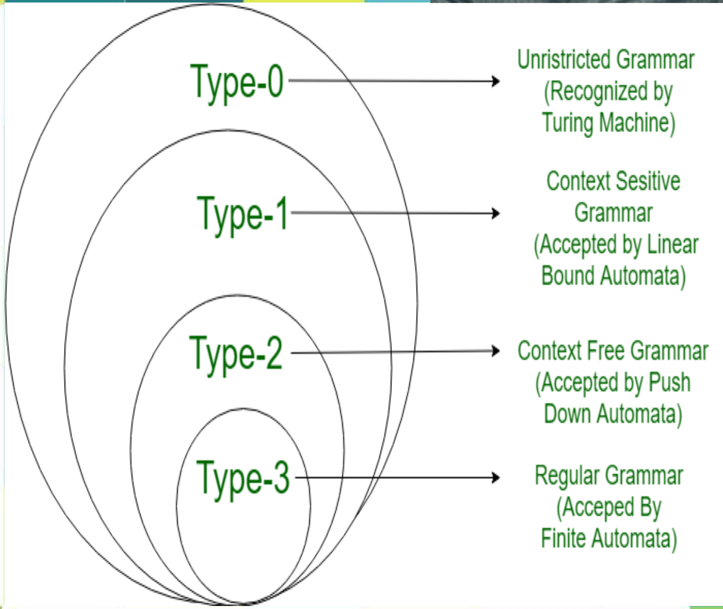
||Shri Hari||

BCSC0011: THEORY OF AUTOMATA & FORMAL LANGUAGES



(Module-II)

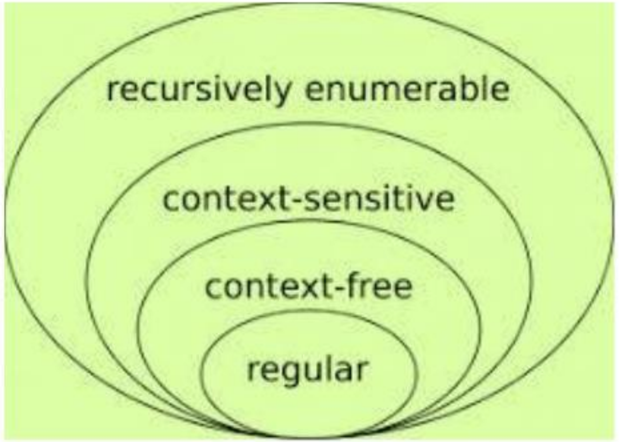
By: Dr. Sandeep Rathor

Module II

- Context Free Grammar and Context Free Languages: Introduction, Derivation Trees or Parse Tree
- Ambiguity in Grammar, Ambiguous to Unambiguous CFG
- **Simplification of CFGs**: Removal of useless symbols, elimination of null productions, elimination of unit productions
- **Normal Forms** for CFGs - CNF and GNF
- Pumping lemma for CFLs
- Equivalence of PDA and CFG
- **Turing Machine**

Context-Free Grammars

- Languages that are **generated** by context-free grammars are context-free languages
- Context-free grammars are more expressive than finite automata: if a language L is **accepted** by a finite automata then L can be **generated** by a context-free grammar
- Beware:
  The converse is NOT true



A Grammar is the collection of 4-tuple i.e.

$G = (V_N, \Sigma, P, S)$ where:

V_N: Set of Variables; **Σ:** Set of Terminals

P: Production Rule; **S:** Start Symbol

V_N ={<Sentence>,<noun>,<verb>,<adverb>}

Σ={sandeep, hari, read, quickly}

S=<Sentence>

<Sentence> <noun><verb><adverb>

<noun> → Sandeep, hari

<verb> → read

<adverb> → quickly

Validation Rules:

1. If S->BC is a given production, it means we can replace S by BC but can not replace BC by S. [Reverse substitution is not possible]
2. If S->BC is a given production, it is not necessary that BC->S is a production. [No inverse operation is permitted]

Formal Definition
Grammar

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

Finite Set of Variables / Non Terminals

Finite Set of terminal symbols

Set of Productions rules

Start variable

All productions in **P** are of the form

$$A \rightarrow \alpha$$

Variable

String of variables and terminals

Example of Context-Free Grammar

$$\boxed{S \to aSb \mid \lambda}$$

$$P = \{S \to aSb,\ S \to \lambda\}$$

productions

$$\mathbf{G} = (\mathbf{V}_N, \Sigma, \mathbf{S}, \mathbf{P})$$

$$\mathbf{V}_N = \{S\}$$

variables

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

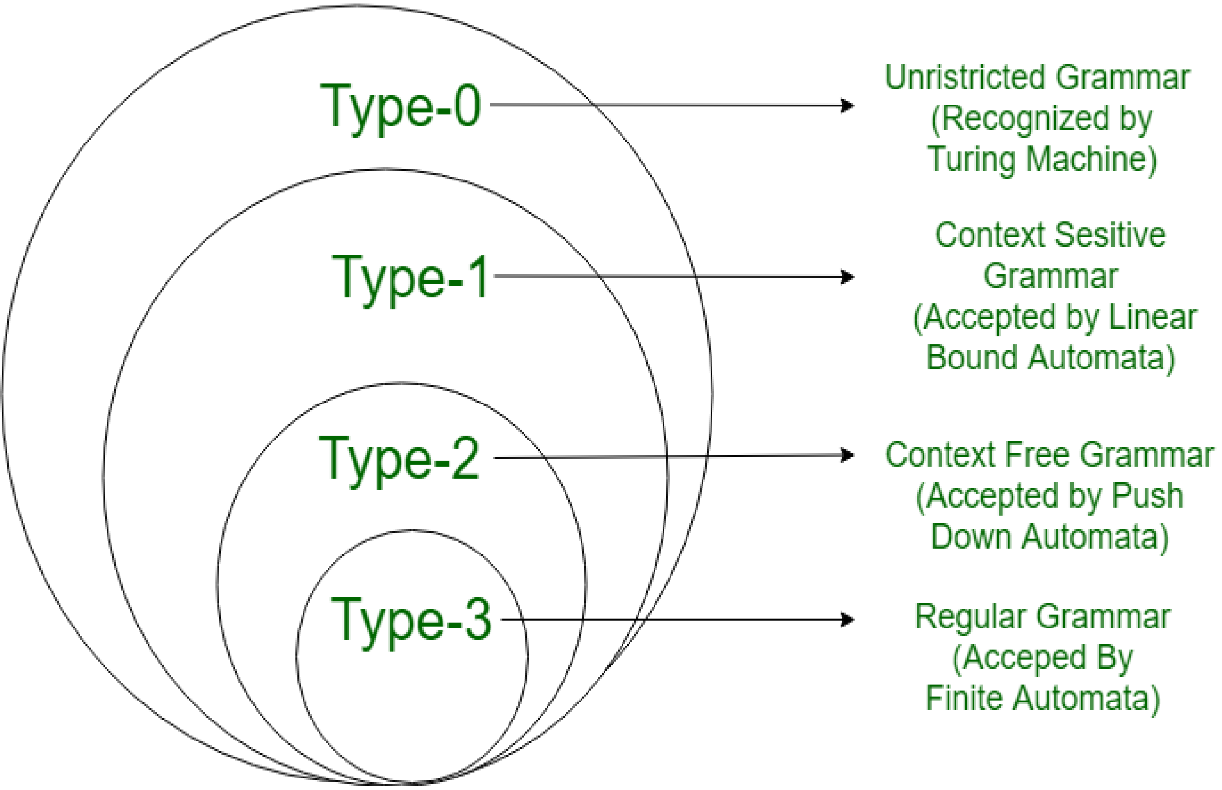
terminals

start variable

Chomsky Classification on Grammar

According to Noam Chomosky, there are four types of grammars – Type 0, Type 1, Type 2, and Type 3.

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	**Context-free grammar**	**Context-free language**	**Pushdown automaton**
Type 3	Regular grammar	Regular language	Finite state automaton




```
Type 0 known as unrestricted grammar.
Type 1 known as context sensitive grammar.
Type 2 known as context free grammar.
Type 3 Regular Grammar.
```

Type-0 grammars
The productions have no restrictions.

Type-1 grammars generate context-sensitive languages. The productions must be in the form $\boldsymbol{\alpha A \beta \rightarrow \alpha \gamma \beta}$ where $\mathbf{A \in N}$ (Non-terminal) and $\boldsymbol{\alpha, \beta, \gamma \in (V_N \cup \Sigma)^*}$ (Strings of terminals and non-terminals)
The strings **α** and **β** may be empty, but **γ** must be non-empty.

```
AB → AbBc
A → bcA
B → b
```

Type-2 grammars generate context-free languages.
The productions must be in the form **A → α,** where **A ∈** V_N (Non terminal)
and **α ∈ (**$V_N \cup \Sigma$**)***
It should be type1

```
S → Xa
X → a
X → aX
X → abc
X → ε
```

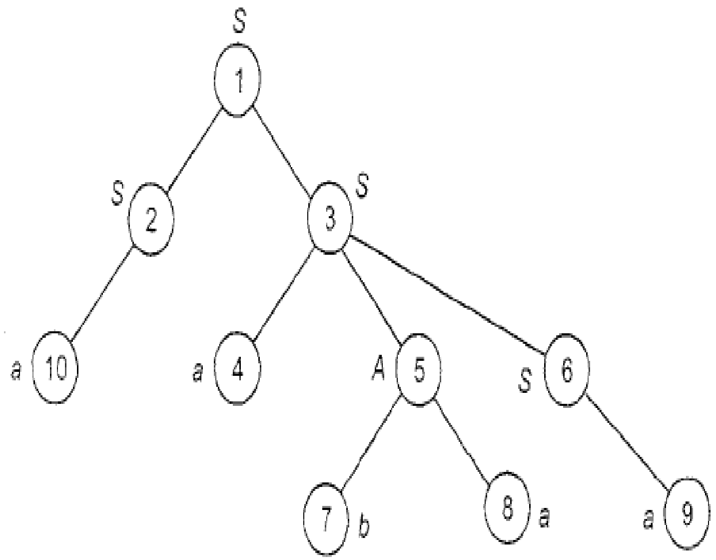
Type-3 grammars generate regular languages.
Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single
The productions must be in the form **X → a or X → aY**
where **X, Y ∈** V_N (Non terminal)
and **a ∈** Σ (Terminal)

```
X → ε
X → a | aY
Y → b
```

Derivation Trees or Parse Tree

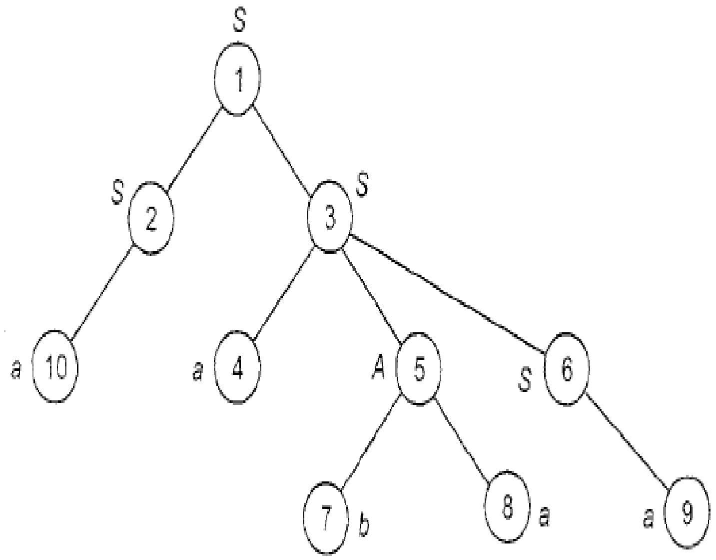
- Given a derivable string, we may represent the derivation by a derivation / parse tree.
- A parse tree is a tree with the following properties:
  - Every vertex has a label which is a variable or terminal or null (λ).
  - The root node is the start symbol S.
  - The label of an internal vertex is a variable.
  - If the vertices n1,n2,…nk written with labels X1, X2,….,Xk are the sons of vertex n with label A, then A->X1,X2,…Xk is a production in P.
  - A vertex n is a leaf if its label is a $\in \Sigma$ or null.

Example: Let G=({S,A}, {a,b}, P, S), where P consists of S->aAS|a|SS,   A->SbA|ba



Note: The yield of a derivation tree is the concatenation of the labels of the leaves without repetition in the left to right ordering

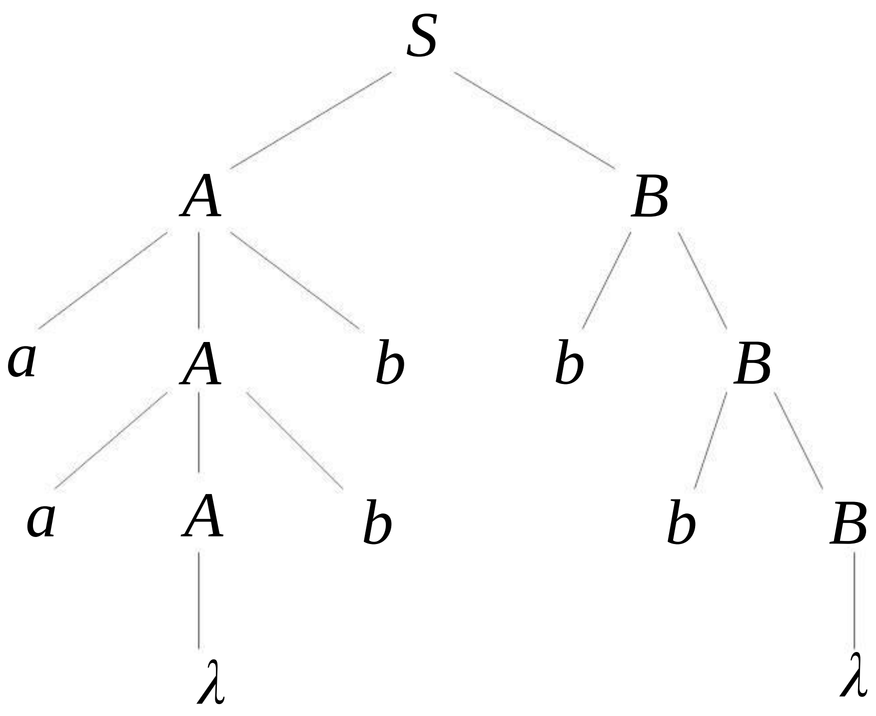
Example: Let G=({S,A}, {a,b}, P, S), where P consists of S->aAS|a|SS, A->SbA|ba. Find derivation tree for string **aabaa**



Example of a Derivation Tree

- Let the grammar G be
  - $S \rightarrow AB$
  - $A \rightarrow aAb \mid \lambda$
  - $B \rightarrow bB \mid \lambda$
- Let the string be $w = aabbbb$.
- Derive the string w.

Now draw the Parse Tree of *aabbbb*



This is the parse tree of many different, but equivalent, derivations of *aabbbb*.

For example,

$S \Rightarrow AB \Rightarrow aAbB \Rightarrow aaAbbB \Rightarrow aabbB \Rightarrow aabbbB \Rightarrow aabbbbB \Rightarrow aabbbb.$

$S \Rightarrow AB \Rightarrow AbB \Rightarrow AbbB \Rightarrow Abb \Rightarrow aAbbb \Rightarrow aaAbbbb \Rightarrow aabbbb.$

Leftmost and Rightmost Derivations

1. Leftmost Derivations

- A *leftmost derivation* of a string is a derivation in which each production rule is applied to the leftmost non terminal in the string.

  or

  A derivation $A \overset{*}{=>} w$ is called a *left most derivation* (LMD) if we apply a production only to the leftmost variable at every step.

2. Rightmost Derivations

- A *rightmost derivation* of a string is a derivation in which each production rule is applied to the rightmost non terminal in the string.

or

A derivation $A \overset{*}{\Rightarrow} w$ is called a *rightmost derivation* (*R*MD) if we apply a production only to the rightmost variable at every step.

Example: Let G be the grammar S->0B|1A, A->0|0S|1AA, B->1|1S|0BB. For the string 00110101, find:

a) LMD b) RMD c) Derivation Tree or Parse Tree

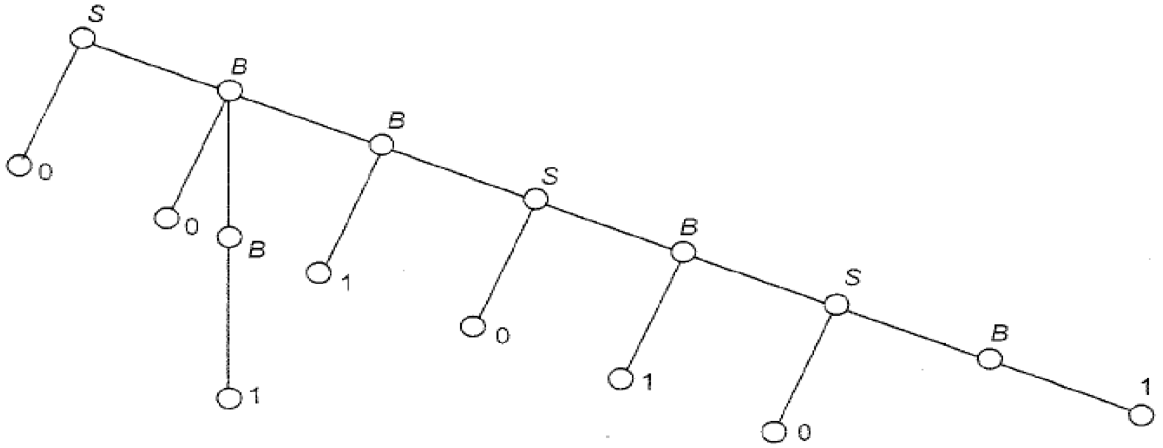
Solution: LMD:

S=>0B=>00BB=>001B=>0011S=>00110B=>001101S=>0011010B=>00110101

RMD:

S=>0B=>00BB=>00B1S=>00B10B=>00B101S=>00B1010B=>00B10101=>00110101

Parse Tree:



Ambiguous Grammars

- If there exists $w \in L(G)$ such that w has two or more different
  - parse trees or
  - leftmost (or rightmost) derivations ,
- then G is *ambiguous*.

Consider the grammar G with production rules –

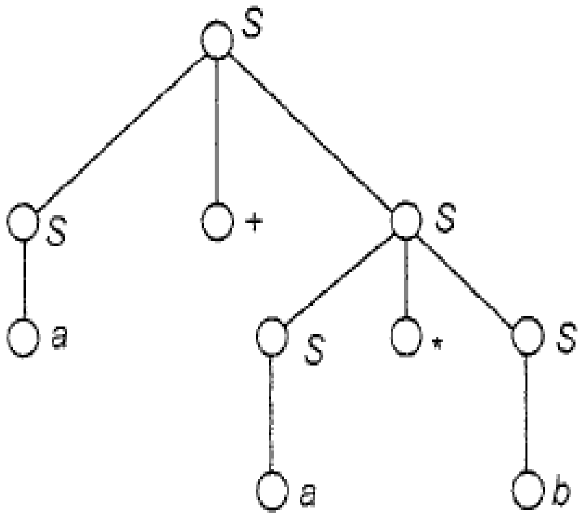
S → S+S | S*S |a| b

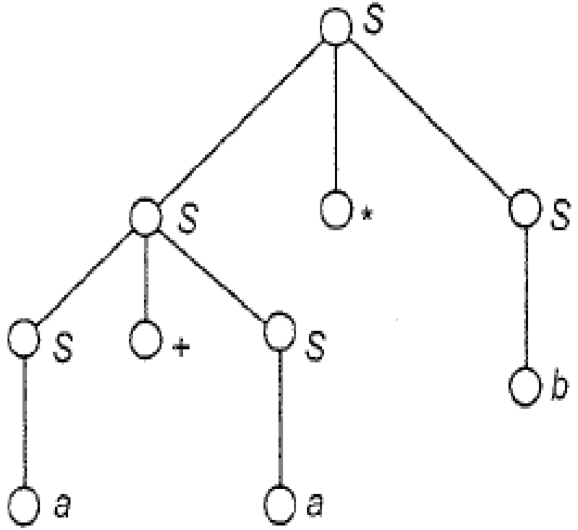
Find two Left Most Derivations of the string "**a+a*b**"

Derivation 1 : S → S+S → a +S → a+ S*S → a+a*S → a+a*b

Derivation 2 : S → S*S → S+S*S → a+ S*S → a+a*S → a+a*b

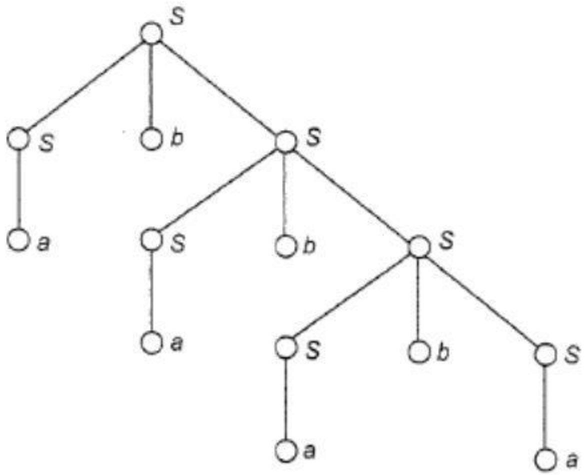
Parse trees for **a+a*b**

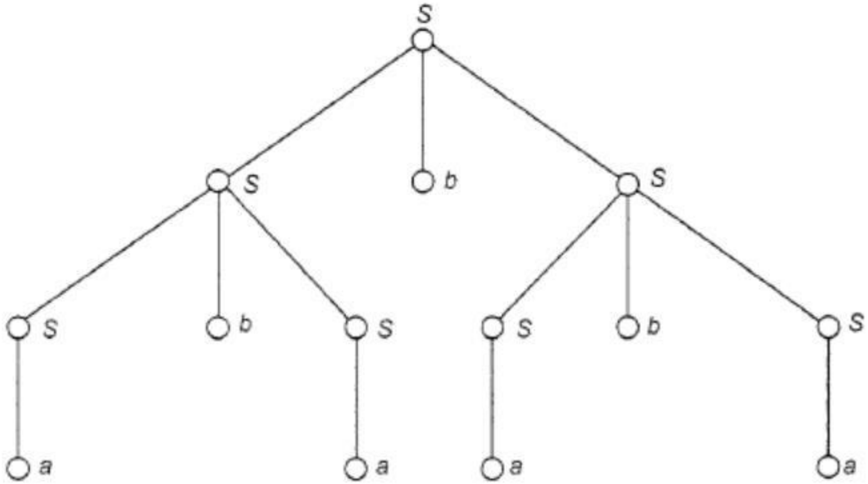




If G is the grammar S → SbS |a, show that G is ambiguous.

Consider w=abababa





Example1

E → E + E

E → E - E

E → id

Prove that grammar is ambiguous…

String "id + id - id“

First Leftmost derivation

E → E + E

→ id + E

→ id + E - E

→ id + id - E

→ id + id- id

Second Leftmost derivation

E → E - E

→ E + E - E

→ id + E - E

→ id + id - E

→ id + id - id

Example1

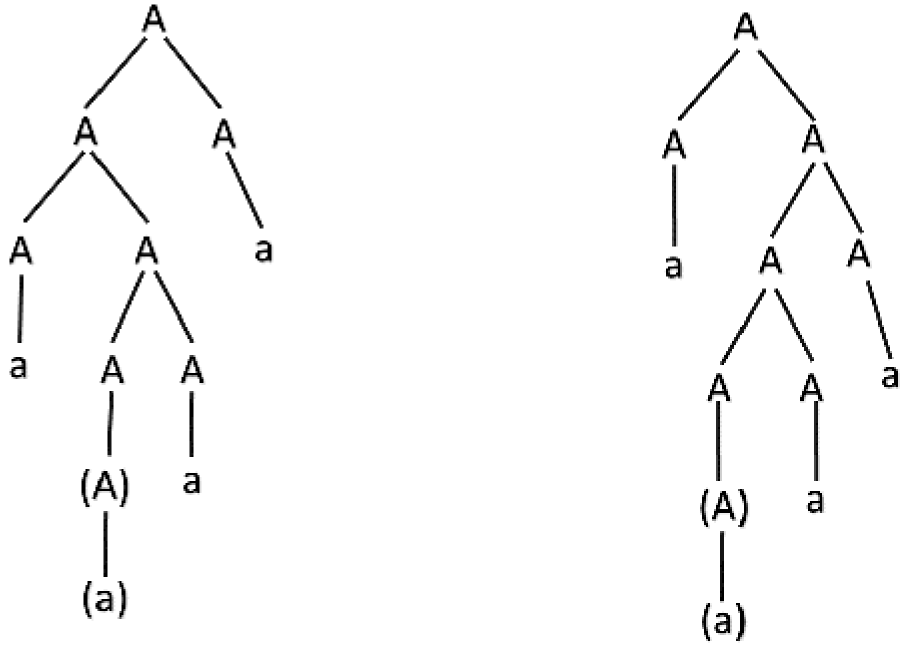
A → AA

A → (A)

A → a

For the string "a(a)aa"

Two different parse tree for the same string:



- Ambiguity can be removed by rewriting the grammar such that there is only one derivation or parse tree possible for a string of the language which the grammar represents.

E.g.

- Ambiguous Grammar

  $S \rightarrow SbS \mid a$

- Can be rewritten as

  $S \rightarrow abS \mid a$

There are some inherently ambiguous languages like $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$

Rule: $A \to A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid \dots \mid A\alpha_m \mid B_1 \mid B_2 \mid \dots \mid B_n$ where no B_i, begin with A. then we replace the A-production by:

$A \to B_1 A' \mid B_2 A' \mid \dots \mid B_n A'$

$A' \to \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_m A' \mid \lambda$

[Note: productions of the form of A->Aα, called immediate left recursion]

Example: E->E+T | T, T->T*F | F, F->(E) |a

Solution:

E->TE' **E'->+TE' | λ**

T->FT' ***T'->*FT'*** | λ

F->(E) |a

GLA UNIVERSITY
ESTD. 2010
ऋते ज्ञानान्न मुक्तिः

Simplification of CFGs

Can’t always eliminate ambiguity.

But, CFG simplification & restriction still useful theoretically & pragmatically.

- Simpler grammars are easier to understand.
- Simpler grammars can lead to faster parsing.
- Restricted forms useful for some parsing algorithms.
- Restricted forms can give you more knowledge about derivations.

- Context Free Grammar can be simplified by:
  - Useless production elimination

    or

    Construction of Reduced Grammar
  - ε production elimination
  - Unit production elimination

Useless Productions

- A non-terminal *X* is **useless** if:
  - *X* does not generate any string of terminals.
    - It cannot derive a terminal string

  OR

  - *X* does not occur in any sentential form
    - It cannot be reached from start symbol

Derive some strings

S → aSb | c | Ac

A → aA

B → bB | b

Any production involving a useless symbol is a **useless production**.

Elimination of Useless Productions

Phase 1 – Derivation of an equivalent grammar, **G′**, from the CFG, **G**, such that **each variable derives some terminal string.**

Construction of V′:

- We define $V_i' \subseteq V$ by recursion
- **$V_1'=\{A \in V \mid$ there exists a production $A \rightarrow \omega$ where $\omega \in \Sigma^*\}$**
- Vi+1′=$V_i' \cup \{A \in V \mid$ there exists some production $A \rightarrow \alpha$ with $\alpha \in (\Sigma \cup V_i')^*\}$
- At some point **$V_k'=V_{k+1}'$**. Then we get $V'=V_k'$
- Construction of P′
- $P' = \{A \rightarrow \alpha \mid A, \alpha \in (V' \cup \Sigma)^*\}$

Let G be $\mathbf{S \rightarrow AB,\ A \rightarrow a,\ B \rightarrow b,\ B \rightarrow C,}$ **and** $\mathbf{E \rightarrow c}$

Find G’ such that every variable in G’ derives some terminal string.

$V_1' = \{A, B, E\}$

$V_2' = \{A, B, E, S\}$

$V_3' = \{A, B, E, S\}$

$= V_2'$

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

$S \rightarrow AB$

Here we get $\mathbf{V' = \{S, A, B, E\}}$.

So $\mathbf{P' = S \rightarrow AB,\ A \rightarrow a,\ B \rightarrow b}$ and $\mathbf{E \rightarrow c}$

Phase 2 – Derivation of an equivalent grammar, **G″**, from the CFG, **G′**, such that each symbol appears in a sentential form.

Construction of V″:

- We define $V_i'' \subseteq V$ by recursion
- $\mathbf{V_1''=\{S\}}$
- $V_{i+1}''=V_i'' \cup \{X \in V' \mid$ there exists some production $A \rightarrow \alpha$ with $A \in V_i''$ & α contains the symbol $X\}$
- At some point $\mathbf{V_k''=V_{k+1}''}$. Then we get $V''=V_k''$
- Construction of P″
- $P'' =\{A \rightarrow \alpha \mid A, \alpha \in (V'' \cup \Sigma)^*\}$

We had **G** as **S →AB, A →a, B →b, B →C, and E → c**

Then we got **G′** as **S →AB, A →a, B →b and E → c**

$V_1'' = \{S\}$

$V_2'' = \{S, A, B\}$

$V_3'' = \{S, A, B\}$

$= V_2''$

S →AB

A →a, B →b

Here we get V″={S,A,B}.

So P″= S →AB, A →a, B →b

Question: Find a reduced grammar equivalent to the grammar G, having production rules, P: S → AC | B, A → a, C → c | BC, E → aA |e.

Solution:

Step1:

Terminals = { a, c, e }

W_1 = { A, C, E } from rules A → a, C → c and E → e

W_2 = { A, C, E } U { S } from rule S → AC

W_3 = { A, C, E, S } U Ø

Since $W_2 = W_3$, we can derive G' as –

G' = { { A, C, E, S }, { a, c, e }, P, {S}}, where P: S → AC, A → a, C → c , E → aA | e

Step2:

V_1 = { S }

V_2 = { S, A, C } from rule S → AC

V_3 = { S, A, C, a, c } from rules A → a and C → c

V_4 = { S, A, C, a, c }

Since $V_3 = V_4$, we can derive G" as –

G" = { { A, C, S }, { a, c }, P, {S}}, where P: S → AC, A → a, C → c

Construct a reduced grammar equivalent to the grammar:

S-> aAa,   A -> Sb |bCC | DaA,   C -> abb | DD,

E -> aC,   D -> aDA

Solution: Step1:

W1 ={C} as C-> abb is the only production with a terminal string on the R.H.S.

W2 = {C} U {E, A}, as E -> aC and A -> bCC are productions with R.H.S. in (Σ U {C})*

W3 = {C, E, A} U {S}, as S -> aAa and aAa is in (Σ U W2) *

w4 = W3 U Ø

Hence, Vn= W, = {S, A, C, E}

P’= {S-> aAa, A->Sb | bCC, C-> abb, E-> aC}

G’= (Vn, {a, b}, P', S)

Solution: Step2:

W1 = {S}, As we have S-> aAa,

W2 = {S} U {A, a}, As A -> Sb | bCC,

W3 = {S, A, a} U {S, b, C} = {S, A, C, a, b}

As we have C ->abb,

W4 = W3 U {a, b}= W3

Hence P”={{S-> aAa, A->Sb | bCC, C-> abb }

G”= ({S, A, C}, {a, b}, P”, S) is the reduced grammar.

Question: Find a reduced grammar equivalent to the grammar G whose productions are

S->AB|CA, B->BC|AB, A->a, C->aB|b

Solution:

Step1:

G'=({S,A,C}, {a,b}, {S->CA, A->a, C->b}, S)

Step2:

G"=({S,A,C}, {a,b}, {S->CA, A->a, C->b}, S)

Elimination of Null Productions

- A CFG may have productions of the form A → λ. So, a production in the form of A → λ, where A is variable, is c/d *null production*.

- If G is a context free grammar, then we can find a context free grammar G1 having no null productions such that L(G)=L(G)-{λ}

Step 1: Construction of the set W of all nullable variables

- $W_1=\{A_1 \in V \mid A_1 \rightarrow \lambda$ is a production in P$\}$
- $W_{i+1}= W_i \cup \{K \in V \mid \exists$ a production $K \rightarrow \alpha$ with $\alpha \in W_i^* \}$
- At some point $W_{i+1}= W_i$

$S \rightarrow aS \mid AB$

$A \rightarrow \lambda \mid a$

$B \rightarrow \lambda \mid b$

$D \rightarrow b$

$W_1=\{A, B\}$

$W_2=\{A, B, S\}$

$W_3=\{A, B, S\}$

$= W_2$

Step 2: Construction P′

- All prod^{n}s whose RHS doesn't have any nullable variables are included in P′
- If $A \to X_1 X_2 \ldots X_k$ is in P then include

$$A \to \alpha_1 \alpha_2 \cdots \alpha_k \qquad \textbf{where}$$

$$\alpha_i = \begin{cases} X_i & \text{if } X_i \notin W \\ \lambda \text{ or } X_i & \text{if } X_i \in W \end{cases}$$

Provided $A \to \alpha_1 \alpha_2 \cdots \alpha_k \neq \lambda$

$A \to a$

$B \to b$

$D \to b$

$S \to aS \mid a \mid AB \mid A \mid B$

Question: Remove null production from the following –

S → ASA | aB | b, A → B, B → b | ∈

Solution:

S→ASA | aB | b | a | SA | AS | S,   A → B, B → b

Eliminate Null productions:

1. S -> ABAC
   A -> aA / ϵ
   B -> bB / ϵ
   C -> c

Solution after null elimination:

S -> ABAC / ABC / BAC / BC / AAC / AC / C
A -> aA / a
B -> bB / b
C -> c

Eliminate Null productions:

2. S -> aSb/aAb/ab/a
   A -> ε

Solution after null elimination:

S -> aSb/aAb/ab/a

Eliminate Null(ε)–Productions

1. S –> AB, A –> aAA/ε, B –> bBB/ε

Solution:

S->AB|A|B,

A->aAA|aA|a

B->bBB|bB|b

2. S -> ABCd

   A -> BC, B -> bB | ε, C -> cC | ε

Solution:

S -> ABCd | ABd | ACd | BCd | Ad | Bd |Cd | d

A -> BC | B | C

B -> bB | b

C -> cC | c

Eliminate Null(ε)-Productions

1. S-> ABA, A-> aA/ε,   B -> bB/ε

Solution:

S->ABA|AB|BA|AA|A|B

A->aA|a

B->bB|b

2. S -> ABCd

A -> BC,   B -> bB | ε,   C -> cC | ε

Solution:

S -> ABCd | ABd | ACd | BCd | Ad | Bd |Cd | d

A -> BC | B | C

B -> bB | b

C -> cC | c

Eliminate Null Productions

$S \rightarrow ABCBCDA$

$A \rightarrow CD$

$B \rightarrow Cb$

$C \rightarrow a \mid \lambda$

$D \rightarrow bD \mid \lambda$

The CFG will have total 40 prod^{n}s!!

- 32 S prod^{n}s
- $A \rightarrow CD \mid C \mid D$
- $B \rightarrow Cb \mid b$
- $C \rightarrow a$
- $D \rightarrow bD \mid b$

Elimination of Unit Productions

$A \rightarrow B$ where $A, B \in V$

Step 1: Construction of the set variables, derivable from A

- $W_1(A)=\{A\}$
- $W_{i+1}(A)= W_i(A) \cup \{C \in V \mid$ there is a production $B \rightarrow C$ & $B \in W_i(A)\}$
- At some point $W_{i+1}(A)= W_i(A)$

$S \rightarrow A \mid bb$

$A \rightarrow B \mid b$

$B \rightarrow S \mid a$

$W(S) =\{S, A, B\}$

$W(A) =\{S, A, B\}$

$W(B) =\{S, A, B\}$

$W_1(S) =\{S\}$

$W_2(S) =\{S, A\}$

$W_3(S) =\{S, A, B\} = W_4(S)$

$W(S) =\{S, A, B\}$

Step 2: All non unit prod^{n}s are included as such

$S \rightarrow bb$

$A \rightarrow b$

$B \rightarrow a$

Step 3: Add prod^n $A \rightarrow \alpha$ whenever

- $B \in W(A)$ and
- $B \rightarrow \alpha$ is in P and
- $\alpha \notin V$

$S \rightarrow A \mid bb$

$A \rightarrow B \mid b$

$B \rightarrow S \mid a$

$W(S) = \{S, A, B\}$

$W(A) = \{S, A, B\}$

$W(B) = \{S, A, B\}$

$S \rightarrow b \mid a$

$A \rightarrow bb \mid a$

$B \rightarrow bb \mid b$

Hence prod^n s in G' are:

$S \rightarrow a \mid b \mid bb$

$A \rightarrow a \mid b \mid bb$

$B \rightarrow a \mid b \mid bb$

Eliminate Unit-productions

S -> Aa | B

A -> b | B

B -> A | a

Solution:

S -> Aa | b | a

A -> b | a

B -> a | b

B-production doesnt occur in the production ‘S’, then the following grammar becomes

S->Aa|b|a

A->b|a

Question: Remove unit production from the following:

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

Solution:

$$\boxed{S \rightarrow XY,\ X \rightarrow a,\ Y \rightarrow a \mid b,\ Z \rightarrow a,\ M \rightarrow a,\ N \rightarrow a}$$

Now Z, M, and N are unreachable, hence we can remove those.

The final CFG is unit production free –

$$\boxed{S \rightarrow XY,\ X \rightarrow a,\ Y \rightarrow a \mid b}$$

Eliminates unit production:

E -> E + T | T,   T -> T*F | F,   F-> (E)|a

Solution after Elimination of unit production:

E-> E + T | T*F | (E) | a

T ->T * F | (E) | a

F->(E) | a

Simplify the following grammar:

S -> 0A0| 1B1 |BB, A -> C, B-> S|A C-> S|ε

Solution after null removal:

S -> 0A0| 00 |1B1| 11 |B |BB
A ->C
B -> S|A
C-> S

Solution without unit production:

S -> 0A0| 00 |1B1| 11 |BB
A -> 0A0| 00 |1B1| 11 |BB
B -> 0A0| 00 |1B1| 11 |BB
C-> 0A0| 00 |1B1| 11 |BB

Simplified Grammar:

Symbol C is not reachable i.e. useless

so final simplified grammar:

S -> 0A0| 00 |1B1| 11 |BB

A -> 0A0| 00 |1B1| 11 |BB

B -> 0A0| 00 |1B1| 11 |BB

Simplify the following grammar:

S -> AB| BC |aACb | a, A-> AAB | BD |abD | C, C-> CA | S|a , D-> d , E-> ab

*Answer is not verified, pls verify it.

Solution without unit production:

S -> AB| BC |aACb | a
A-> AAB | BD |abD| CA | AB| BC |aACb | a
C-> CA | AB| BC |aACb | a
D-> d , E-> ab

Simplified Grammar:

Symbol E is not reachable i.e. useless so

final simplified grammar:

S -> AB| BC |aACb | a
A-> AAB | BD |abD| CA | AB| BC |aACb | a
C-> CA | AB| BC |aACb | a
D-> d

Normal Forms

- Chomsky Normal Form (CNF)
- Griebach Normal Form (GNF)

By:
Dr. Sandeep Rathor

Chomsky Normal Form

- A CFG is in CNF if all the prod^{n}s are in any one of the following forms
  - $A \to a$
  - $A \to BC$

  where A, B, and C are non-terminals and **a** is a terminal

Steps of Chomsky Normal Form

Step1: Elimination of null production and unit production

Step2: Elimination of terminals on RHS

Step3: Restricting the number of variable on RHS

Conversion to CNF

Algorithm to Convert into Chomsky Normal Form –

Step 1 – If the start symbol **S** occurs on some right side, create a new start symbol **S'** and a new production **S'→ S**.

Step 2 – Remove Null productions. (Using the Null production removal algorithm discussed earlier)

Step 3 – Remove unit productions. (Using the Unit production removal algorithm discussed earlier)

Step 4 – Replace each production $A \rightarrow B_1 \ldots B_n$ where $n > 2$ with $A \rightarrow B_1C$ where $C \rightarrow B_2 \ldots B_n$. Repeat this step for all productions having two or more symbols in the right side.

Step 5 – If the right side of any production is in the form **A → aB** where a is a terminal and **A, B** are non-terminal, then the production is replaced by **A → XB** and **X → a**. Repeat this step for every production which is in the form **A → aB**.

Convert the following CFG into CNF

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

- No null or unit prod^{n}s or $\varepsilon \in L$
- Add prod^{n}s already in CNF
- Convert $S \to aSa$
- Convert $S \to bSb$
- Convert $S \to aa$
- Convert $S \to bb$

$S \to a \mid b$

$S \to N_a\ S\ N_a$, $\quad N_a \to a$

$S \to N_a\ S_1$

$S_1 \to S\ N_a$

$S \to N_b\ S\ N_b \quad N_b \to b$

$S \to N_b\ S_2$

$S_2 \to S\ N_b$

$S \to N_a\ N_a$

$S \to N_b\ N_b$

Convert the following CFG into CNF

$$\mathbf{S \to aAD,\ A \to aB|bAB,\quad B \to b,\ D \to d}$$

- No null or unit prod^{n}s
- Add prod^{n}s already in CNF

  $\mathbf{B \to b,\ D \to d}$

$S \to aAD$ gives rise to $S \to CaAD$ and $Ca \to a$.
$A \to aB$ gives rise to $A \to CaB$.
$A \to bAB$ gives rise to $A \to CbAB$ and $Cb \to b$.
$V' = \{S, A, B, D, Ca, Cb\}$.

$P1$ consists of $S \to CaAD$, $A \to CaB \mid CbAB$, $B \to b$, $D \to d$, $Ca \to a$, $Cb \to b$.
$A \to CaB$, $B \to b$, $D \to d$, $Ca \to a$, $Cb \to b$ are added to $P2$
$S \to CaAD$ is replaced by $S \to CaC1$ and $C1 \to AD$.
$A \to CbAB$ is replaced by $A \to CbC2$ and $C2 \to AB$.
Let
$G2 = (\{S, A, B, D, Ca\ Cb, Cl, C2\}\ \{a, b, d\}, P2, S)$
where $P2$ consists of $S \to CaC1$, $A \to CaB|CbC2$, $C1 \to AD$, $C2 \to AB$, $B \to b$, $D \to d$, $Ca \to a$. $Cb \to b$. G2 is in CNF and equivalent to G.

Question: Convert the following CFG into CNF

$S \rightarrow ASA \mid aB,\ \ A \rightarrow B \mid S \mid \in,\ \ B \rightarrow b \mid \in$

Solution:

(1) Since **S** appears in R.H.S, we add a new state $\mathbf{S_0}$ and $\mathbf{S_0 \rightarrow S}$ is added to the production set and it becomes $\mathbf{S_0 \rightarrow S,\ S \rightarrow ASA \mid aB,\ A \rightarrow B \mid S,\ B \rightarrow b \mid \in}$

(2) Now we will remove the null productions –

$B \rightarrow \in$ and $A \rightarrow \in$

After removing $B \rightarrow \varepsilon$, the production set becomes –

$S_0 \rightarrow S,\ S \rightarrow ASA \mid aB \mid a,\ A \rightarrow B \mid S \mid \in,\ B \rightarrow b$

After removing $A \rightarrow \in$, the production set becomes –

$S_0 \rightarrow S,\quad S \rightarrow ASA \mid aB \mid a \mid AS \mid SA \mid S,\quad A \rightarrow B \mid S,\quad B \rightarrow b$

(3) Now we will remove the unit productions.

After removing $S \rightarrow S$, the production set becomes –

$S_0 \rightarrow S$, $S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$, $A \rightarrow B \mid S$, $B \rightarrow b$

After removing $S_0 \rightarrow S$, the production set becomes –

$S_0 \rightarrow ASA \mid aB \mid a \mid AS \mid SA$, $S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$

$A \rightarrow B \mid S$, $B \rightarrow b$

After removing $A \rightarrow B$, the production set becomes –

$S_0 \rightarrow ASA \mid aB \mid a \mid AS \mid SA$, $S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$

$A \rightarrow S \mid b$

$B \rightarrow b$

After removing $A \rightarrow S$, the production set becomes –

$S_0 \rightarrow ASA \mid aB \mid a \mid AS \mid SA$, $S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$, $A \rightarrow b \mid ASA \mid aB \mid a \mid AS \mid SA$, $B \rightarrow b$

(4) Now we will find out more than two variables in the R.H.S

Here, $S_0 \rightarrow ASA$, $S \rightarrow ASA$, $A \rightarrow ASA$ violates two Non-terminals in R.H.S.

Hence we will apply step 4 and step 5 to get the following final production set which is in CNF −

$S_0 \rightarrow AX \mid aB \mid a \mid AS \mid SA$

$S \rightarrow AX \mid aB \mid a \mid AS \mid SA$

$A \rightarrow b \mid AX \mid aB \mid a \mid AS \mid SA$

$B \rightarrow b$

$X \rightarrow SA$

(5) We have to change the productions $S_0 \rightarrow aB$, $S \rightarrow aB$, $A \rightarrow aB$

And the final production set becomes −

$\mathbf{S_0 \rightarrow AX \mid YB \mid a \mid AS \mid SA}$

$\mathbf{S \rightarrow AX \mid YB \mid a \mid AS \mid SA}$

$\mathbf{A \rightarrow b \mid AX \mid YB \mid a \mid AS \mid SA}$

$\mathbf{B \rightarrow b}$

$\mathbf{X \rightarrow SA}$

$\mathbf{Y \rightarrow a}$

Practice Session

Question1: $S \to aAbB$, $A \to aA \mid a$, $B \to bB \mid b$. Convert it into CNF

Solution:

G: = ({S, *A, B,* Ca, Cb,CI,C2*}, {a, b}, P2,* S),

where *P2* consists of $S \to CaC1$, $C1 \to AC2$, $C2 \to CbB$, $A \to CaA$, $B \to CbB$, $Ca \to a$, $Cb \to b$, $A \to a$ and $B \to b$.

Question2:

Find a grammar in CNF equivalent to the grammar

$$S \to \sim S \mid [S \supset S] \mid p \mid q \qquad (S \text{ being the only variable})$$

$$G_2 = (\{S, A, B, C, D, C_1, C_2, C_3\}, \Sigma, P_2, S)$$

where P_2 consists of $S \to p \mid q \mid AS \mid BC_1$, $A \to \sim$, $B \to [$, $C \to \supset$, $D \to]$, $C_1 \to SC_2$, $C_2 \to CC_3$, $C_3 \to SD$. G_2 is in CNF and equivalent to the given grammar.

Find the CNF:

S → a | aA | B

A → aBB | ε

B → Aa | b

Solution:

S → a | XA | AX | b

A → RB

B → AX | b | a

X → a

R → XB

Greibach Normal Form

- A context-free grammar is in Greibach Normal Form if the right-hand side of each rule has one terminal followed by zero or more non-terminals:

$$A \rightarrow a\,\alpha$$

where

$a \in \Sigma$

$\alpha \in V^*$

Example:

$S \rightarrow aB \mid bA$

$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

Algorithm to Convert a CFG into Greibach Normal Form

Step 1 − If the start symbol **S** occurs on some right side, create a new start symbol **S’** and a new production **S’ → S**.

Step 2 − Remove Null productions. (Using the Null production removal algorithm discussed earlier)

Step 3 − Remove unit productions. (Using the Unit production removal algorithm discussed earlier)

Step 4 − Remove all direct and indirect left-recursion.

Step 5 − Do proper substitutions of productions to convert it into the proper form of GNF.

To convert a CFG in to a GNF:

1. the grammar has to be in Chomsky Normal Form

2. there should be no Left-Recursive Rules

Lemma1: Let G=(V_N, Σ, P , S) be a CFG. Let A→ Bγ be an A-production in P. Let the B-productions be B→ β1| β2| β3|…| βs. Define P_1=(P - {A→ Bγ }U{A→ β_i γ }) |1≤i ≤ s}. Then, G1= (V_N, Σ, P_1, S) is a CFG equivalent to G.

Lemma2: Let G=(V , Σ, P , S) be a CFG. Let set of A- producitons be

A→ Aα_1 | Aα_2 | … |Aα_r |β1| β2| β3|…| βs. (βi do not start with A)

Let z be a new variable. Let G1=(G = (V_N ∪ {Z }), Σ, P_1 , S) where P_1 is defined as:

Set of A- productions:

A → β1 | β2 | … | βs

A → β1Z | β2 Z | … | βs Z

Set of Z- productions:

Z→ α_1 | α_2 | … | α_r

Z→ α_1Z | α_2 Z | … | α_r Z

Conversion to GNF

1. Eliminate **null**, **unit productions** & **useless symbols** from the grammar G
2. Convert it to **Chomsky Normal Form** (CNF) generating the language $L(G') = L(G) - \{\varepsilon\}$ where $G' = (V', \Sigma, P', S)$ in.
3. **Rename the variables** like $A_1, A_2, \ldots A_n$ starting with $S = A_1$. **Rewrite the prod^{n}s**

4. Checking for loop
   - Prod^{n}s of following form don't give rise to loops

     $A_i \rightarrow a\ X$ where $a \in \Sigma$, $X \in (V \cup \Sigma)^*$

     $A_i \rightarrow A_j\ X$ where $a \in \Sigma$, $X \in (V \cup \Sigma)^*$ and $i < j$

   - In prod^{n}s having loop, carry out substitution to get Left Recursion.

     $A_i \rightarrow A_j\ X$ where $a \in \Sigma$, $X \in (V \cup \Sigma)^*$ and $i > j$

5. **Eliminate Left Recursion** from all the Prod^{n}s
6. Do **substitution** in prod^{n}s not in GNF

Convert given CFG to GNF

$S \rightarrow AA \mid a$
$A \rightarrow SS \mid b$

Step 1 & 2: Convert to CNF

No null and unit productions and Already in CNF

Step 3: Rename the variables & Rewrite the prod^{n}s

$S - A_1$ $A - A_2$

$A_1 \rightarrow A_2 A_2 \mid a$
$A_2 \rightarrow A_1 A_1 \mid b$

Step 4

A1-productions are in the required form. They are A1 → *A2A2| a.*
(ii) *A2* → *b* is in the required form.
Apply Lemma 1 to *A2* → *A1A1.*
The resulting productions are *A2* → *A2A2A1,*
A2 → *aA1*
Thus the A2- productions are
A2 → *A2A2A1,* *A2* → *aA1, A2* → *b*

Step 5

We have to apply Lemma 2 to A2-Productions as we have A2 →*A2A2A1.* Let Z2 be the new variable. The resulting productions are
A2 → *aA1, A2* →*b*
A2 → *aA1Z2, A2* →*bZ2*
Z2→ *A2A1,* Z2→ *A2A1Z2,*

Finally A2-productions are:

$A2 \rightarrow aA1 \mid b \mid aA1Z2 \mid bZ2$ (1)

$A1 \rightarrow A2A2 \mid a$

Apply lemma1 to A1-production

$A1 \rightarrow A2A2$ then we find

$A1 \rightarrow aA1A2 \mid bA2 \mid aA1Z2A2 \mid bZ2A2$

Finally A1- productions are:

$A1 \rightarrow a \mid aA1A2 \mid bA2 \mid aA1Z2A2 \mid bZ2A2$ (2)

Z2-productions are:

$Z2 \rightarrow A2A1$, $Z2 \rightarrow A2A1Z2$, *[Apply lemma1]*

$Z2 \rightarrow aA1A1 \mid bA1 \mid aA1Z2A1 \mid bZ2A1$

$Z2 \rightarrow aA1A1Z2 \mid bA1Z2 \mid aA1Z2A1Z2 \mid bZ2A1Z2$ (3)

Equivalent grammar is:
$G' = (\{A1, A2, Z2\}, \{a,b\}, P1, A1)$ where P1 shown as: (1)(2) and (3).

Convert given CFG to GNF

$S \rightarrow AB$
$A \rightarrow BS \mid b$
$B \rightarrow SA \mid a$

Step 1 & 2: Convert to CNF

Already in CNF

Step 3: Rename the variables & Rewrite the prod^{n}s

$S - A_1$ $\quad$ $A - A_2$ $\quad$ $B - A_3$

$A_1 \rightarrow A_2 A_3$
$A_2 \rightarrow A_3 A_1 \mid b$
$A_3 \rightarrow A_1 A_2 \mid a$

Step 4: Check for Loop

$A_3 \rightarrow A_1 A_2$ $\quad$ // i > j
$A_3 \rightarrow A_2 A_3 A_2$ $\quad$ // i > j
$A_3 \rightarrow A_3 A_1 A_3 A_2 \mid b A_3 A_2$

Step 5: Eliminate Left Recursion

$A_1 \rightarrow A_2 A_3$
$A_2 \rightarrow A_3 A_1 \mid b$
$\boxed{A_3 \rightarrow A_3 A_1 A_3 A_2} \mid b A_3 A_2 \mid a$

$$A_3 \rightarrow A_3\ \underbrace{A_1\ A_3\ A_2}_{\alpha_1}\ |\ \underbrace{b\ A_3\ A_2}_{\beta_1}\ |\ \underbrace{a}_{\beta_2}$$

α's & β's ??

A prodⁿs are:

$A_3 \rightarrow b\ A_3\ A_2\ |\ a$

$A_3 \rightarrow b\ A_3\ A_2\ Z_3\ |\ a\ Z_3$

Z prodⁿs are:

$Z_3 \rightarrow A_1\ A_3\ A_2\ Z_3$

$Z_3 \rightarrow A_1\ A_3\ A_2$

5. **Step 6: Do substitution in prod^ns not in GNF**

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 \mid b$

$A_3 \rightarrow b A_3 A_2 \mid a$

$A_3 \rightarrow b A_3 A_2 Z_3 \mid a Z_3$

$Z_3 \rightarrow A_1 A_3 A_2 Z_3$

$Z_3 \rightarrow A_1 A_3 A_2$

Total 24 prod^ns

All A_3 prod^ns are in GNF

$A_3 \rightarrow b A_3 A_2 \mid b A_3 A_2 Z_3 \mid a Z_3 \mid a$

Substitute in $A_2 \rightarrow A_3 A_1 \mid b$

So we get 5 A_2 prod^ns

Substitute in $A_1 \rightarrow A_2 A_3$

So we get 5 A_1 prod^ns

Substitute in $Z_3 \rightarrow A_1 A_3 A_2 Z_3 \mid A_1 A_3 A_2$

So we get 5 + 5 Z_3 prod^ns

Convert into GNF

S → XB | AA

A → a | SA

B → b

X → a

Solution:

S → aB | aCA | aBACA | aA | aBAA

A → aC | aBAC | a | aBA

C → aCAC | aBACAC | aAC | aBAAC

C → aCA | aBACA | aA | aBAA

B → b

X → a

Find GNF of the given CFG:

E -> E + T | T,   T -> T*F | F,   F-> (E)|a

Questions for Practice

1. Convert following grammar into CNF:

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

2. Convert following grammar to PDA:

$$S \to aAA$$
$$A \to aS / bS / a$$

3. Eliminate Null and Unit production:

$$S \to aXbX$$
$$X \to aY / bY / \wedge$$
$$Y \to X / c$$

After Null

$$S \to aXbX / abX / aXb / ab$$
$$X \to aY / bY / a / b$$
$$Y \to X / c$$

After Unit

$$S \to aXbX / abX / aXb / ab$$
$$X \to aY / bY / a / b$$
$$Y \to aY / bY / a / b / c$$

Discussion

Eliminate null production:

S → ASA | aB | b, A → B, B → b | ∈

Solution:

S→ASA | aB | b | a | SA | AS | S, A → B| b, B → b

Remove unit production:

S → XY, X → a, Y → Z | b, Z → M, M → N, N → a

Solution:

S → XY, X → a, Y → a | b, Z → a, M → a, N → a (after removal of unit production)

S → XY, X → a, Y → a | b [after removal of unreachable symbols since, Z, M, and N are unreachable]

Eliminate null production:

S → XYX

X → 0X | ε

Y → 1Y | ε

Solution:

S → XY | YX | XX | X | Y

X → 0X | 0

Y → 1Y | 1

Pumping Lemma
for
Context Free Languages

- Let L be a CFL. There exists some integer n such that for all w in L with $|w| \geq n$,
- $w = uvxyz$ with $|vxy| \leq n$ and $|vy| > 0$ such that
- $uv^ixy^iz \varepsilon L$ for all $i = 0, 1, 2, 3 \ldots$.

- Let G be a grammar in CNF generating the language L – {ε}
- Let the grammar have m variables.
- Pick $n = 2^m$.
- Let $w \in L(G)$ & $|w| \geq n$
- Any derivation tree for w has height at least m+1.

Consider the following grammar.

S → AB | a

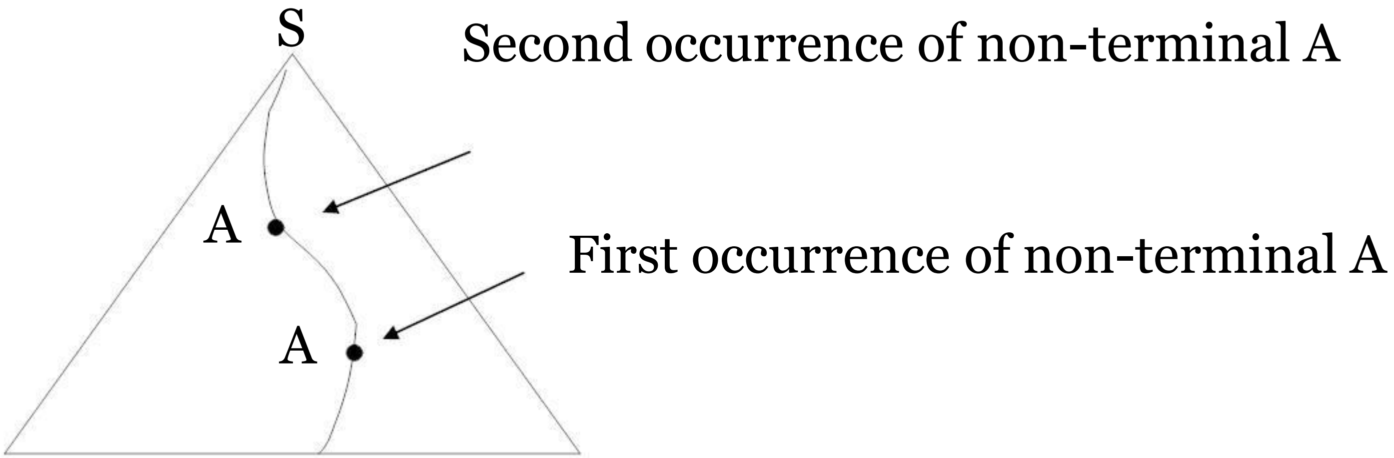
A → AS | b

B → AS | c

Take any string of length 2^p & check that the height is atleast p+1.

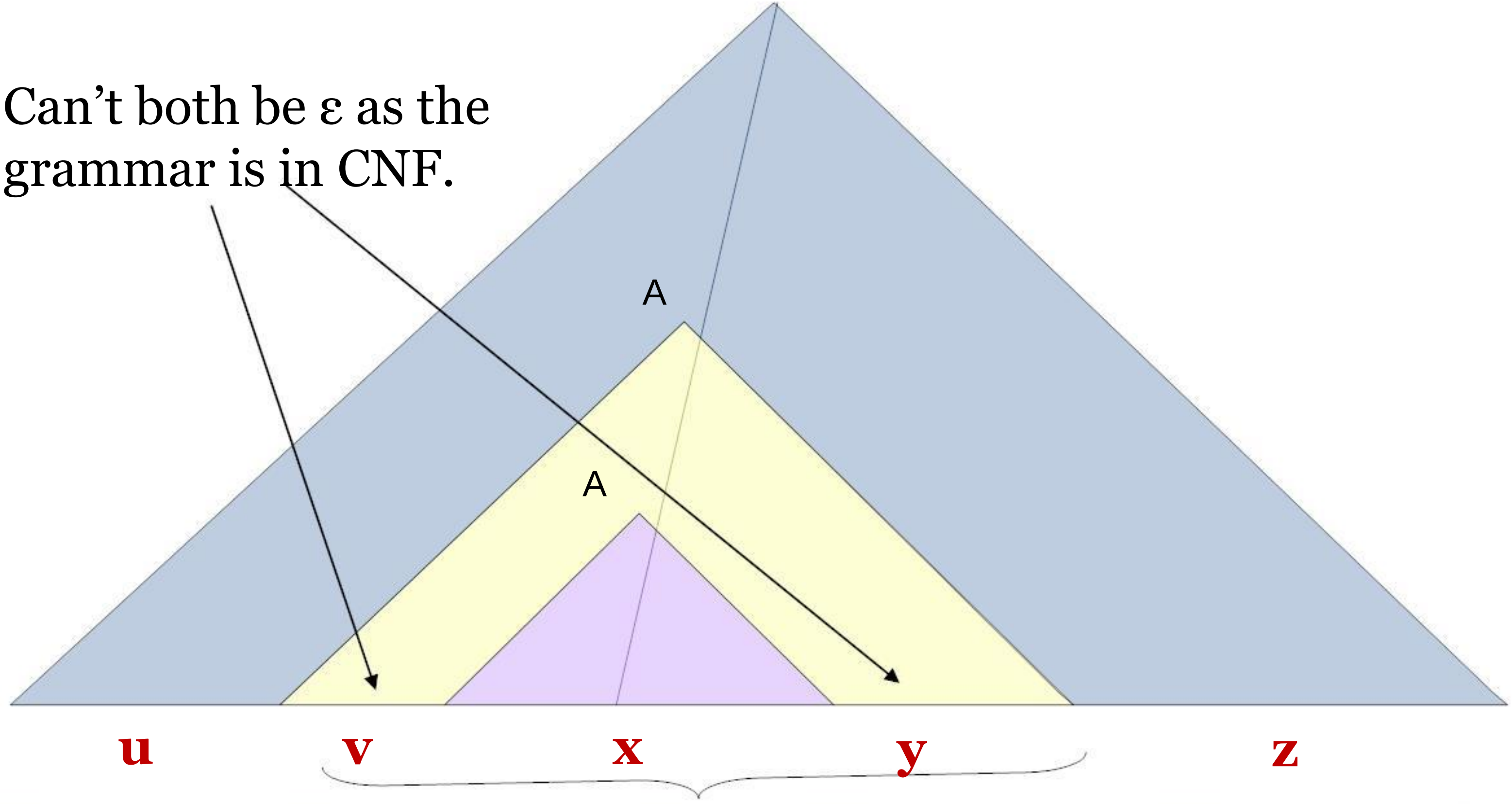
- If any derivation tree for w has height at least m+1, then the number of nodes in the path is at least m + 2.
- One node is the leaf node labeled with a terminal.
- At least m + 1 node are internal nodes labeled with non-terminals.
- Since there are only m non-terminals in the grammar, and since m+1 appear on this long path, it follows that some non-terminal (and perhaps many) appears at least twice on this path.

- Consider the first non-terminal that is repeated, when traversing the path from the leaf to the root.

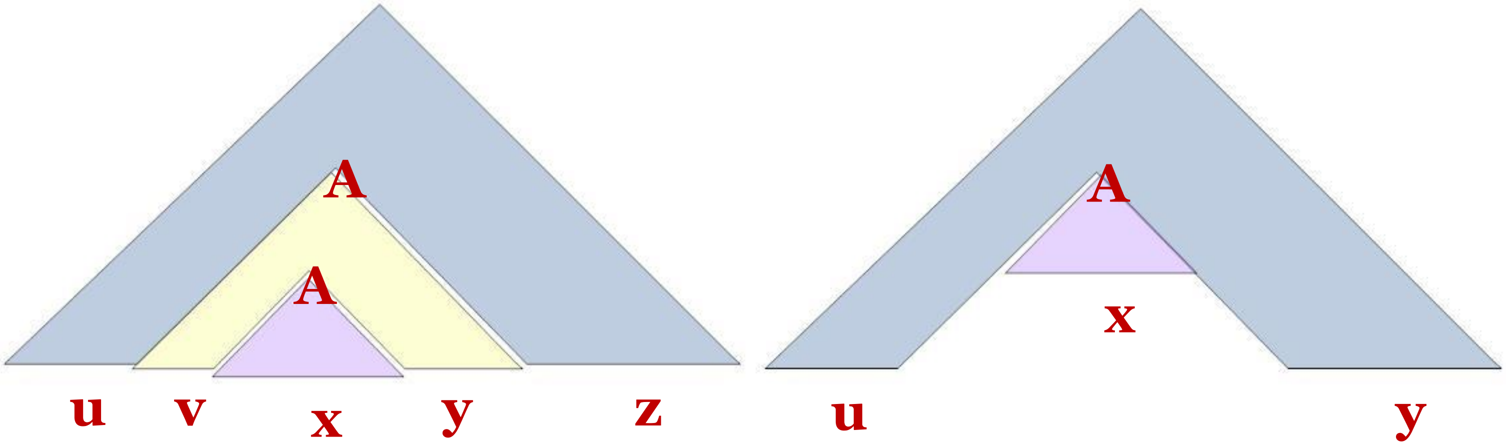


Parse Tree in the Pumping-Lemma Proof

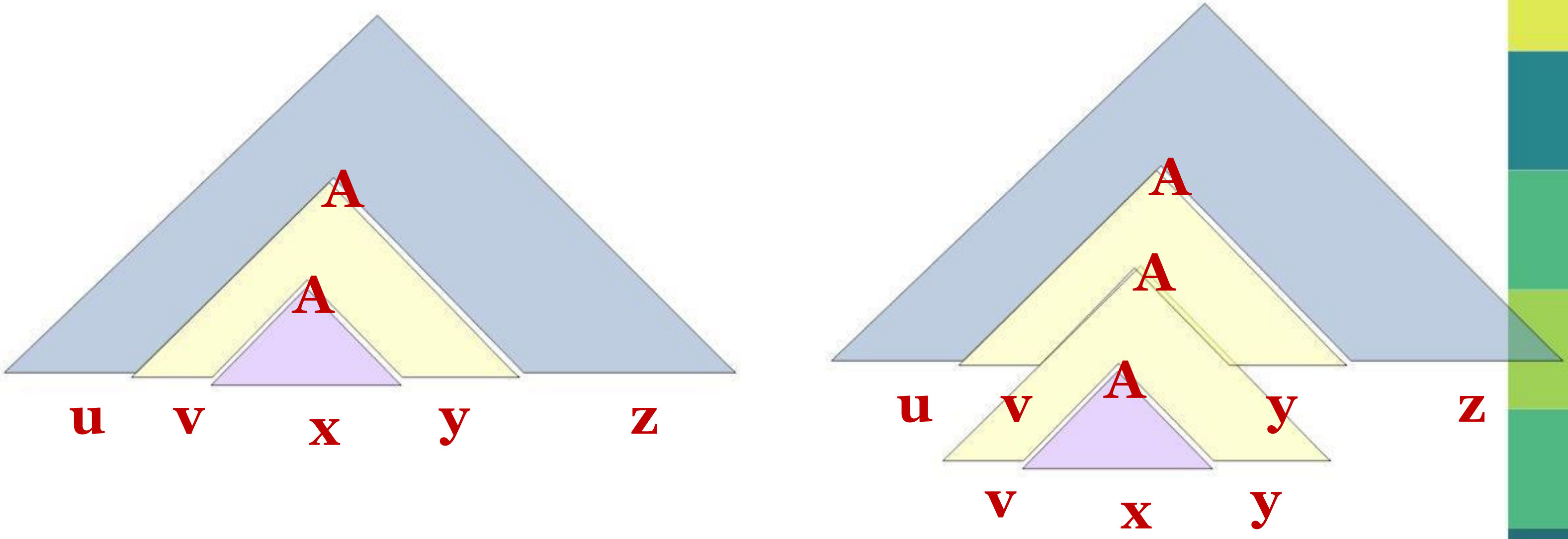
Can't both be ε as the grammar is in CNF.



Pump Zero Times



Pump Twice, Thrice …



Observe

We have the production rules which lead us to the following derivations:

- $S \Rightarrow u\,A\,z$
- $A \Rightarrow v\,A\,y$
- $A \Rightarrow x$

- $\boldsymbol{w = uvxyz}$ with $|\boldsymbol{vxy}| \leq \boldsymbol{n}$ and $|\boldsymbol{vy}| > 0$ such that
- $\boldsymbol{uv^ixy^iz} \in \boldsymbol{L}$
  for all $\boldsymbol{i} = 0, 1, 2, 3 \ldots$

Prove that $L = \{a^i b^i c^i \mid i \geq 0\}$ is not a CFL.

Proof:

- Let L be a CFL. Since L is infinite, the pumping lemma can be applied. Let **n** be the constant of the lemma.
- Let $w \in L$ & $w = a^k b^k c^k$ for some $k > n$.
  - Then we can write $w = uvxyz$
  - Such that $|vy| > 0$
  - And $|vxy| \leq n$
- Since $|vxy| \leq n$, therefore it cannot contain all 3 symbols – a, b & c

- **Case I: Suppose vxy contains one of the symbol in Σ.**
  - Suppose vxy contains at least one a.
  - Then uv^2xy^2z will more a's than b or c.
  - Therefore uv^2xy^2z does not belong to L.
  - This is a contradiction of the pumping lemma.
- **Case II: Suppose vxy contains two of the symbol in Σ.**
  - Suppose vxy contains at least one a & at least one b.
  - Then uv^2xy^2z will more a's & b's than c.
  - Therefore uv^2xy^2z does not belong to L.
  - This is a contradiction of the pumping lemma.

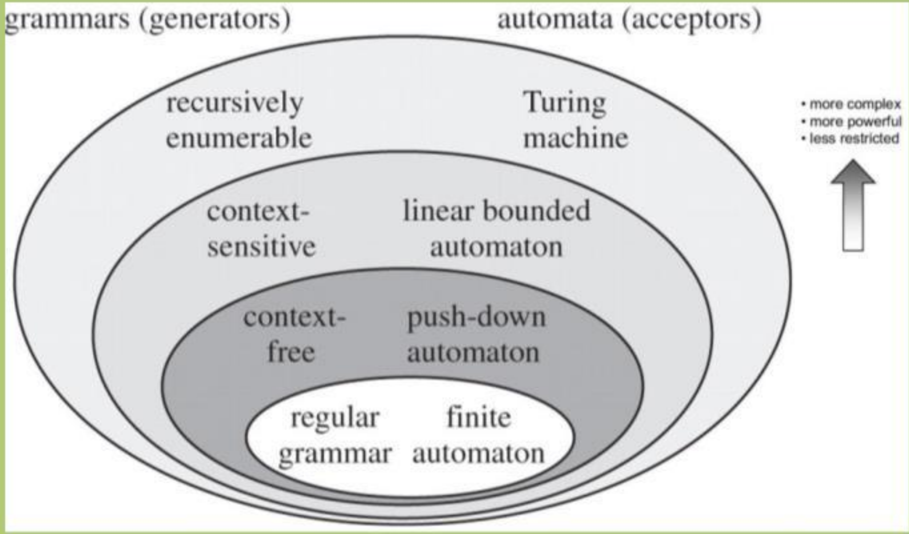
In all the cases we get a contradiction.
Therefore our assumption that L is context free is wrong.

Prove that the following languages are not CFLs.

1) $L = \{0^i 1^i 0^i 1^i \mid i > 0\}$

2) $L = \{0^{2^i} : i \geq 1\}$

Push Down Automata



Equivalence of Push Down Automata
&
Context Free Languages

By:
Dr. Sandeep Rathor

CFG to PDA

- The PDA simulates the left-sentential forms that G uses to generate any string w
- Let G = (V, Σ, P, S)
- Construct PDA N that accepts L(G) by empty stack
- **N = ({q}, Σ, V ∪ Σ, δ, q, S)**
- Transitions are defined as
  - For each variable A

$$\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \rightarrow \beta \text{ is a prod}^{n} \text{ in } G\}$$

  - For each terminal a

$$\delta(q, a, a) = \{(q, \varepsilon)\}$$

Find a PDA equivalent to the grammar **$S \rightarrow aSbb \mid a$**
Generate the string aabb and also simulate the PDA for it

- **$N = (\{q\}, \Sigma, V \cup \Sigma, \delta, q, S)$**

Where $\Sigma = \{a, b\}$, $V = \{S\}$ & δ is as following

1. $\delta(q, \varepsilon, S) = \{(q, aSbb), (q, a)\}$
2. $\delta(q, a, a) = \{(q, \varepsilon)\}$
3. $\delta(q, b, b) = \{(q, \varepsilon)\}$

$S \Rightarrow aSbb \Rightarrow aabb$

$$
\begin{aligned}
& (q, aabb, S) \\
\vdash\ & (q, aabb, aSbb) \\
\vdash\ & (q, abb, Sbb) \\
\vdash\ & (q, abb, abb) \\
\vdash\ & (q, bb, bb) \\
\vdash\ & (q, b, b) \\
\vdash\ & (q, \varepsilon, \varepsilon)
\end{aligned}
$$

Stack Empty – string accepted

Find a PDA equivalent to the following grammar that generates simple expressions like a * (a + b010)

$$E \rightarrow I \mid E * E \mid E + E \mid (E)$$
$$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

Let N be a PDA that accepts the language generated by the given CFG

- $N = (\{q\}, \Sigma, V \cup \Sigma, \delta, q, S)$
- $\Sigma = \{a, b, 0, 1, (,), *, +\}$
- $V = \{E, I\}$

1. $\delta(q, \varepsilon, E) = \{(q, I), (q, E*E), (q, E+E), (q,(E))\}$
2. $\delta(q, \varepsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$
3. $\delta(q, a, a) = \{(q, \varepsilon)\}$
4. $\delta(q, b, b) = \{(q, \varepsilon)\}$
5. $\delta(q, 0, 0) = \{(q, \varepsilon)\}$
6. $\delta(q, 1, 1) = \{(q, \varepsilon)\}$
7. $\delta(q, (, () = \{(q, \varepsilon)\}$
8. $\delta(q,),)) = \{(q, \varepsilon)\}$
9. $\delta(q, *, *) = \{(q, \varepsilon)\}$
10. $\delta(q, +, +) = \{(q, \varepsilon)\}$

PDA to CFG

We construct grammar G as follows.

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

- **$V = \{S\} \cup \{q, z, q'] \mid q, q' \in Q, z \in \Gamma\}$**

The productions in P are induced by the moves of PDA as follows

- **Rule 1:**

The S productions are given by

$$\mathbf{S \rightarrow [q_o, Z_o, q] \ \forall\ q \in Q}$$

Rule 2:

Each move erasing a pushdown symbol given by $(q', \varepsilon) \in \delta(q, a, z)$ induces the prod^n

$$[q, z, q'] \rightarrow a$$

Rule 3:

Each move not erasing a pushdown symbol given by $(q_1, z_1 z_2 \ldots z_m) \in \delta(q, a, z)$ induces many prod^{n}s of the following form where $q', q_2, \ldots, q_m$ can be any state in Q

$$[q, z, q'] \rightarrow a\,[q_1, z_1, q_2]\,[q_2, z_2, q_3] \ldots [q_m, z_m, q']$$

Construct a context-free grammar G which accepts L(N), where

$$\mathbf{N = (\{q_0, q_1\}, \{a. b\}, \{Z_0, Z\},}\ \delta,\ \mathbf{q_0, Z_0, \varphi)}$$

δ is given by

1. $\delta(q_0, b, Z_0) = \{(q_0, ZZ_0)\}$
2. $\delta(q_0, \varepsilon, Z_0) = \{(q_0, \varepsilon)\}$
3. $\delta(q_0, b, Z) = \{(q_0, ZZ)\}$
4. $\delta(q_0, a, Z) = \{(q_1, Z)\}$
5. $\delta(q_1, b, Z) = \{(q_1, \varepsilon)\}$
6. $\delta(q_1, a, Z_0) = \{(q_0, Z_0)\}$

Let G = {V, {a, b}, P, S}

V = {S} $\cup$ {q, z, q′] | q, q′ $\in$ Q, z $\in$ Γ}

V = {S} $\cup$ {[q0, Z0, q0], [q0, Z0, q1],

[q0, Z , q0], [q0, Z , q1],

[q1, Z0, q0], [q1 , Z0, q1],

[q1, Z , q0], [q1, Z , q1],

The productions in P are induced by the moves of PDA

- **Rule 1: $S \rightarrow [q_0, Z_0, q] \; \forall \; q \in Q$**

1. $S \rightarrow [q_0, Z_0, q_0]$
2. $S \rightarrow [q_0, Z_0, q_1]$

- **Rule 2:** $\delta(q, a, z) = (q', \varepsilon)$ induces $[q, z, q'] \to a$

- $\delta(qo, \varepsilon, Zo) = \{(qo, \varepsilon)\}$

3. $[qo, Zo, qo] \to \varepsilon$

- $\delta(q1, b, Z) = \{(q1, \varepsilon)\}$

4. $[q1, Z, q1] \to b$

Rule 3: $\delta(q, a, z) = (q_1, z_1 z_2 \dots z_m)$

$$[q, z, q'] \rightarrow a\, [q_1, z_1, q_2]\, [q_2, z_2, q_3] \dots [q_m, z_m, q']$$

- $\delta(qo, b, Zo) = \{(qo, ZZo)\}$

$$[qo, Zo, \blacklozenge\,] \rightarrow b\, [qo, Z, \blacklozenge\,]\, [\blacklozenge, Zo, \blacklozenge\,]$$

5. [qo, Zo, qo] → b [qo, Z, qo] [qo, Zo, qo]
6. [qo, Zo, qo] → b [qo, Z, q1] [q1, Zo, qo]
7. [qo, Zo, q1] → b [qo, Z, qo] [qo, Zo, q1]
8. [qo, Zo, q1] → b [qo, Z, q1] [q1, Zo, q1]

- δ(q0, b, Z) = {(q0, ZZ)}

[q0, Z, ◆] → b [q0, Z, ◆] [◆, Z, ◆]

9. [q0, Z, q0] → b [q0, Z, q0] [q0, Z, q0]
10. [q0, Z, q0] → b [q0, Z, q1] [q1, Z, q0]
11. [q0, Z, q1] → b [q0, Z, q0] [q0, Z, q1]
12. [q0, Z, q1] → b [q0, Z, q1] [q1, Z, q1]

- $\delta(q_0, a, Z) = \{(q_1, Z)\}$

$$[q_0, Z, \blacklozenge\ ] \rightarrow a\ [q_1, Z, \blacklozenge\ ]$$

13. $[q_0, Z, q_0\ ] \rightarrow a\ [q_1, Z, q_0\ ]$
14. $[q_0, Z, q_1\ ] \rightarrow a\ [q_1, Z, q_1\ ]$

- $\delta(q_1, a, Z_0) = \{(q_0, Z_0)\}$

$$[q_1, Z_0, \blacklozenge\ ] \rightarrow a\ [q_0, Z_0, \blacklozenge\ ]$$

15. $[q_1, Z_0, q_0\ ] \rightarrow a\ [q_0, Z_0, q_0\ ]$
16. $[q_1, Z_0, q_1\ ] \rightarrow a\ [q_0, Z_0, q_1\ ]$

Closure Properties of Languages

Property	*Regular*	*CFL*	*DCFL*	*CSL*	*Recursive*	*RE*
Union	Yes	Yes	No	Yes	Yes	Yes
Intersection	Yes	No	No	Yes	Yes	Yes
Set Difference	Yes	No	No	Yes	Yes	No
Complementation	Yes	No	Yes	Yes	Yes	No
Intersection with a regular lang.	Yes	Yes	Yes	Yes	Yes	Yes
Concatenation	Yes	Yes	No	Yes	Yes	Yes
Kleen Closure	Yes	Yes	No	Yes	Yes	Yes
Kleen Plus	Yes	Yes	No	Yes	Yes	Yes
Reversal	Yes	Yes	No	Yes	Yes	Yes
Homomorphism	Yes	Yes	No	No	No	Yes
[illegible] Homomorphism	Yes	Yes	No	Yes	Yes	Yes

4. Convert following grammar into CNF:

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

5. Convert following grammar to PDA:

$$S \to aAA$$
$$A \to aS / bS / a$$

6. Eliminate Null and Unit production:

$$S \to aXbX$$
$$X \to aY / bY / \wedge$$
$$Y \to X / c$$

After Null

$$S \to aXbX / abX / aXb / ab$$
$$X \to aY / bY / a / b$$
$$Y \to X / c$$

After Unit

$$S \to aXbX / abX / aXb / ab$$
$$X \to aY / bY / a / b$$
$$Y \to aY / bY / a / b / c$$