Theory of Computation
Chapter 2: Context Free Language

GATE Computer Science Lectures

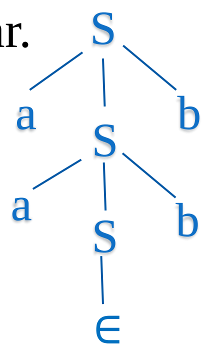
By

Monalisa Pradhan

- **Section 6: Theory of Computation($\cong$ 10mark)**
  Regular expressions and finite automata. Context-free grammars and push-down automata. Regular and context free languages, pumping lemma. Turing machines and undecidability.

- Chapter 1:Regular Language [RL,FA,RE ,Pumping lemma]
- Chapter 2: Context free Language [Grammar(RG,CFG),CFL,PDA, Pumping lemma]
- Chapter 3: Recursive enumerable Language [CSL, LBA ,RS,RES,TM]
- Chapter 4: Undecidability

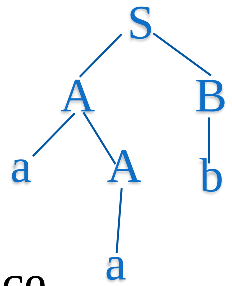
Grammar



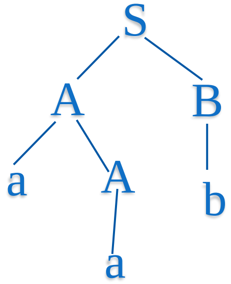
- Finite Set of rules which are used to generate the string is called as grammar.
- It has 4 tuples G=(V,T,P,S)
- V=set of variables or non-terminal symbols
- T=Set of all terminal
- P=Set of production rules
- S=Start Symbol
- Ex:S→aSb/∈ ,V={S},T={a,b},P={S→aSb/∈},S={S}
- Derivation: The process of deriving a string is called as derivation .Graphical representation of derivation is called derivation tree or parse tree.
- w=aabb,S →aSb →aaSbb →aa∈ bb →aabb
- Type of Derivation
- *1.Left Most Derivation* : The process of deriving a string by expanding left most Variable is called LMD and graphical representation called LMDT.
- *2.Right most Derivation* : The process of deriving a string by expanding right most non terminal is called RMD and graphical representation called RMDT

- Ex:S→AB,
- A →aA/a,
- B →bB/b
- w=aab

- LMD
- S→AB
- S→aAB
- S→aaB
- S→aab

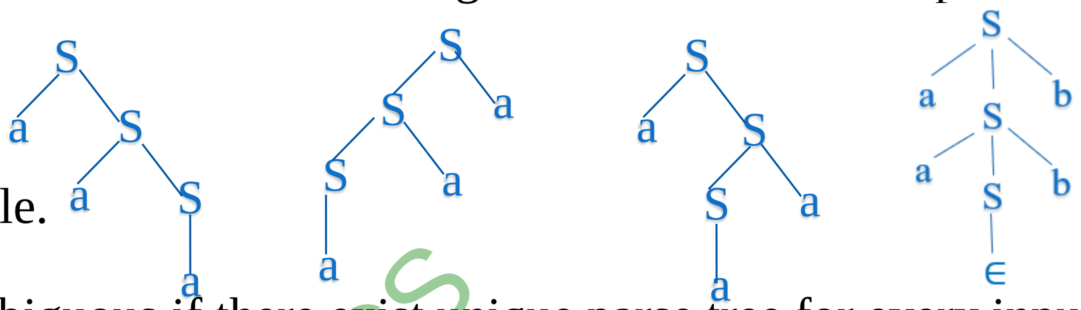


- RMD
- S→AB
- S→Ab
- S→aAb
- S→aab



- Grammar is generating device.
- Every grammar contain only one start symbol.
- The derivation of any string start from start symbol.
- If G is any grammar then L(G) is language generated by G.
- Every Grammar generate only one language but a language can be generated by more than one form of Grammar. So it is not unique.
- Two grammar G_1 & G_2 are equal iff $L(G_1)=L(G_2)$
- <u>Classification of Grammar</u>
- Grammar can be classified in two ways
- 1.Based on Derivation tree
  - Ambiguous Grammar
  - Unambiguous Grammar
- 2.Based on number of string
  - Recursive Grammar
  - Non Recursive Grammar

- Ambiguous Grammar: The grammar is said to be ambiguous if more than one parse tree exist for at least one string.
- Ex:S→aS/Sa/a ,w=aaa



- Ambiguity of CFG is undecidable.
- Unambiguous Grammar :
- The grammar is said to be unambiguous if there exist unique parse tree for every input string.Ex:S →aSb/ ∈,w=aabb
- No algorithm exist to convert ambiguous grammar to unambiguous grammar except operator grammar.
- The Ambiguous grammar which can't be converted to unambiguous is called inherent Ambiguous grammar .
- Recursive Grammar :If at least one production contain same variable both at LHS and RHS. Ex:S →aSb/ ∈
- Non Recursive Grammar :If no Product contain same variable both at LHS and RHS
- Ex:S→aA/b,A→a
- Non Recursive →Finite Language
- Recursive →Infinite Language

- <u>Construct Grammar:</u>
- **Finite Language(RL)**
- $L_1=\{a,ab\}$
- S → aA
- A → b/∈
- $L_2=\{\in ,a,b,ab\}$
- S → AB
- A → a/∈
- B →b/ ∈
- $L_3=\{a^n b^n \mid 0\leq n \leq 2\}$
- $L_3=\{\in, ab, aabb\}$
- S →aAb/ ∈
- A →ab/ ∈
- $L_4=\{a^m b^n \mid m+n=2\}$
- $L_4 =\{aa,ab,bb\}$
- S →aA/bb
- A →a/b
- $L_5=\{a^m b^n \mid m.n=2\}$
- $L_5 =\{aab,abb\}$
- S →aAb
- A →a/b
- $L_6=\{w\in \Sigma^* \mid |w|=3,w=w^r\}$
- $L_6=\{aaa,aba,bab,bbb\}$
- S →aAa/bAb
- A →a/b
- **Infinite Language(RL)**
- $L_1=a^*b$
- S → aS/b
- $L_2=a^+$
- S → aS/a
- $L_3=a^*$
- S → aS/∈
- $L_4=ba^*$
- S → bA
- A →aA/∈
- $L_5=a^*b^*$
- S →AB
- A →aA/∈
- B →bB/∈
- $L_6=(ab)^n \mid n>0$
- S→ aA
- A →bS/b

- $\Sigma=\{a,b\}$
- $L_7=(a+b)^*$ S→ aS/bS/ ∈
- $L_8=(a+b)^+$ S→ aS/bS/a/b
- L_9={Every String Start with a}
- S→ aA
- A →aA/bA/∈
- L_{10}={Every String end with b}
- S→ Ab
- A →aA/bA/∈
- L_{11}={Every String contain substring ab}
- S→ AabA
- A →aA/bA/∈
- L_{12}={Every String Start & end with a}
- S→ aAa/a
- A →aA/bA/∈
- L_{13}={Every String Start & end with Same symbol}
- S→ aAa/bAb/a/b
- A →aA/bA/∈
- L_{14}={Every String Start & end with different symbol}
- S→ aAb/bAa
- A →aA/bA/∈
- L_{15}={The 3rd symbol from left end is a}
- S→ AAaB
- A→a/b
- B→aB/bB/∈
- L_{16}={The 4th symbol from right end is b}
- S→AbBBB
- A→aA/bA/ ∈
- B →a/b

- $\Sigma=\{a,b\}$
- L_{17}={Length of the string is exactly 3}
- S→ XXX
- X→a/b
- L_{18}={Length of the string is at most 3}
- S→ XXX
- X→a/b /∈
- L_{19}={Length of the string is at least 3}
- S→AB
- A→XXX
- X→a/b
- B →aB/bB/∈
- L_{20}={Length of the string is even}
- S→ XXS /∈
- X→a/b
- L_{21}={Length of the string is odd}
- S→ XY
- X→a/b
- Y →XXY /∈

- L_{22}={Length of the string =2(mod 3)}
- RE=$(a+b)^2((a+b)^3)^*$
- S → XXB
- X→a/b
- B →XXXB/∈
- $L_{23}=\{a^m b^n | m,n \geq 0\}$
- S →aS/Sb/∈
- Or
- S →AB
- A →aA/∈
- B →bB/∈
- $L_{24}=\{a^m b^n | m,n \geq 1\}$
- S →AB
- A →aA/a
- B →bB/b

- $L_{25}=\{a^m b^n | m \geq 0, n \geq 1\}$
- S →AB
- A →aA/∈
- B →bB/b
- $L_{26}=\{a^m b^n | m \geq 1, n \geq 0\}$
- S →AB
- A →aA/a
- B →bB/ ∈

Context Free Grammar

- $L_1=\{a^m b^n | m=n, m,n \geq 0\}$
- $S \rightarrow aSb / \in$
- $L_2=\{a^m b^n | m \geq n, m,n \geq 0\}$
- $S \rightarrow aS/aSb/ \in$
- $L_3=\{a^m b^n | m \leq n, m,n \geq 0\}$
- $S \rightarrow Sb/aSb/ \in$
- $L_4=\{a^m b^n | m=2n, m,n \geq 0\}$
- $S \rightarrow aaSb/ \in$
- $L_5=\{a^m b^n | m=n+2, m,n \geq 0\}$
- $S \rightarrow aSb/aa$
- $L_6=\{a^m b^n c^m | m,n \geq 0\}$
- $S \rightarrow aSc/A$
- $A \rightarrow bA/\in$
- $L_7=\{a^m b^n c^n | m,n \geq 0\}$
- $S \rightarrow aS/A$
- $A \rightarrow bAc/\in$
- or $S \rightarrow AB$
- $A \rightarrow aA/\in$
- $B \rightarrow bBc /\in$
- $L_8=\{a^m b^m c^n | m,n \geq 0\}$
- $S \rightarrow Sc/A$
- $A \rightarrow aAb/\in$
- $L_9=\{a^m b^n c^p | m=n \text{ or } n=p, m,n,p \geq 0\}$
- $S \rightarrow A/B$
- $A \rightarrow XY$
- $X \rightarrow aXb/\in$
- $Y \rightarrow cY/\in$
- $B \rightarrow PQ$
- $P \rightarrow aP /\in$
- $Q \rightarrow bQc /\in$
- $L_{10}=\{a^m b^n c^p | n=m+p, m,n,p \geq 0\}$
- $=a^m b^m b^p c^p$
- $S \rightarrow AB$
- $A \rightarrow aAb/\in$
- $B \rightarrow bBc/\in$

- $L_{11}=\{a^m b^n c^n d^m | m,n \geq 0\}$
- S → aSd/A
- A →bAc /∈
- $L_{12}=\{wxw^R | w \in (a+b)^*\}$
- S → aSa/bSb/x
- L_{13}=even Palindrome or $\{ww^R | w \in (a+b)^*\}$
- S → aSa/bSb/∈
- L_{14}=odd Palindrome or $\{waw^R \cup wbw^R | w \in (a+b)^*\}$
- S → aSa/bSb/a/b
- L_{15}=Palindrome or $\{waw^R \cup wbw^R \cup ww^R | w \in (a+b)^*\}$
- S → aSa/bSb/a/b/∈
- $L_{16}=\{n_a(w)=n_b(w)\}$
- S → SaSbS/SbSaS/∈
- $L_{17}=\{n_a(w)=2n_b(w)\}$
- S → SaSaSbS/SbSaSaS/SaSbSaS/∈

- **Context Sensitive Grammar**
- $L_1=\{a^n b^n c^n | n \geq 1\}$
- S →aSAc/abc
- cA →Ac
- bA →bb
- Let w=aabbcc
- S → aSAc
-    → aabcAc
-    → aabAcc
-    → aabbcc

Chomsky Hierarchy:

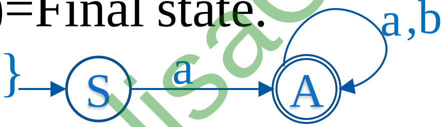
Types	Language	Grammar	Automata
Type 0	Recursive Enumerable Language	Recursive Enumerable Grammar	Turing Machine
Type 1	Context sensitive language	Context sensitive Grammar	Linear bounded Automata
Type 2	Context free Language	Context free Grammar	Push Down Automata
Type 3	Regular Language	Regular Grammar	Finite Automata

- **REG:**
- If every production is in the form $\alpha \rightarrow \beta$
- $\alpha \in (V+T)^+ , \beta \in (V+T)^*$
- Ex:aA →bB/∈
- Every Problem which is executable or decidable is REL
- **CSG**:
- If every production is in the form $\alpha \rightarrow \beta$
- $\alpha, \beta \in (V+T)^+, \beta \neq \in ,|\alpha| \leq |\beta|$
- Ex:aA →bAb/bb

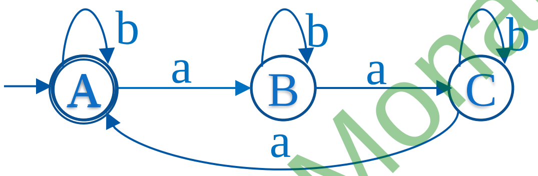
- CSG doesn't contain any production to generate ∈.
- Every Programming Language is CSL.
- Ex:L={$a^nb^nc^n$/n≥1},L={ww/w ∈$(a+b)^+$}
- **CFG**:
- If every production is in the form $\alpha \rightarrow \beta$
- α ∈ V, β ∈ (V+T)∗
- Ex:S →aSb/ ∈
- DCFL is used to define Programming Language.
- **RG:**
- If every production is in the form $\alpha \rightarrow x\beta/x$ or $\alpha \rightarrow \beta x/x$
- α, β ∈ V ,x ∈T*
- Ex:S →aS/∈
- **Type of Regular Grammar**
- 1.Right Linear Grammar(RLG)
- 2.Left Linear Grammar(LLG)
- RLG has right associative & LLG has Left Associative
- Every RG is Unambiguous.
- Every RG is also CFG.

Conversion of Right linear Grammar → Finite Automata:

- V →T*V/T*
- Start symbol is initial state.
- If A→B is a production then δ(A,∈)=B
- If A→aB is a production then δ(A,a)=B
- If A→∈ is a production then A is a final state.
- If A→a is a production then δ(A,a)=Final state.
- Ex1: L_1={Every String Start with a}



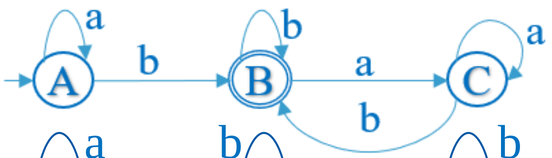
- S→ aA
- A→aA/bA/∈



- Ex2:
- A →aB/bA/b
- B →aC/bB
- C →aA/bC/a

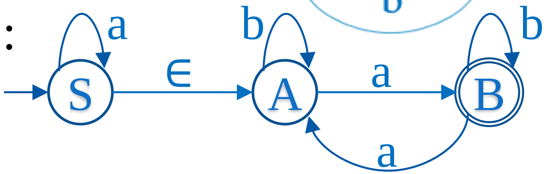
- Ex3:
- S→aS/bS/ ∈

- **Conversion of Finite Automata → Right linear Grammar :**
- Initial state is the start symbol.
- Number of state=number of non terminal.
- If δ(A,a)=B is a transition then A →aB is Production.
- If δ(A,∈)=B is a transition then A→B is production.
- If B is final state then add B → ∈.
- In case of DFA remove dead state then find Grammar.
- Consider out degree.
- Ex1:



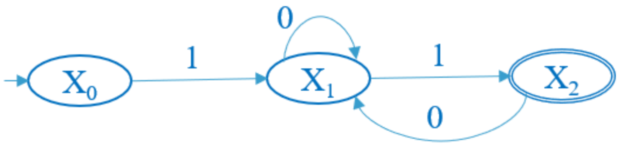
- A→aA/bB
- B →aC/bB/∈
- C →aC/bB

- Ex2:



- S→aS/A
- A →aB/bA
- B →aA/bB/∈

- Ex3:GATE CS 2015 Set-2,Q35



- $X_0 \rightarrow 1X_1$
- $X_1 \rightarrow 0X_1/1X_2$
- $X_2 \rightarrow 0X_1/\in$

$X_0 = 1X_1$
$X_1 = 0X_1 + 1\ X_2$
$X_2 = 0\ X_1 + \{\lambda\}$

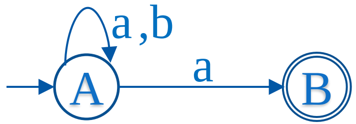
Conversion of Left linear Grammar → Finite Automata:

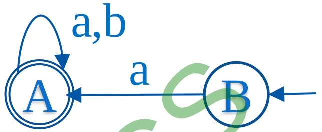
- V →V T* /T*
- Reverse the RHS of every production
- Construct FA according to RLG →FA process
- Reverse FA

LLG → RLG → FA → FA^R

L → L^R → L^R → L

- Ex1:
- A→ Aa/Ab/Ba
- B→∈
- Step 1:A →aA/bA/aB, B→∈

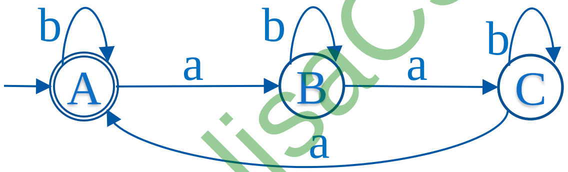


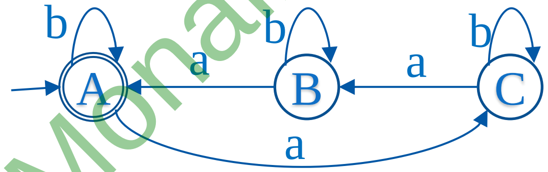


- RLG:
- B→aA
- A →aA/bA/∈

- Ex2:
- A →Ba/Ab/∈
- B →Ca/Bb
- C →Aa/Cb

- Step 1:
- A →aB/bA/ ∈
- B →aC/bB
- C →aA/bC





- RLG:
- A →aC/bA/ ∈
- B→ aA/bB
- C →aB/bC

- Ex3:
- S→Sa/Sb/ ∈
- Step 1:
- S→aS/bS/ ∈

- Shortcut:
- Consider indegree.
- ∈ production is for Initial state.
- Start symbol is final state.

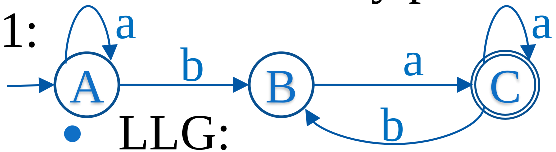
Conversion of Finite Automata → Left linear Grammar :

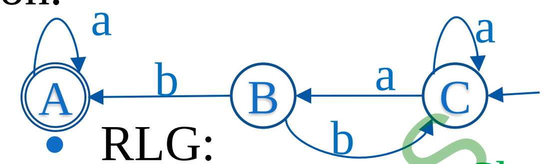
- Find Reverse of FA.
- Obtain RLG.
- Reverse RHS of every production.

FA → FA^R → RLG → LLG

L → L^R → L^R → L

- Ex1:



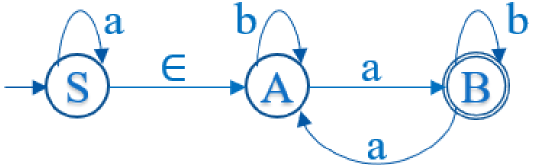


- LLG:
- C→Ca/Ba
- B →Ab/Cb
- A →Aa/ ∈

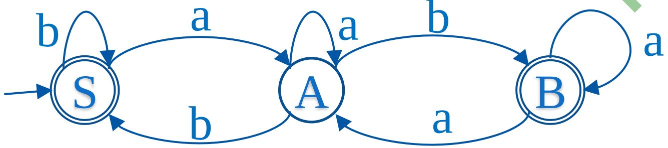
- RLG:
- C→aC/aB
- B →bA/bC
- A →aA/ ∈
- LLG:
- B→Bb/Aa
- A→Ba/Ab/S
- S→Sa/ ∈

- Shortcut:
- Consider indegree.
- Add ∈ production for Initial state.
- Final state is Start symbol .

- Ex2:

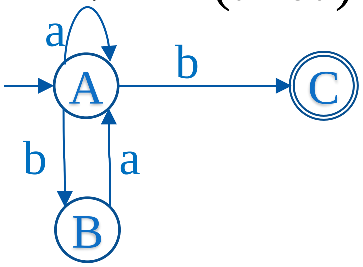


- Ex3:



- LLG:
- X→S/B
- B →Ba/Ab
- A →Aa/Sa/Ba
- S→Aa/Sb/ ∈

- **RG→RE:** RE of Start symbol is RE for Grammar.
- A→ αA/β,RE= $\alpha^*\beta$
- A→A α/β,RE= $\beta\alpha^*$
- Ex1:A →01A/00 , RE=(01)*00
- Ex2:A → A10/11, RE=11(10)*
- Ex3:S →01A/11B , RE=RE(S)=01(11)*0+11(01)*10
- A →11A/0 , RE(A)=(11)*0
- B →01B/10 , RE(B)=(01)*10
- Ex4:S→ Ab , RE=RE(S)=(a+b)*b
- A →aA/bA/∈ , RE(A)=(a+b)*
- **RE→RG:**
- Ex1:RE=ab*,RLG=A →aB,B →bB/∈ ,LLG=B → Bb/a
- Ex2: RE=(a+ba)*b



- RLG:
- A→aA/bB/bC
- B → aA
- C → ∈

- LLG:
- C→Ab
- B →Ab
- A →Ba/Aa/∈

https://monalisacs.com/

- **Simplification of CFG:**
- The process of detection and elimination of useless symbol ,Unit production and null production is called simplification of CFG
- <u>Useless Symbol:</u>
- The variable or non terminal which not involve in derivation of any string called useless symbol.
- Ex 1:
- S→aS/A
- A →b
- B → a

➢ S→aS/A
➢ A →b

- Ex 2:
- S→aS/A/bB
- A →b

➢ S→aS/A
➢ A →b

- Ex 3:
- S→aA/bBC
- A → aB/bA/a
- B →BA/bC/b

➢ S→aA
➢ A → aB/bA/a
➢ B →BA/b

- The variable that can't reach from start symbol of grammar is useless symbol.
- The variable which is reachable from start symbol but doesn't derive any terminal is useless symbol.
- The Grammar which is free from Useless symbol is reduced CFG.
- Ex 4:
- S→aB/BC/Ab
- A → bB/aA/b
- B →BC/aC
- C →CaB/CA

➢ S→Ab
➢ A →aA/b

https://www.youtube.com/@MonalisaCS

- Unit Production:
- The production of the form A→B, Where A,B∈V is called Unit production.
- Remove the Unit production and replace the equal derivation.
- Ex 1:
- S→aA
- A →bA/B/b
- B → a
  - ➢ S→aA
  - ➢ A →bA/a/b
- Ex 2:
- S→XaY
- X →Y/b
- Y →X/a
  - ➢ S→XaX
  - ➢ X →a/b
- Ex 3:
- A →BaX
- B → X/aB/b
- X →Y
- Y →Z/c
- Z →d
  - ➢ A →Bac/Bad
  - ➢ B → c/d/aB/b
- Null Production:
- The production of the form A→∈, where A∈V is called Null production.
- Remove the Null production and replace the equal derivation.
- Ex 1:
- A→XaY
- X→bX/∈
- Y→cY /∈
  - ➢ A→XaY/Xa/aY/a
  - ➢ X→bX/b
  - ➢ Y→cY /c
- Ex 2:
- A→BC
- B →aB/ ∈
- C →bC/ ∈
  - ➢ A→BC/B/C/ ∈
  - ➢ B →aB/a
  - ➢ C →bC/ b
- If a language generating ∈ then we can't remove it.
- Order of Simplification process:
- 1.Remove Null/ ∈ production.
- 2.Remove Unit Production.
- 3.Elimination of Useless Symbol

- **Normalization of CFG:**
- The process of removing redundant production from CFG is called Normalization.
- The CFG can be normalized by converting into CNF or GNF .
- Grammar should be free from null production before normalization.
- Chomsky normal form (CNF) :
- The grammar is in CNF if every production in the form V →VV/T
- Ex 1:S→aSb/∈
- Remove null production
- S→aSb /ab
- Ex 2:
- S→aSa/bSb/∈
- Remove null production
- S →aSa/bSb/aa/bb
- Ex 3:
- S→aA/Bb
- A →aAb/b
- B →bB/b

- CNF 1:
- S→ AC/AB
- C →SB
- A→a ,B→b
- CNF 2:
- S→ AC/BD/AA/BB
- C→SA , D→SB
- A →a ,B →b
- CNF 3:
- S→XA/BY
- X→a , Y→b
- A →XZ/b ,Z →AY
- B →YB/b

- Greibach normal form(GNF):
- The grammar is said to be in GNF if every production is in the form V →TV*
- Ex 1:S→aSb/∈
- Remove null production
- S→aSb /ab
- Ex 2:
- S→aSa/bSb/∈
- Remove null production
- S →aSa/bSb/aa/bb
- Ex 3:
- S→aSa/Bb
- A →bBa/bA/a
- B → aB/b
- Ex 4:
- S→aA/Bb
- A →bAa/b
- B →Bb/a

- Replace B with its production
- S→aSa/aBb/bb
- A →bBa/bA/a
- B → aB/b
- Convert B →Bb/a to right recursive
- S→aA/aB'b/ab
- A →bAa/b
- B →aB'/a
- B'→bB'/b

> GNF 1:
> S→aSB/aB
> B→b

> GNF 2:
> S→aSA/bSB/aA/bB
> A →a ,B →b

> GNF 3:
> S→aSX/aBY/bY
> X→a ,Y →b
> A→bBX/bA/a
> B →aB/b

> GNF 4:
> S →aA/aB'Y/aY
> A →bAX/b
> B →aB'/a
> B'→ bB'/b
> X →a, Y →b

- If A →Aa/b Left Recursive
- Then A →bA'
- A'→aA'/∈ Right Recursive

- Number of derivation to generate a string of length n in CNF=2n-1,GNF=n

- Ex 1:S→aSb/∈ ,w=aabb,|w|=4
- CNF 1:
- S→ AC/AB
- C →SB
- A→a ,B→b
- GNF 1:
- S→aSB/aB
- B→b
- Ex 2:
- S→aSa/bSb/∈
- W=abba, |w|=4
- CNF 2:
- S→ AC/BD/AA/BB
- C→SA , D→SB
- A →a ,B →b
- GNF 2:
- S→aSA/bSB/aA/bB
- A →a ,B →b

- Derive from CNF:
- S→AC
- →aC
- →aSB
- →aABB
- →aaBB
- →aabB
- →aabb

- Derive from GNF:
- S→aSB
- →aaBB
- →aabB
- →aabb

- Derive from CNF:
- S→AC
- →aC
- →aSA
- →aBBA
- →abBA
- →abbA
- →abba

- Derive from GNF:
- S→aSA
- →abBA
- →abbA
- →abba

- **Decision Properties of CFG**:
- CFG is decidable for Emptiness , Finiteness, Membership, Equality of CFG of DCFL.
- CFG is undecidable for ambiguity , equality, Regularity of CFG.
- Emptiness:
- Convert the grammar into reduced CFG.
- If the reduced CFG generate at least one string then the Grammar generate non empty language else generate empty language.
- Ex 1:
- S→aAB/∈
- A →Bb/AB/a
- B →BA
  - S→ ∈
  - A→a
  - S→ ∈
- Non Empty language as|∈|=1
- Ex 2:
- S→AaB/Aa
- A →bA/ ∈
  - S→Aa/a
  - A→bA/b
- Non Empty language as RE(S)=b*a
- Ex 3:
- S→aAB
- A →bA/a
- Empty language

- <u>Finiteness:</u>
- If one Grammar if Recursive then Infinite Language .If non recursive then finite language.
- <u>Membership:</u>
- Membership is a property to verify the string is generated by a Grammar or not.
- 1.Try to derive the string from Grammar .If it can generate then the string is member of that Grammar
- 2.Convert the Grammar into CNF . Apply CYK(Cocke–Younger–Kasami) algorithm
- CYK uses dynamic programming
- Ex :S →AB
- A →BA/SA/b
- B →BB/BS/a

w=ab

a	b
B	A
A	

w=ba

b	a
A	B
S	

w=aba

a	b	a
B	A	B
A	S	
B,S		

w=abab

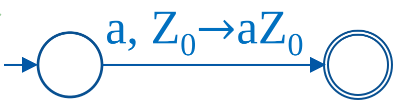
a	b	a	b
B	A	B	A
A	S	A	
B,S	A		
A			

w=abaa

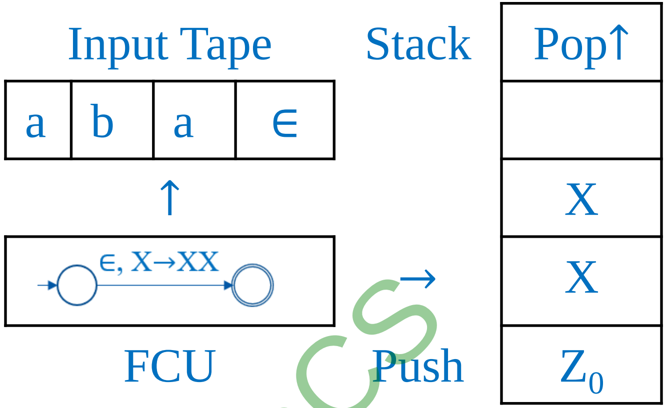
a	b	a	a
B	A	B	B
A	S	B	
B,S	S		
B,S			

Push Down Automata

- CFL can be represented by CFG or PDA.
- Mathematical representation of CFL is called as PDA/FA with one stack.
- PDA has 7 tuple $M=(Q,\Sigma,\delta,\Gamma,Z_0,q_0,F)$
- Q:Set of States.
- Σ:Set of input alphabet.
- **δ**:Transition function where $\delta: Q\times\Sigma\times \Gamma \rightarrow Q\times \Gamma^*$
- Γ:Set of Stack Symbol.
- Z_0:Top most symbol of stack.
- q_0:Initial state
- F:Set of Final State.
- **Instantaneous Description (ID):**
- ID describe the movements of PDA.
- The movement of PDA depends on 3 entity.
- Current State,Current Input Symbol,Current topmost symbol of stack
- $\boldsymbol{\delta}\ (q_i,a, Z_0)=(q_j,aZ_0)$

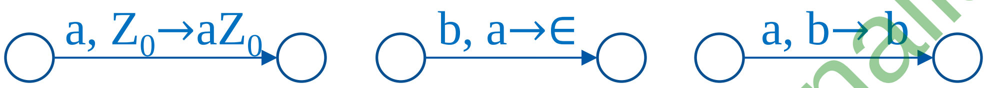


- **Block diagram of PDA**
- PDA consist of 4 component
- 1.Input Tape
- 2.Tape Header
- 3.Finite control Unit,
- 4.Stack



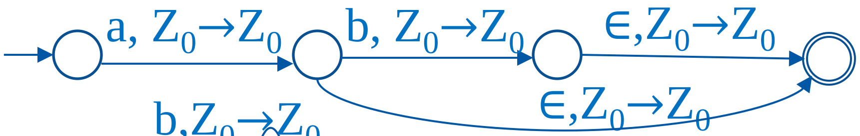
- FA+Stack=PDA
- PDA uses stack as external storage location.
- E(PDA)=2
- PDA can accept RL and CFL.
- PDA is of two types : DPDA & NPDA
- DPDA= $\delta: Q\times(\Sigma \cup \in)\times \Gamma \rightarrow Q\times \Gamma^*$
- NPDA= $\delta: Q\times(\Sigma \cup \in)\times \Gamma \rightarrow 2^{(Q\times\Gamma^*)}$
- Every DPDA is NPDA but no algorithm exist to convert NPDA to DPDA.
- NPDA is more powerful than DPDA.
- DPDA is more efficient than NPDA.
- In general the PDA is NPDA.
- L(DPDA)⊂L(NPDA)

- **Acceptance of PDA:**
- PDA can accept string in 2 way
- 1.Acceptance by empty stack : After reading complete input string if the stack is empty then the input string is accepted by PDA .
- 2.Acceptance by Final State : After reading complete input string if the PDA reaches the Final state .
- **Operation in PDA:**
- 1.Push, 2.Pop, 3.Skip

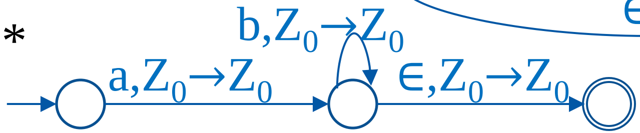


- Every transition on PDA associated with any one operation.
- PDA=RL if every transition is skip operation.
- PDA=NRL if at least one operation represent push operation.
- PDA can accept RL using final state mechanism.
- PDA can accept Non RL either by using empty stack mechanism or final state mechanism.

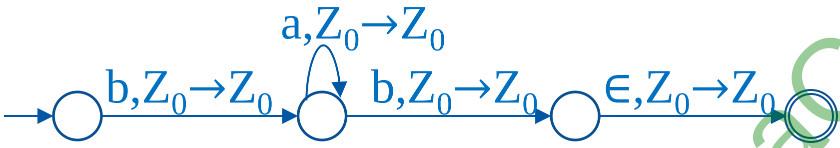
- **Construct PDA for following Regular Language, $\Sigma=\{a,b\}$**
- $L_1=\{a,ab\}$



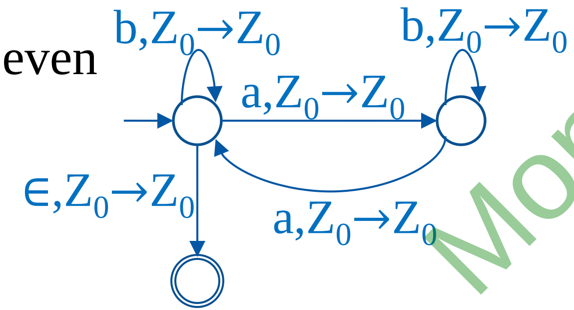
- $L_2=ab^*$



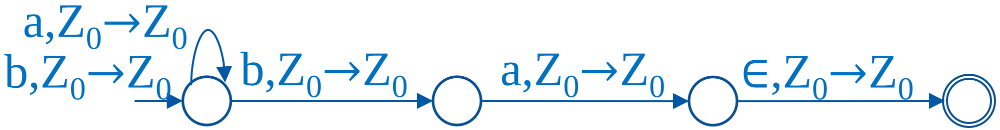
- $L_3=ba^*b$



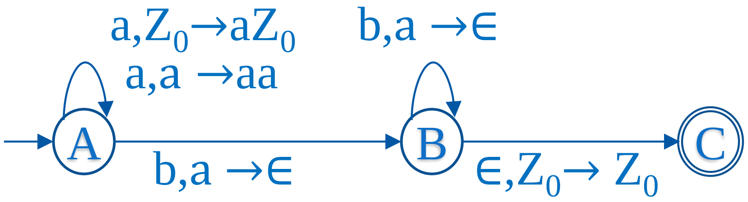
- $L_4=n_a(w)=\text{even}$



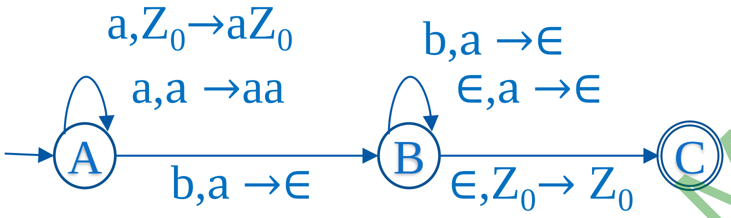
- L_5={Every string ends with ‘ba’}



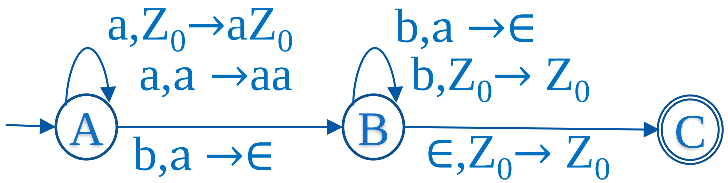
- **Construct PDA for following Context Free Language**
- $L_1=\{a^m b^n / m,n \geq 1, m=n\}$ or $\{a^n b^n / n \geq 1\}$
- $=\{ab, aabb, aaabbb, \ldots\ldots\}$



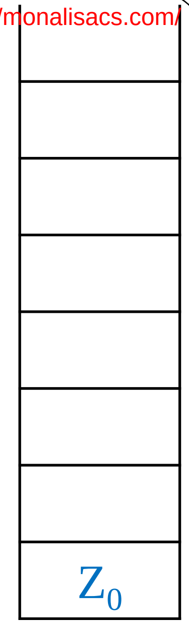
- $L_2=\{a^m b^n / m,n \geq 1, m \geq n\}=\{aab, aaabb, aaaabb, ..\}$



- $L_3=\{a^m b^n / m,n \geq 1, m \leq n\}=\{abb, abbb, aabbb, \ldots\}$

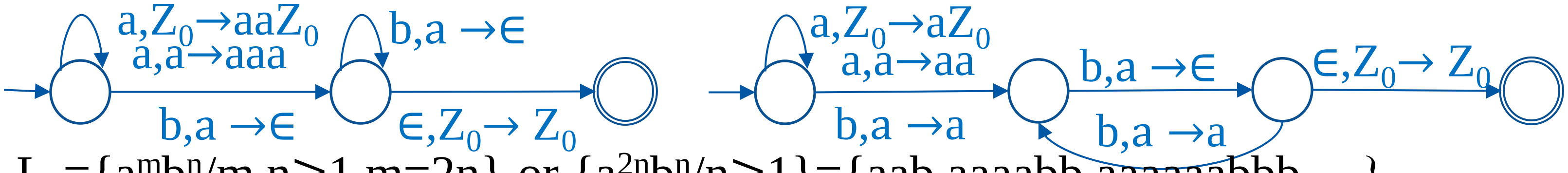


- Final State vs Empty Stack
- If $m,n \geq 0$
- **Transition Function**
- $\delta(A,a,Z_0)=(A,aZ_0)$
- $\delta(A,a,a)=(A,aa)$
- $\delta(A,b,a)=(B,\in)$
- $\delta(B,b,a)=(B,\in)$
- $\delta(B,\in,Z_0)=(C, Z_0)$

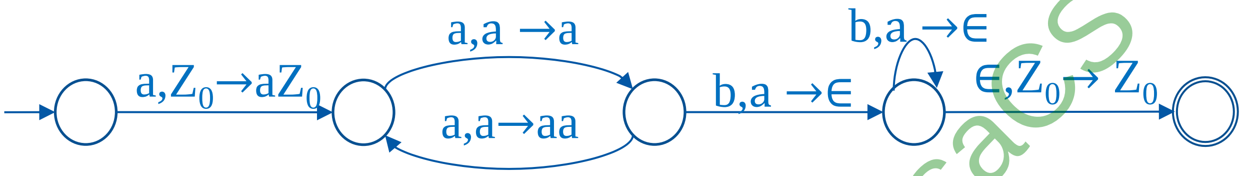


Stack

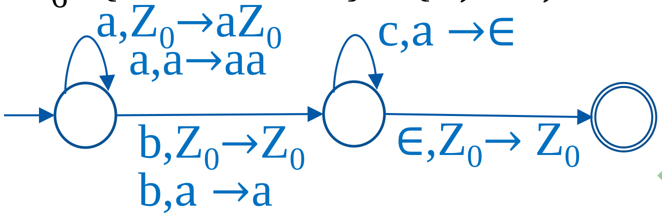
- $L_4=\{a^mb^n/m,n\geq 1,2m=n\}$ or $\{a^nb^{2n}/n\geq 1\}=\{abb,aabbbb,aaabbbbbb,\ldots\}$



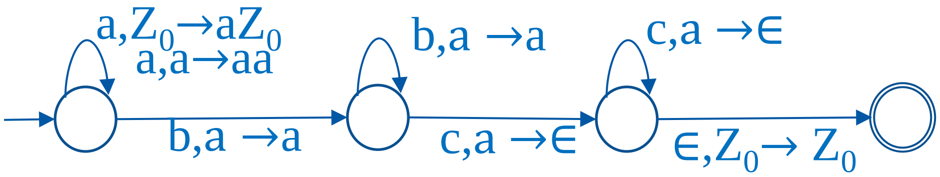
- $L_5=\{a^mb^n/m,n\geq 1,m=2n\}$ or $\{a^{2n}b^n/n\geq 1\}=\{aab,aaaabb,aaaaaabbb,\ldots\}$

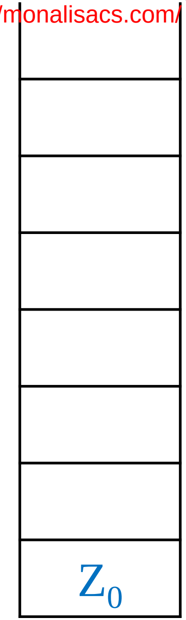


- $L_6=\{a^nbc^n/n\geq 0\}=\{b,abc,aabcc,aaabccc,\ldots\}$



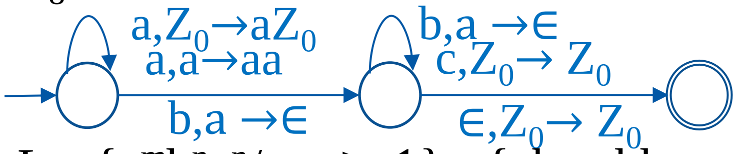
- $L_7=\{a^mb^nc^m/m,n\geq 1\}=\{abc,aabbbcc,aaabbccc,\ldots\}$



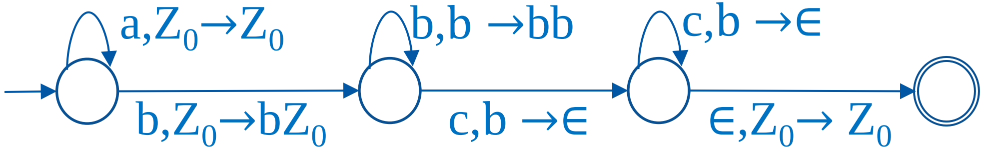


Stack

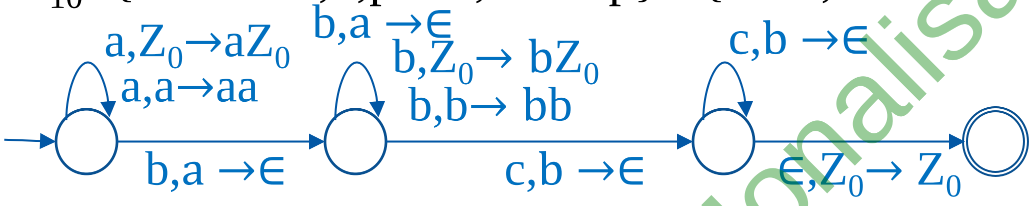
- $L_8=\{a^m b^m c^n / m,n \geq 1\}$ =\{abc,aabbc,aaabbbcccccc,.....\}



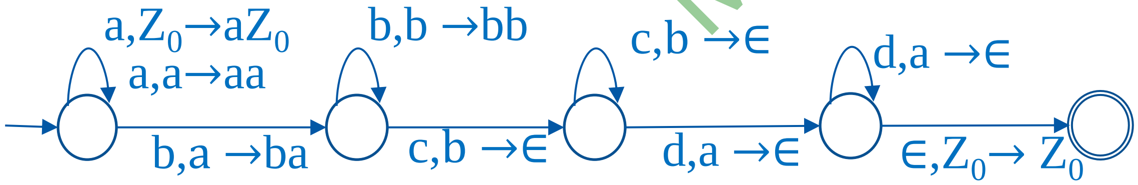
- $L_9=\{a^m b^n c^n / m,n \geq 1\}$ =\{abc,abbcc,aaaaabbbccc,.....\}

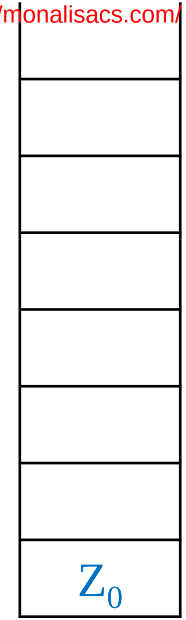


- $L_{10}=\{a^m b^n c^p / m,n,p \geq 1, n=m+p\}$ =\{abbc,aa bb bbb ccc ,.....\}= $a^m b^m b^p c^p$



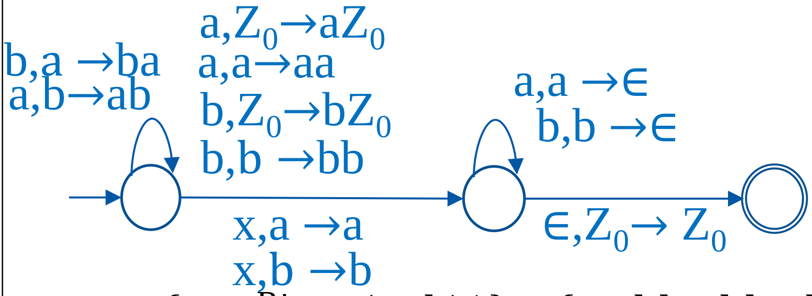
- $L_{11}=\{a^m b^n c^n d^m / m,n \geq 1\}$ =\{abcd,abbccd ,aabcdd.....\}



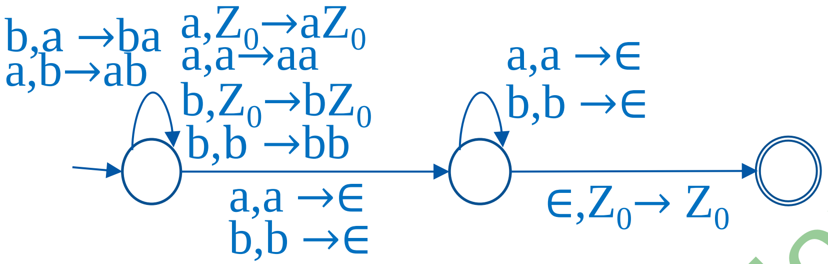


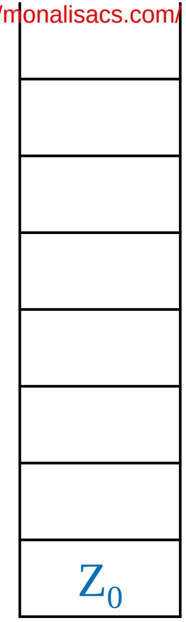
Stack

- $L_{12}=\{wxw^R | w\in(a+b)^+\}$ =\{axa,bxb,abxba,baaxaab,.....\}



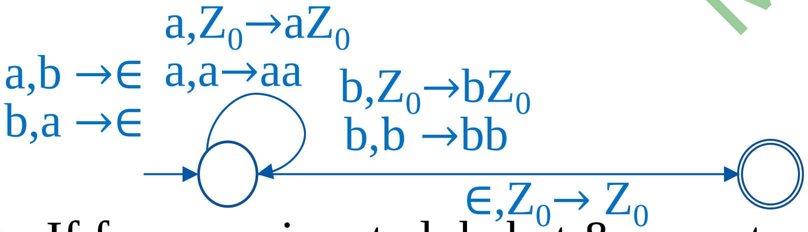
- $L_{13}=\{ww^R | w\in(a+b)^+\}$ =\{aa,bb,abba,baaaab,.....\}Even palindrome





Stack

- $L_{14}=\{n_a(w)=n_b(w) | w\in(a+b)^*\}$ =\{∈,ab,ba,abba,bbaa,baab,.....\}



- If for same input alphabet & same top most symbol of stack there are more than one transition then it's a NPDA.

- **DPDA** :The PDA is said to be deterministic if every $\delta(q,a,x)$ has at most one outcome for all $a \in \Sigma$ or $a = \in, x \in \Gamma$.
- **DCFL**: The Language accepted by DPDA is called DCFL.
- Ex: $\{a^n b^n / n \geq 1\}, \{a^m b^n c^m / m,n \geq 1\}$, $\{wxw^R | w \in (a+b)^+\}$, $\{n_a(w)=n_b(w) | w \in (a+b)^*\}$
- Every RL is DCFL ,But DCFL need not be Regular.
- Every language accepted by DPDA is accepted by NPDA.Every DCFL=CFL.
- **Closer Property of DCFL:**
- DCFL is closed under Complement , Inverse homeomorphism , Quotient with RL, Intersection with RL, Difference with RL
- DCFL is not closed under Union, Concatenation ,Intersection , Kleene Closer, Substitution ,Homeomorphism ,Reversal ,Quotient
- **NPDA**:The PDA is said to be Non deterministic if every $\delta(q,a,x)$ has more than one outcome for all $a \in \Sigma$ or $a = \in, x \in \Gamma$.
- If for same input alphabet & same top most symbol of stack there are more than one transition then it’s a NPDA.
- **CFL:**The Language accepted by NPDA is called CFL.
- Ex: $\{ww^R | w \in (a+b)^+\}$, $\{waw^R | w \in (a+b)^+\}$, $\{wbw^R | w \in (a+b)^+\}$

- **Closer Property of CFL:**
- CFL is closed under following operation
- Union ,Concatenation, Kleene Closer ,Substitution ,Homomorphism ,Inverse Homomorphism ,Reverse ,
- Intersection with RL, Quotient with RL.
- CFL is not closed under following operation.
- Complement , Intersection , Difference ,Symmetric difference, Quotient
- $L_1=\{a^m b^n c^p | m=n\}$CFL
- $L_2=\{a^m b^n c^p | n=p\}$CFL
- $L_1 \cap L_2=\{a^m b^n c^p | m=n=p\}$CSL
- $\overline{L_1}$=CSL
- CFL∩RL=CFL

Pumping Lemma

- If L is any CFL.There exists a pumping length n s.t for every string $w \in L, |w| \geq n$.
- We can break w into 5 strings ,w=uvxyz
- 1.$vy \neq \varepsilon$
- 2.$|vxy| \leq n$
- 3.$uv^kxy^kz \in L, \forall k \geq 0$
- Pumping lemma is used to prove that some of the language is not CFL.
- If there exist at least one k for which $uv^kxy^kz \notin L$ then L is not CFL.
- Ex 1:Prove that $L=\{a^nb^nc^n \mid n \geq 1\}$ is not CFL
- Let w= aabbcc $\in$L ,Pumping Length=3, $|w| \geq n=6 \geq 3$
- u=a,v=a,x=bb,y= ε,z=cc
- $vy=a\epsilon=a \neq \varepsilon$
- $|vxy|=3 \leq 3$
- Now check $uv^kxy^kz \in L, \forall k \geq 0$
- Let k=0, uv^0xy^0z =a a^0 bb ε^0 cc =abbcc $\notin$ L
- Let k=1, uv^1xy^1z =a a^1 bb ε^1 cc =aabbcc $\in$ L
- Let k=2, uv^2xy^2z =a a^2 bb ε^2 cc =aaabbcc $\notin$ L
- So L is not CFL.

- Ex 2:Prove that $L=\{ww \mid w \in (a+b)^+\}$ is not CFL
- Let w= abb abb $\in L$, Pumping Length=2, $|w| \geq n=6 \geq 2$
- $u=ab, v=b, x=\varepsilon, y=a, z=bb$
- $vy=ba \neq \varepsilon$
- $|vxy|=2 \leq 2$
- Now check $uv^kxy^kz \in L, \forall k \geq 0$
- Let k=0, $uv^0xy^0z = ab\ b^0\ \varepsilon\ a^0\ bb = ab\ bb \notin L$
- Let k=1, $uv^1xy^1z = ab\ b^1\ \varepsilon\ a^1\ bb = abb\ abb \in L$
- Let k=2, $uv^2xy^2z = ab\ b^2\ \varepsilon\ a^2\ bb = abbb\ aabb \notin L$
- So L is not CFL.

- **Weak form of Pumping lemma**
- If L is any language defined over the alphabet Σ with only one symbol s.t. the length of string of L are in some AP then L is CFL
- $L_1=\{a^{2n}|n\geq 0\}$CFL [0,2,4,6…....AP]
- $L_2=\{a^{3n+2}|n\geq 0\}$CFL [2,5,8,11…..AP]
- $L_3=\{a^{2n-5}|n\geq 3\}$CFL [1,3,5,7, …..AP]
- $L_4=\{a^{n^2}|n\geq 0\}$NCFL [0,1,4,9,…..Not in AP]
- $L_5=\{a^{2^n}|n\geq 0\}$NCFL [1,2,4,8, ….Not in AP]
- $L_6=\{a^{n!}|n>0\}$NCFL [1,2,6,24,…Not in AP]
- $L_7=\{a^p|p$ is a +ve prime$\}$NCFL [1,3,5,7,11, ….Not in AP]
- Lets check L_1 by pumping Lemma.
- W=aaaa ,u=a,v=a,x=a,y= ε,z=a
- Now check $uv^kxy^kz\in L, \forall k\geq 0$
- Let k=0, uv^0xy^0z =a a^0 a ε^0 a =aaa $\notin$ L
- Let k=2, uv^2xy^2z =a a^2 a ε^2 a =aaaaa $\notin$ L
- So L is NCFL.