph form of representation.
- RULE
  - Start with 'S'.→ Root
  - The final answer should be terminal. →Leaves
  - The derivation should be applied from left to right.
  - The intermediate derivation should be terminal or nonterminal. → vertex

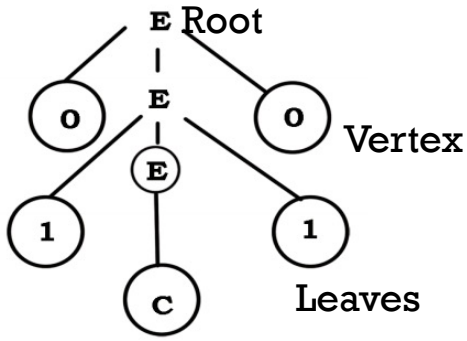


- W = 01C10 , E → 0E0 E →1E1 E → C
- DERIVATION E => 0E0

  => 01E10 ∴ E => 1E1

  => 01C10 ∴ E => C

  => E => 01C10
- PARSE TREE (OR) DERIVATION TREE



TYPES OF DERIVATION

- LEFT MOST DERIVATION (LMD): If at each step-in derivation, a production is applied to the left most variable (or) left most non-terminal then the derivation method is called left most derivation.
- Example w = id+id*id E → E+E , E → E*E, E → id

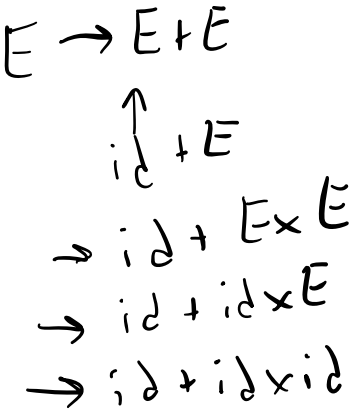
E → E+E

E → id+E ∴ E=>id

E → id+E*E ∴ E=>E*E

E → id+id*E ∴ E=>id

E → id+id* id



- RIGHT MOST DERIVATION (RMD): A derivation in which the right most variable is replaced at each step then, the derivation method is called right most derivation.
- Example

E → E+E

E → E+E*E ∴ E=>E*E

E → E+E*id ∴ E=>id

E → E+id*id => id+id*id

EXAMPLE

- Draw derivation for the string abbabba For thee CFG given G where production is S → bA/aB A → a/aS/Baa B → b/bS/Abb
  - s → aB
  - s → aBB
  - s → abSB
  - s → abSbs
  - s → abbAbs
  - s → abbAbbA
  - s → abbabbA
  - s → abbabba

EXAMPLE

Draw derivation tree for the string abaaba for the CFG given by G where p is S → aSa/bSb/b/a/ε

Solution:

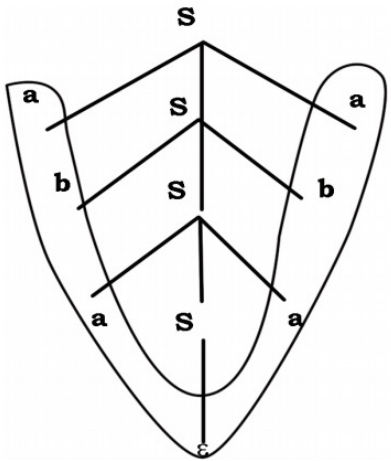
S → aSa

S → abSba

S → abaSaba

S → abaεaba

S → abaaba



TRY YOURSELF!

- Draw derivation tree for the string aabbaabba For thee CFG given G where production is S → bA/aB A → a/aS/bAA B →b/bS/aBB
- Draw the derivation tree for the string abb where S → aAB A → bBb B → A/ε.
- Draw the derivation tree for the given graph CFG G=(v,t,p,s) where p={S → aSb/ ε} and the input string :aabb

S → a AB

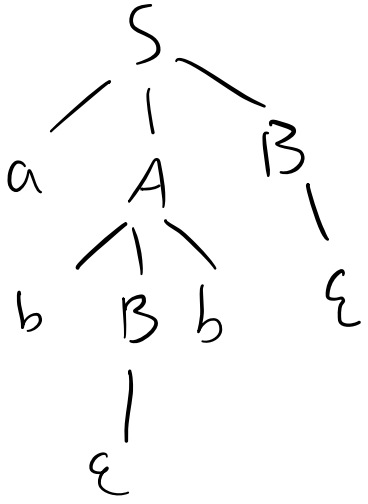
S → a bBb B

S → a b ε b B

S → a b ε b ε

S → abb

derivation/parse tree

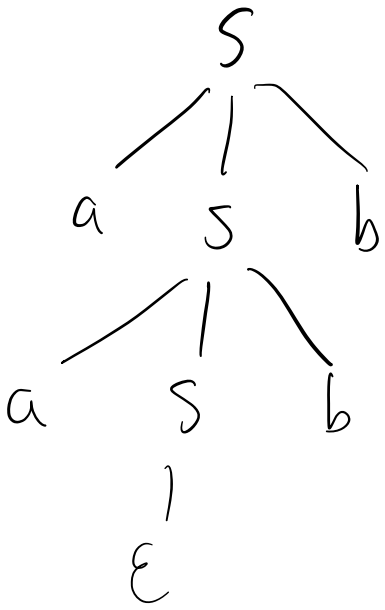


$S \to aSb$

$S \to aaSbb$

$S \to aa\varepsilon bb$

$S \to aabb$



$S \to aB$

$S \to aaBB$

$S \to aabB$

$S \to aabbS$

$S \to aabbaB$

$S \to aabbaaBB$

$S \to aabbaabB$

$S \to aabbaabbS$

$S \to aabbaabbaB$

Can't derive

EXAMPLE

- Draw derivation for the string abbabba For thee CFG given G where production is S → bA/aB A → a/aS/Baa B → b/bS/Abb/BB
  - s → aB
  - s → aBB
  - s → abSB
  - s → abSbS
  - s → abbAbS
  - s → abbAbbA
  - s → abbabbA
  - s → abbabba

TRY YOURSELF!

- Consider the grammar G=({A,S},{a,b},P,S) where p consist of S → aAS/a A → SbA/SS/ba Draw the derivation for string"aabbaa"
- Draw the derivation for the string abb where S → aAB A → bBb B → A/ε

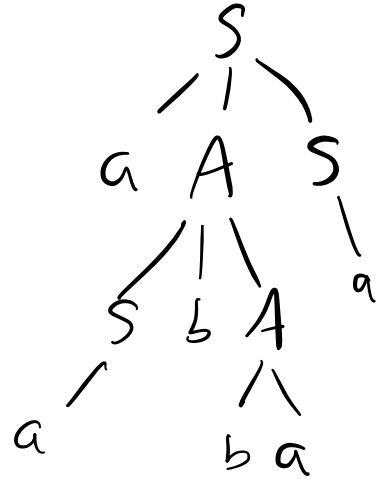
$S \rightarrow aAS$

$S \rightarrow aSbAS$

$S \rightarrow aabAS$

$S \rightarrow aabbaS$

$S \rightarrow aabbaa$



TRY YOURSELF!

- Draw derivation tree for the string aabbaabba For thee CFG given G where production is S → bA/aB A → a/aS/bAA B →b/bS/aBB
- Draw the derivation tree for the string abb where S → aAB A → bBb B → A/ε.
- Draw the derivation tree for the given graph CFG G=(v,t,p,s) where p={S → aSb/ ε} and the input string :aabb

AMBIGUITY

- CFG G is ambiguous if string w ∈ L(G) having different parse trees (or equivalently, different leftmost derivations).
- A string is derived ambiguously in a CFG if it has two or more different leftmost derivations
- A grammar that produces more than one parse tree (or) derivation tree for some string then the grammar is said to be an ambiguous grammar.
- An ambiguous grammar produces more than one LMD (or) more than 1 RMD then, the given grammar is said to be an ambiguous grammar.
- Leftmost derivation : At every step in the derivation the leftmost variable is replaced
- A grammar is ambiguous if it generates *some* string ambiguously
- Some context free languages are inherently ambiguous, that is, every grammar for the language is ambiguous

- LMD:

E => id+E ∴ E => id

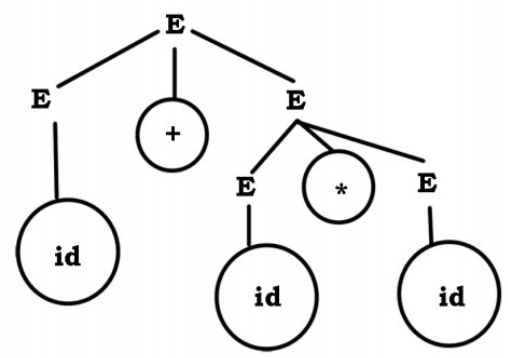
E => id+E*E

E => id+id*E

E => id+id*id

E=> id+id*id

PARSE TREE



Rules:
E → E+E ,E → E*E ,E → (E) ,E → id
w= id+id*id

- RMD:

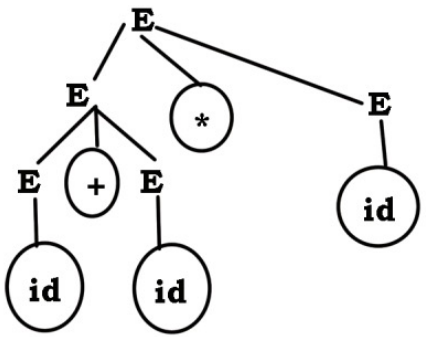
E => E*E

E => E+E*E

E => E+E*id

E => E+id*id

E => id+id*id



Therefore, the above grammar is ambiguous.

Show that CFG having productions A → a/Aa/bAA/AAb/AbA is ambiguous. The string can be baaaa

$A \rightarrow bAA$

$A \rightarrow bAaA$

$A \rightarrow bAaaA$

$A \rightarrow baaaA$

$A \rightarrow baaaa$

$A \rightarrow bAA$

$A \rightarrow baA$

$A \rightarrow baAa$

$A \rightarrow baAaa$

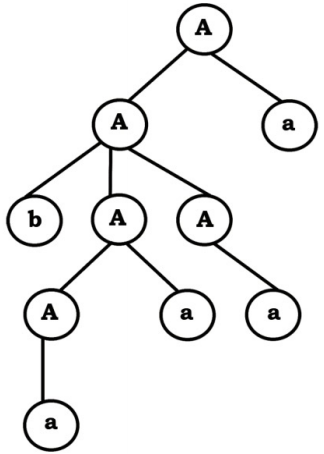
$A \rightarrow baaaa$

Yes ambiguous

Show that CFG having productions A → a/Aa/bAA/AAb/AbA is ambiguous.

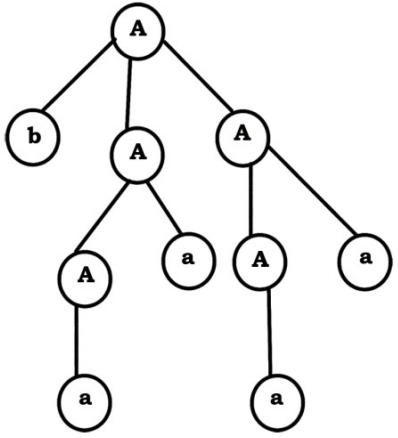
Parse tree 1 A → Aa => bAAa => bAaaa

A → baaaa



- Parse tree 2 A → bAA => bAaA => bAaAa

  A → baaaa



- Therefore, the above grammar is ambiguous.

TRY YOURSELF!

- Prove that given CFG is ambiguous, S →0B/1A A → 0/0S/1AA B → 1/1S/0BB. The string can be randomly chosen. For reference I have chosen "0011010"

- Show that grammar is S =>aSbS/ bSaS/ ε ambiguous.

$S \to 0B$

$S \to 00BB$

$S \to 001B$

$S \to 0011S$

$S \to 00110B$

$S \to 00110S$

$S \to 001101B$

$S \to 0011010B$

$S \to 0B$

$S \to 00BB$

$S \to 001SB$

$S \to 0011AB$

$S \to 00110SB$

$S \to 001101AB$

$S \to$

Not able to derive

$S \to 0B$

$S \to 00BB$

$S \to 001SB$

$S \to 0010BB$

$S \to 00101B$

$S \to 0B$

$S \to 00BB$

$S \to 001B$

$S \to 0010BB$

$S \to 00101B$

$S \to 001011$

$S \to 001011$

ambigious

RESOLVING AMBIGUITY

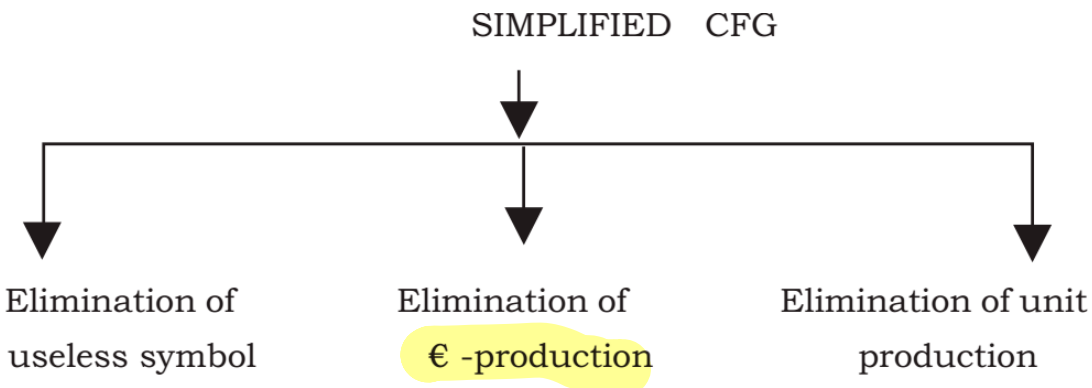
- Designing unambiguous grammars is tricky and requires planning from the start.
- It's hard to start with an ambiguous grammar and to manually convert it into an unambiguous one.
- Often, must throw the whole thing out and start over.
- We have just seen that this grammar is ambiguous:

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

  If we take a string id+id*id or id+id+id we get two parse trees.
- Goals:
  - Eliminate the ambiguity from the grammar.
  - Make the only parse trees for the grammar the ones corresponding to operator precedence.

SIMPLIFYING GRAMMAR

SIMPLIFIED CONTEXT FREE GRAMMAR



1. Eliminate useless symbols i.e.,symbols or terminals which do not appear in any derivation of a terminal string from start symbol, 2 types, nonterminating (generating), non reachable,
2. Eliminate ε-productions which are of the form $A \rightarrow \varepsilon$ form some variable A.
3. Eliminate unit production which are of the form $A \rightarrow B$ for variables A and B.

ELIMINATING USELESS PRODUCTIONS

- **Non-generating symbols** are those symbols which do not produce any terminal string.
- **Non-reachable symbols** are those symbols which cannot be reached at any time starting from the start symbol.
- Find the non-generating symbols, i.e., the symbols which do not generate any terminal string. If the start symbol is found non-generating, leave that symbol. For removing non-generating symbols, remove those productions whose right side and left side contain those symbols.
- Now find the non-reachable symbols, i.e., the symbols which cannot be reached, starting from the start symbol. Remove the non-reachable symbols.

- Remove the useless symbols from the given CFG.

All reachable

$S \rightarrow AC$
$\rightarrow aCB$
$\rightarrow aACB$
$\rightarrow a$

The C stays
C is non terminal

~~$S \rightarrow AC$,~~
$S \rightarrow BA$,
~~$C \rightarrow CB$,~~
~~$C \rightarrow AC$,~~
$A \rightarrow a$,
$B \rightarrow$ ~~a~~C/b

$S \rightarrow A\,C \mid BA$
$C \rightarrow CB \mid AC$
$A \rightarrow a$
$B \rightarrow aC \mid b$

- Those symbols which do not produce any terminal string are non-generating symbols. Here, C is a non-generating symbol. So, we have to remove the symbol C.
- Minimized grammar will be: S → BA, A → a, B → b
- Now, we have to find non-reachable symbols, the symbols which cannot be reached at any time starting from the start symbol. There is no non-reachable symbol in the grammar.
- The minimized grammar is, $S \rightarrow BA, A \rightarrow a, B \rightarrow b$

A is non reachable

B is non terminal

EXAMPLE

Remove the useless symbols from the given CFG.

1. S → aB/bX, A → Bad/bSX/a, B → aSB/bBX, X → SBD/aBx/ad
2. S → AB | CA, A→ a, B→BC|AB, C→aB|b
3. S→aA|a|Bb|cC, A→aB, B→a|Aa, C→cCD, D→ddd

S→aA|a|Bb , A→aB , B→a|Aa

S→bX X→ad S→CA , A→a , C→b

TRY YOURSELF!

- S→ aC|SB, A→ bSCa|ad, B→ aSB|bBC, C→ aBC|ad S→aC, C→ad
- S→aAa, A→ bBB, B→ab, C→ aB S→aAa, A→bBB, B→ab
- S→ aS|A|C, A→a, B→aa, C→aCb

S→aS|A , A→a

REMOVAL OF UNIT PRODUCTIONS

- Any production rule of the form $A \rightarrow B$, where A,B $\in$ Non terminal is called Unit production.
- Steps:
  - To remove A → B, add production, A→X to the grammar rule whenever B → x occurs in grammar. X → Null
  - Delete A→B
  - Repeat step 1 until all the unit productions are removed.

REMOVAL OF UNIT PRODUCTIONS

- A unit production is a production which is of form A → B, where both A and B are variables.
- Production in the form ***non-terminal*** → ***single non-terminal*** is called unit production.
- Let there be a grammar S → AB, A → E, B → C, C → D, D → b, E → a
- From here, if we are going to generate a language, then it will be generated by the following way
  - S → AB => EB => aB => aC => aD => ab => 6 steps
- The grammar, by removing unit production and as well as minimizing, will be
  - S → AB, A → a, B → b => 3 steps

TRY YOURSELF!

S → 0AB
A → 01|B
B → 0A11

S → 0AB
A → 01|0A11
B → 0A11

- S → 0AB A → 01 | B B → 0A | 1
- S→0A | 1B | C A→0S | 00 B→1 | A C→01
- Remove the unit production from the following grammar. S → AB, A → a, B → C, C → D, D → b
- Remove the unit production from the following grammar. S→ aX/Yb/Y, X→S, Y→Yb/b
- Remove the unit production from the following grammar. S→AA, A→B/BB, B→ abB/b/bb

Start with the last state

$S \to 0A \mid 1B \mid 01$

$A \to 0S \mid 00$

$B \to 1 \mid 0S \mid 00$

$S \to AB$

$A \to a$

$B \to d$

$S \to aX \mid Yb \mid b$

$X \to aX \mid Yb \mid b$

$Y \to Yb \mid b$

There is still a way to reach Y.

$S \to AA$

$A \to abB \mid b \mid bb \mid BB$

$B \to abB \mid b \mid bb$

There is still a way to reach B.

REMOVAL OF NULL PRODUCTIONS

$A \to \varepsilon \quad B \to \varepsilon$

- In CFG, a nonterminal symbol A is a nullable variable if there is a production, A → ε or there is a derivation that starts at A and leads to ε.
- Steps:
  - To remove A → ε look for all productions whose right side contains A.
  - Replace each occurrences of A in each of these productions with ε.
  - Add the resultant production to the grammar.
- Example: S → ABAC, A→aA/ε, B→bB/ε, C→c
  - Null productions: A→ ε and B→ ε

$S \to ABAC \mid BAC \mid ABC \mid BC$

$A \to aA \mid a \quad B \to bB \mid \varepsilon \quad C \to c$

- To eliminate A→ ε, look for production whose right side contains A.
  - S → ABAC, replace A→ ε
  - Possible outcomes, S→ABC/BAC/BC
  - A→aA, replace A→ ε , A→a
  - S→ABAC/ABC/BAC/BC, A→aA/a, B→bB/ε, C→c

$S \to ABAC \mid BAC \mid ABC \mid BC \mid AAC \mid AC \mid C$

$A \to aA \mid a \quad B \to bB \mid b \quad C \to c$

REMOVAL OF NULL PRODUCTIONS

- To eliminate B→ ε, look for production whose right side contains B.
  - S → ABAC, replace B→ ε
  - Possible outcomes, S→AAC/AC/C
  - B→bB, replace B→ ε , B→b
  - S→ABAC/ABC/BAC/BC/AAC/AC/C, A→aA/a, B→bB/b, C→c

TRY YOURSELF!

S → aSa|bSb|aa|bb

- S→aSa|bSb|ε
- S→aS|A, A→aA|ε S→aS|A|ε A→aA|a ; S→aS|A|a A→aA|a
- S→ a|Ab|aBa, A→ b|ε, B→ b|A S→a|Ab|aBa|b A→b B→b|A|ε
- S→AB, A→ aAA|ε, B→ bBB|ε S→a|Ab|aBa|b|aa A→b B→b|A

Start with useless

Start with epsilon

Unit production

S→a|Ab|aBa|b|aa , A→b , B→b

Fully reduced

CHOMSKY NORMAL FORM

CHOMSKY NORMAL FORM

Chomsky Normal Form. A grammar where every production is either of the form $A \rightarrow BC$ or $A \rightarrow c$ (where A, B, C are arbitrary variables and c an arbitrary symbol).

Example: $S \rightarrow AS \mid a$ $A \rightarrow SA \mid b$

(If language contains ε, then we allow $S \rightarrow \varepsilon$ where S is start symbol, and forbid S on RHS.)

WHY CHOMSKY NORMAL FORM?

- The key advantage is that in Chomsky Normal Form, every derivation of a string of n letters has exactly 2n − 1 steps.
- Thus, one can determine if a string is in the language by exhaustive search of all derivations.

CHOMSKY NORMAL FORM

- Every context free grammar/ CFL without ε can be generated by a grammar for which every production in the form of $A \rightarrow AA$ and $A \rightarrow a$ where A is variable and a is a terminal.
- CNF format:
  - Non-terminal $\rightarrow$ non-terminal non-terminal
  - non-terminal $\rightarrow$ terminal
- The conversion to Chomsky Normal Form has four main steps:
  - 1. Get rid of all ε productions.
  - 2. Get rid of all unit productions (where RHS is one variable).
  - 3. Replace every production that is too long by shorter productions.
  - 4. Move all terminals to productions where RHS is one terminal.

CAN ALWAYS PUT CFG INTO CHOMSKY NORMAL FORM

- Recall: CFG in Chomsky normal form if each rule has form:
  - $A \rightarrow BC$ or $A \rightarrow x$ or $S \rightarrow \varepsilon$ where $A \in V$; $B, C \in V - \{S\}$; $x \in \Sigma$.
- Theorem: Every CFL can be described by a CFG in Chomsky normal form.
- Proof Idea:
  - Start with CFG $G = (V, \Sigma, R, S)$.
  - Replace, one-by-one, every rule that is not "Chomsky".
  - Need to take care of: Start variable (not allowed on RHS of rules) ε-rules ($A \rightarrow \varepsilon$ not allowed when A isn't start variable) all other violating rules ($A \rightarrow B$, $A \rightarrow aBc$, $A \rightarrow BCDE$)

- A context-free grammar is in Chomsky normal form if every rule is of the form

$A \rightarrow BC$

$A \rightarrow a$

where a is any terminal and A, B, and C are any variables—except that B and C may not be the start variable.

We can also have $S \rightarrow \varepsilon$, where S is the start variable.

Theorem: Any context-free language is generated by a context-free grammar in Chomsky normal form.

- Add a new start variable S0 and the rule $S0 \rightarrow S$, where S was the original start variable. This change guarantees that the start variable doesn't occur on the right-hand side of a rule.
- Remove an ε-rule $A \rightarrow \varepsilon$, where A is not the start variable. If $R \rightarrow uAv$ is a rule in which u and v are strings of variables and terminals, we add rule $R \rightarrow uv$. We do so for each occurrence of an A, so the rule $R \rightarrow uAvAw$ causes us to add $R \rightarrow uvAw$, $R \rightarrow uAvw$, and $R \rightarrow uvw$. If we have the rule $R \rightarrow A$, we add $R \rightarrow \varepsilon$ unless we had previously removed the rule $R \rightarrow \varepsilon$.

- Remove all unit rules $A \rightarrow B$.
  - Whenever a rule $B \rightarrow u$ appears, add the rule $A \rightarrow u$
  - Repeat these steps until all unit rules are eliminated.
- Convert all remaining rules rule $A \rightarrow u_1 u_2 \cdots u_k$, where $k \geq 3$ with the rules

  $A \rightarrow u_1 A_1$,

  $A_1 \rightarrow u_2 A_2$,

  .....

  $A_{k-2} \rightarrow u_{k-1} u_k$
- The A_i's are new variables. Replace any terminal u_i in the preceding rule(s) with the new variable U_i and add the rule $U_i \rightarrow u_i$.

Convert CFG to CNF: P, S→ASA/aB, A→B/S/ε, B→b/ε

- Convert CFG to CNF: P, S→ASA/aB, A→B/S/ε, B→b/ε
- Step 1: Since S appears in RHS, we **add new state S0 and S0→S**
  - S0→S, S→ASA/aB, A→B/S/ε, B→b/ε
- Step 2: Remove Null productions:
  - A → ε, S→ASA/aB
  - S → SA/AS/S
  - B → ε, S→ASA/aB
  - S → a
  - P: S0→S, S→ASA/SA/AS/aB/a/S, A→B/S, B→b
- Step 3: Remove Unit productions:
  - S→S,S0→S,A→B,A→S
  - Removing S→S: S0→S,S→ASA/aB/a/AS/SA, A→B/S, B→b
  - Removing S0→S: S0→ ASA/aB/a/AS/SA,S→ASA/aB/a/AS/SA, A→B/S, B→b
  - Removing A→B: S0→ ASA/aB/a/AS/SA,S→ASA/aB/a/AS/SA, A→b/S, B→b
  - Removing A→S: S0→ ASA/aB/a/AS/SA,S→ASA/aB/a/AS/SA, A→b/ASA/aB/a/AS/SA, B→b

- Step 4: Find out the production that has more than two variables in RHS.
  - S0→ASA, S→ASA, A→ ASA, X→SA
  - After removing we get,
    - S0 → AX/aB/a/AS/SA
    - S → AX/aB/a/AS/SA
    - A → AX/aB/a/AS/SA/b
    - B → b
    - X → SA
- Step 5: Now change the production, S0 → aB, S→aB, A→aB, to remove 'a' make Y→a
  - S0→AX/YB/a/AS/SA
  - S→ AX/YB/a/AS/SA
  - A→b/ AX/YB/a/AS/SA
  - B→b
  - X→SA
  - Y→a

Convert the following CFG to CNF, S → ABA A → aA|ε , B → bB|ε

ANOTHER EXAMPLE

- Convert the following CFG to CNF.
  S → ABA A → aA|ε , B → bB|ε
- Step 1: Since S appears in RHS, we add new state S0 and S0→S
  - S0→S, S→ABA, A → aA|ε, B → bB|ε
- Step 2: Remove Null production.
  - A → ε, S→AB/BA/B, A→a
  - B → ε, S→AA/A, B→b
  - S0→S, S→ABA/AB/BA/AA/A/B, A→aA/a, B→bB/b
- Step 3: Remove Unit productions:
  - S0→S,S→B,S→A
  - Removing S→A, S→ABA/AB/BA/AA/a/B/aA, A→aA/a, B→bB/b
  - Removing S→B, S→, A→aA/a, B→bB/b
  - Removing S0→S, S0→ ABA/AB/BA/AA/a/b/aA/bB, A→aA/a, B→bB/b

- Step 4: Find out the production that has more than two variables in RHS.
  - S0→ABA, S→ABA, X→AB
  - After removing we get,
    - S0 → XA/AB/BA/AA/a/b/aA/bB,
    - S → XA/AB/BA/AA/a/b/aA/bB,
    - A → aA/a,
    - B → bB/b
    - X → AB
- Step 5: Now change the production, S0 → aA, S→aA, A→aA,B→bB, S0 → bB, S→bB to remove 'a' make Y→a,to remove 'b' make Z→b.
  - S0 → XA/AB/BA/AA/a/b/YA/ZB,
  - S → XA/AB/BA/AA/a/b/YA/ZB,
  - A → YA/a,
  - B → ZB/b
  - X → AB
  - Y→a
  - Z→b

EXAMPLE: CONVERT CFG INTO CHOMSKY NORMAL FORM

$S \rightarrow XSX \mid aY$

$X \rightarrow Y \mid S \mid \varepsilon$

$Y \rightarrow b \mid \varepsilon$

EXAMPLE: CONVERT CFG INTO CHOMSKY NORMAL FORM

- Initial CFG G0:

$S \rightarrow XSX \mid aY$

$X \rightarrow Y \mid S \mid \varepsilon$

$Y \rightarrow b \mid \varepsilon$

Step 1. Introduce new start variable S0 and new rule $S0 \rightarrow S$:

$S0 \rightarrow S$

$S \rightarrow XSX \mid aY$

$X \rightarrow Y \mid S \mid \varepsilon$

$Y \rightarrow b \mid \varepsilon$

- Step 2: Remove ε-rules for which left side is not start variable:
  - (i) remove $Y \rightarrow \varepsilon$
  - $S0 \rightarrow S$
  - $S \rightarrow XSX \mid aY \mid a$
  - $X \rightarrow Y \mid S \mid \varepsilon$
  - $Y \rightarrow b$
  - (ii) remove $X \rightarrow \varepsilon$
  - $S0 \rightarrow S$
  - $S \rightarrow XSX \mid aY \mid a \mid SX \mid XS \mid S$
  - $X \rightarrow Y \mid S$
  - $Y \rightarrow b$
- Step 3: Remove unit rules:
  - (i) remove unit rule, $S \rightarrow S$
    - $S0 \rightarrow S, S \rightarrow XSX \mid aY \mid a \mid SX \mid XS, X \rightarrow Y \mid S, Y \rightarrow b$
  - (ii) remove unit rule $S0 \rightarrow S$
    - $S0 \rightarrow XSX \mid aY \mid a \mid SX \mid XS, S \rightarrow XSX \mid aY \mid a \mid SX \mid XS, X \rightarrow Y \mid S, Y \rightarrow b$
  - (iii) remove unit rule $X \rightarrow Y$
    - $S0 \rightarrow XSX \mid aY \mid a \mid SX \mid XS, S \rightarrow XSX \mid aY \mid a \mid SX \mid XS, X \rightarrow S \mid b, Y \rightarrow b$
  - (iv) remove unit rule $X \rightarrow S$
    - $S0 \rightarrow XSX \mid aY \mid a \mid SX \mid XS, S \rightarrow XSX \mid aY \mid a \mid SX \mid XS, X \rightarrow b \mid XSX \mid aY \mid a \mid SX \mid XS, Y \rightarrow b$

- Step 4: Replace problematic terminals a by variable U with $U \rightarrow a$.
  - $S0 \rightarrow XSX \mid UY \mid a \mid SX \mid XS$
  - $S \rightarrow XSX \mid UY \mid a \mid SX \mid XS$
  - $X \rightarrow b \mid XSX \mid UY \mid a \mid SX \mid XS$
  - $Y \rightarrow b$
  - $U \rightarrow a$
- Shorten long RHS to sequence of RHS's with only 2 variables each
  - $S0 \rightarrow XX_1 \mid UY \mid a \mid SX \mid XS$
  - $S \rightarrow XX_1 \mid UY \mid a \mid SX \mid XS$
  - $X \rightarrow b \mid XX_1 \mid UY \mid a \mid SX \mid XS$
  - $Y \rightarrow b$
  - $U \rightarrow a$
  - $X_1 \rightarrow SX$ which is a CFG in Chomsky normal form.

TRY YOURSELF!

- Convert the following CFG into CNF.

  S → aAD

  A→aB | bAB

  B → b

  D → d

- Convert the following CFG into CNF.

  S → 0A0 | 1B1 | BB

  A → C

  B → S | A

  C → S | ε

- Convert the given CFG to CNF,

  S → AbA

  A → Aa | ε

S → AaA

A → aaBa | CDA | CD

B → bB

C → Ca | D

D → bD | ε

SUMMARY

There are special forms for CFGs such as Chomsky Normal Form, where every production has the form $A \rightarrow BC$ or $A \rightarrow c$. The algorithm to convert to this form involves

(1) determining all nullable variables and getting rid of all ε-productions,

(2) getting rid of all variable unit productions,

(3) breaking up long productions, and

(4) moving terminals to unit productions.