Unit - III

Context Free Grammer (CFG)

context free Grammer can be formally defined as set denoted by $G = (V, T, P, S)$ where V, T are set of non-terminals & terminals respectively, P is set of production rules where each production rule is in the form of nonterminal → terminal (or) nonterminal → nonterminal (In General non terminals are denoted by capital letters and terminals are denoted by lower cases).
S is start symbol

Ex:- 1. $P = \begin{Bmatrix} S \to S+S \\ S \to S*S \\ S \to id \end{Bmatrix}$

from the above production rules $V = \{S\}$, $T = \{id, +, *\}$, $S = S$.

2. $P = \begin{Bmatrix} S \to AaB \\ A \to aA / \epsilon \\ B \to bB / \epsilon \end{Bmatrix}$

from the above production rules $V = \{S, A, B\}$, $T = \{a, b, \epsilon\}$, $S = S$

1. Construct context free Grammer for the following language.

$L = \{\epsilon, a, aa, aaa, \dots\}$ over alphabet $\Sigma = \{a\}$.

$S \to aS \mid a^*$ a^+

context free Grammer for the above language is $S \to as / \epsilon$

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

$S \to as \mid \epsilon$ a^*

2. construct context free Grammer for the any no. of a's (or) any no. of b's over $\Sigma = \{a, b\}$

$(a+b)^*$

$S \to as / bS / \epsilon$

where $V = \{S\}$

$T = \{a, b\}$

$S = S$

construct CFG for $L=\{aa, ab, ba, bb\}$ over $\Sigma=\{a, b\}$.

context free Grammer for the language is

$S \rightarrow AA$

$A \rightarrow a/b$

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

construct CFG for string start with a and ending with b.

context free Grammer for the language is

$S \rightarrow aAb$

$A \rightarrow aA/bA/\epsilon$

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

Construct CFG for $[(a+b)(a+b)]^*$

context free Grammer for the language is

$S \rightarrow AS/\epsilon$

$A \rightarrow BB$

$B \rightarrow a/b$

(or)

$S \rightarrow AASL\epsilon$

$A \rightarrow a/b$

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

construct CFG for $[(a+b)^*(a+b)]^*$

context free Grammer for the language is

$S \rightarrow AS/\epsilon$

$A \rightarrow BC$

$B \rightarrow BB/bB/\epsilon$

$C \rightarrow a/b$

$V=\{S, A, B, C\}$, $T=\{a, b\}$

construct CFG for the language $L=\{a^n b^n / n \geq 0\}$

CFG for the language.

$S \rightarrow aSb/\epsilon$.

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

Construct CFG for $L=\{a^n b^n \mid n \geq 1\}$

CFG for language is $S \rightarrow aSb \mid ab$

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

construct CFG for $L=\{\underbrace{a^n b^n}_{A} \underbrace{c^m}_{B} / m,n \geq 0\}$

CFG for Language is

$S \rightarrow AB$

$A \rightarrow aAb \mid \epsilon$

$B \rightarrow cB \mid \epsilon$

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

construct CFG for the language $L=\{a^n b^m c^n / n,m \geq 0\}$

CFG for language is

~~$S \rightarrow aAc / aSc / \epsilon$~~

~~$A \rightarrow bAc / \epsilon$~~

~~$B \rightarrow cBc / \epsilon$~~

(or)

$S \rightarrow aSc / A \mid \epsilon$

$A \rightarrow bA \mid \epsilon$

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

construct CFG for the language $L=\{a^n b^{2n} / n \geq 0\}$

CFG for language is

$S \rightarrow aSbb \mid \epsilon$

$aS/bS/\epsilon$

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

construct CFG for $L=\{a^n b^m / n,m \geq 0\}$

CFG for language is

$S \rightarrow AB$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

$V=\{S, A, B\}$ $T=\{a,b\}$.

construct CFG for $L = \{a^n b^n c^m d^m / m,n \geq 0\}$

CFG for language is

$S \rightarrow AB$

$A \rightarrow aAb / \epsilon$

$B \rightarrow cBd / \epsilon$

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

construct CFG for $L = \{a^n b^n c^n / n \geq 0\}$

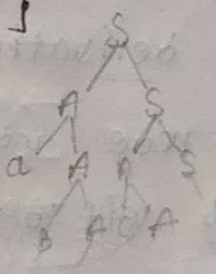
CFG for Language is

$S \rightarrow$

construction not possible.

$S \rightarrow AS / \epsilon$

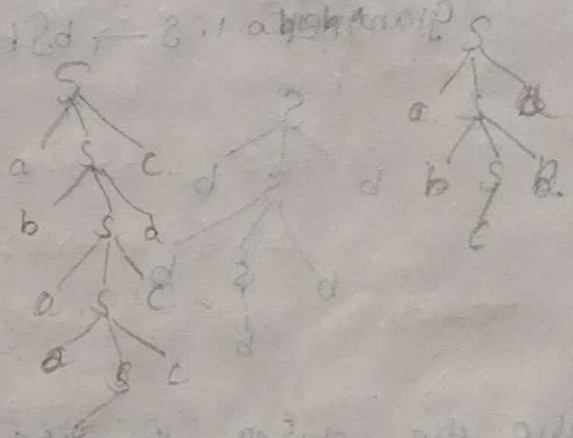
$A \rightarrow aA / bA / cA / \epsilon$



construct CFG for $L = \{a^n b^m c^n d^m / m,n \geq 0\}$.

$S \rightarrow aSc / bSd / \epsilon$

Impossible to Construct



construct CFG for $L = \{WcW^T / (a,b)^*\}$

CFG for language is

$S \rightarrow aSa / \epsilon$

$S \rightarrow bSb / \epsilon$

$S \rightarrow c / \epsilon$

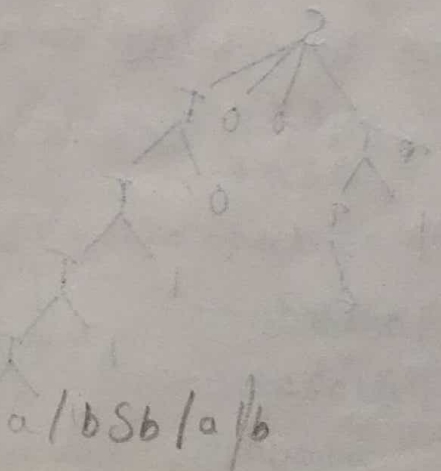
$V = \{S\}$ $T = \{a, b, c\}$

construct CFG for $L = \{ww^R / w \in (a,b)^*\}$

$S \rightarrow aSa / bSb / \epsilon$ → even palindrom. → palindrom.

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

odd palindrom → $S \rightarrow aSa / bSb / a / b$



Derivation Tree

It is a graphical representation for derivation of given production rules for the given context free Grammer.

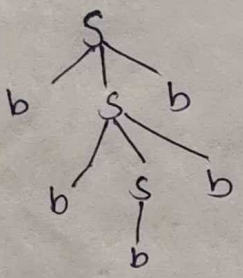
It is a simple way to show how the derivation can be done to obtain some string from given set of production rules.

It is also called as parsetree.

follow are properties of derivation tree:

1. Root node is always a node indicates start symbol.
2. Derivation is read from left to right leaf nodes are always terminal nodes and interior nodes are always non-terminal nodes.

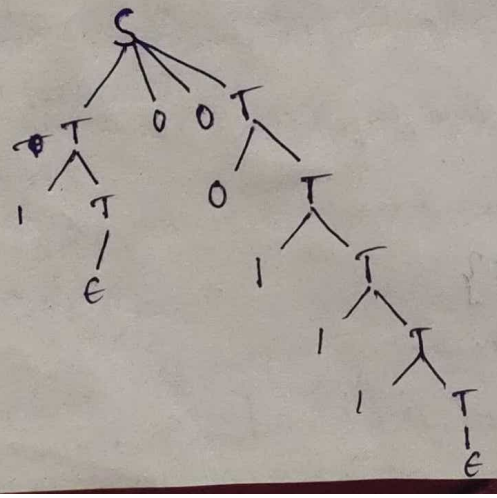
Ex:- consider the following Grammer derive the string "bbbbb" for the Grammer 1. $S \rightarrow bSb/a/b$ where 'S' is start symbol.

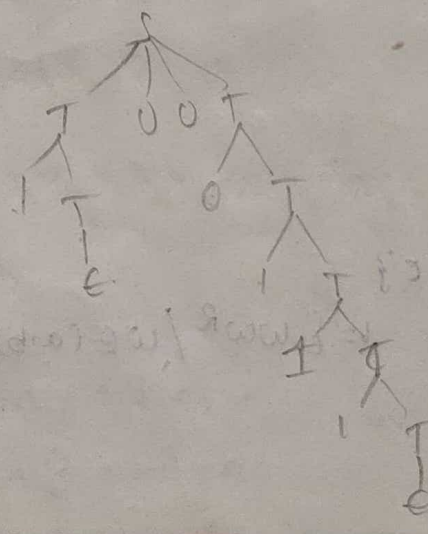


Derive the string from the following condition Grammer.

(i) 1000111 $S \rightarrow T00T$

$T \rightarrow 0T/1T/\epsilon$ where S is start symbol.





Left & Right most derivation

Left most derivation :-

Derivation in which left most non-terminal is replace from sentencial form. first is known as left most derivation (LMD)

Right most derivation :-

Derivation in which Right most non-terminal is replaced from sentencial form first is known as right most derivation (RMD)

Ex:-

Derive LMD & RMD for the string 'aba'. $S \to XXX$

$X \to a$

$Y \to b$ where 'S' is start symbol.

LMD:

$S \to XYX$

$\to aYX$ [$\because X \to a$]

$\to abX$ [$\because Y \to b$]

$\to aba$ [$\because X \to a$]

RMS:

$S \to XYX$

$\to XYa$ [$\because X \to a$]

$\to Xba$ [$\because Y \to b$]

$\to aba$ [$\because X \to a$]

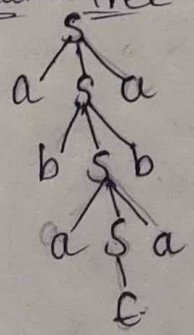
Draw a derivation tree for the string "abaaba" from context free Grammer given by $S \to aSa$

$S \to bSb$

$S \to a/b/\epsilon$

also derive LMD & RMD where 'S' is start symbol.

Derivation tree



LMD :-

$S \to aSa$

$\to aSa$

$\to abSba$ [$\because S \to bSb$]

$\to abaSaba$ [$\because S \to aSa$]

$\to aba\epsilon aba$ [$\because S \to \epsilon$]

$\to abaaba$.

RMD :-

$S \to aSa$.

$\to aSa$

$\to abSba$ [$\because S \to bSb$]

$\to abaSaba$ [$\because S \to aSa$]

$\to aba\epsilon aba$ [$\because S \to \epsilon$]

$\to abaaba$.

Derive LMD & RMD for the following string "aaabbabbba" from CFG

$S \to aB/bA$

$S \to aS/bAA/a$ where 'S' is start symbol

$B \to bS/aBB/b$

$A \to a$

LMD

$S \to aB$

$\to aB$

$\to aaBB$ [∵ $B \to aBB$]

$\to aaaBBB$ [∵ $B \to aBB$]

$\to aaabSBB$ [∵ $B \to bS$]

$\to aaabbABB$ [∵ $S \to bA$]

$\to aaabbaBB$ [∵ $A \to a$]

$\to aaabbabB$ [∵ $B \to bS$]

$\to aaabbabbS$ [∵ $B \to bS$]

$\to aaabbabbbA$ [∵ $S \to bA$]

$\to aaabbabbba$ [∵ $A \to a$]

RMD :-

$S \to aB$

$\to aB$

$\to aaBB$

$\to aaBbS$ [∵ $B \to bS$]

$\to aaBbbA$ [∵ $S \to bA$]

$\to aaBbba$ [∵ $A \to a$]

$\to aaaBBbba$ [∵ $B \to aBB$]

$\to aaaBbbba$ [∵ $B \to b$]

$\to aaabSbbba$ [∵ $B \to bS$]

$\to aaabbAbbba$ [∵ $S \to bA$]

$\to aaabbabbba$ [∵ $A \to a$]

Ambiguous Grammer :-

A Grammer $G = (V, T, P, S)$ is ~~sele~~ to be Ambiguous if there exist one or more parse / derivation tree exis for a given grammer.

i.e there could be more than one LM or RM derivations.

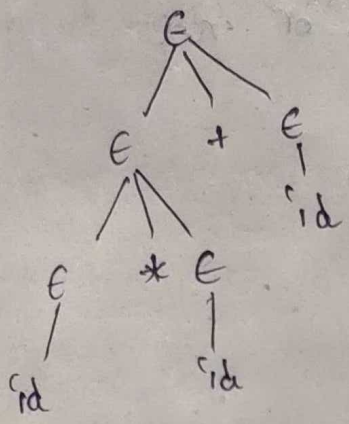
Ex :- Consider the following CFG

$E \to E+E$

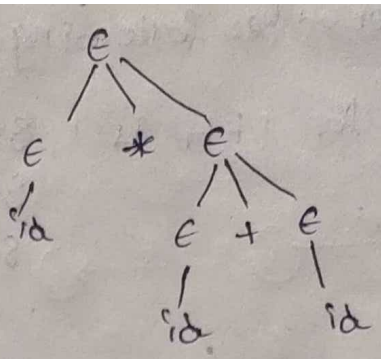
$E \to E*E$ where 'E' is start symbol.

$E \to id$

Grammer is ambiguous or not for the given string "id * id + id

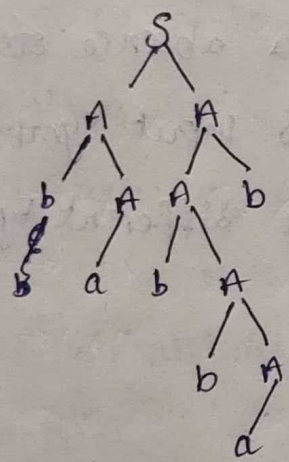


(or)

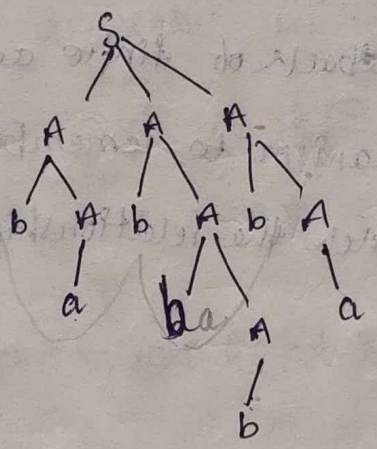


The given gramer is ambiguous grammer because there are more than one derivation tree for the string "id * id + id"

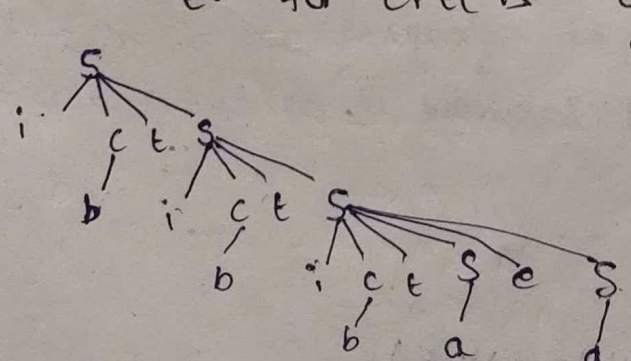
Prove the following grammer is ambiguous which accept the string "babbab" for CFG is $S \rightarrow AA$, $A \rightarrow AAA$, $A \rightarrow a/bA$, $A \rightarrow Ab$ where 'S' is start symbol.



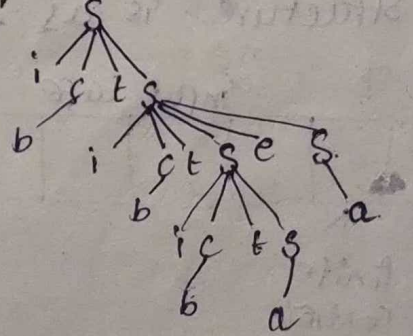
(or)



Check whether the following grammer is ambiguous or not "ibtibt ibt aba" for CFG is $S \rightarrow ictS$, $S \rightarrow ictSeS/a$, $c \rightarrow b$ where S is start symbol

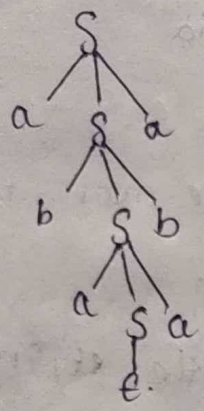


(or)



It is ambiguous.

"abaaba" for CFG is $S \rightarrow aSa$
$S \rightarrow bSb$.
$S \rightarrow a|b|\epsilon$.



It is not an ambiguous.

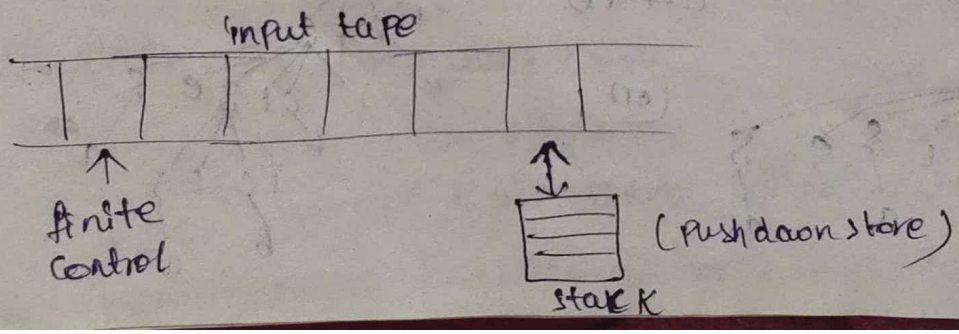
Pushdown Automata (PDA)

In pushdown Automata, automata is a tool to implement context free language (CFL). biggest drawback of finite automata is absence of Memory i.e there is no mechanism to remember count of input symbol. count of symbol refers to match the relationship between different types of symbols occur in string.

Basic structure of PDA

1. PDA will have input tape,
- read head (finite control), stack (pushdown store).

Basic structure is as follows:



*Input tape is one in which input string is place. It is divided into many cells. at each cell one input symbol is placed like wise there input symbol is placed on the tape.

*Read head

It has one point which points to the current symbol which is to be

At the end of input symbol (ϵ / \$ / Δ / ϕ) is placed which indicate the end of

*stack :- stack is also known as pushdown store which is used to store the input symbol or we can remove symbols from stack.

Formal Definition of PDA :-

PDA can be defind as 1. finite set of states represented by Q

2. finite set of input symbols represented by 'Σ'

3. Transition 'S' defined as $Q \times (\Sigma \cup \epsilon) \times \Gamma \longrightarrow Q \times \Gamma^*$ (tow)

4. Start state q_0 of read head i.e $q_0 \in Q$

5. Set of final states denoted by 'F'

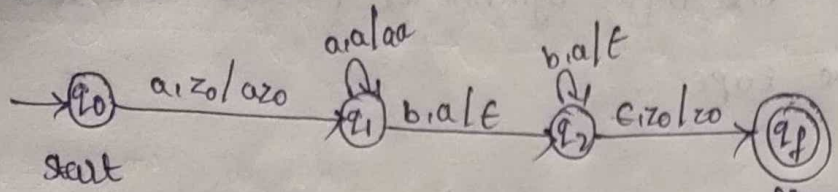
6. stack alphabet denoted by 'Γ' in which set of symbols that can be pushed on to stack.

7. small 'z_0' known as start symbol which is in 'Γ' that indicates that Stack is empty.

~~construct PDA for the language L~~

8. Seven tuple notation is denoted by $A = (Q, \Sigma, S, q_0, F, \Gamma, z_0)$

Construct PDA for the language $L = \{a^n b^n / n \geq 1\}$



$Q = \{q_0, q_1, q_2, q_f\}$

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

$\delta(q_0, a, z_0) = (q_1, az_0)$

$\delta(q_1, a, a) = (q_1, aa)$

$\delta(q_1, b, a) = (q_2, \epsilon)$

$\delta(q_2, b, a) = (q_2, \epsilon)$

$\delta(q_2, \epsilon, z_0) = (q_f, z_0)$

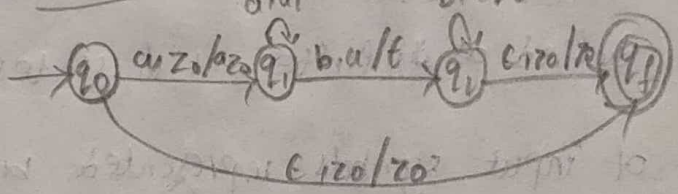
$\Gamma = \{a, z_0\}$

$q_0 = q_0$

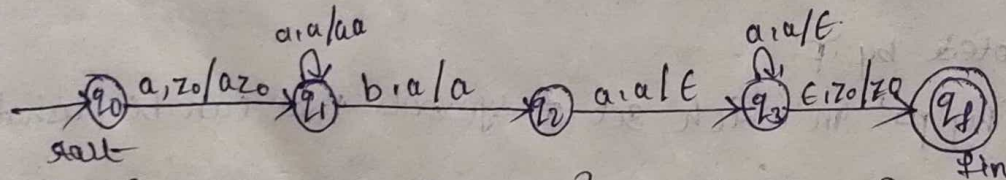
$F = q_f$

$z_0 = z_0$.

$L = \{a^n b^n / n \geq 0$



construct PDA for the language $L = \{a^n b a^n / n \geq 1\}$



$Q = \{q_0, q_1, q_2, q_3, q_f\}$ $\Sigma = \{a, b\}$

$\delta(q_0, a, z_0) = \{q_1, az_0\}$ push operation

$\delta(q_1, a, a) = \{q_1, aa\}$ "

$\delta(q_1, b, a) = \{q_2, a\}$ Read operation, (i.e no operation performed)

$\delta(q_2, a, a) = \{q_3, \epsilon\}$ POP "

$\delta(q_3, a, a) = \{q_3, \epsilon\}$ POP "

$\delta(q_3, \epsilon, z_0) = \{q_f, z_0\}$

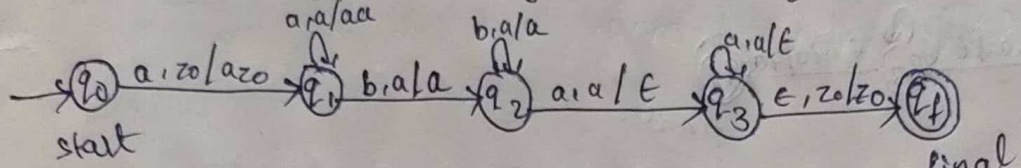
$q_0 = q_0$

$F = q_f$

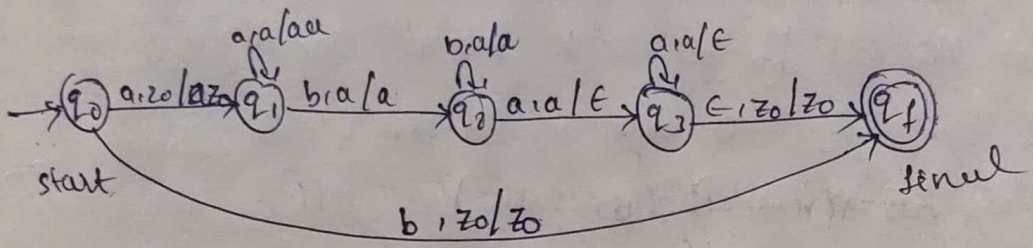
$z_0 = z_0$

$\Gamma = \{a\}$

design PDA for the language $L = \{a^n b^m a^n / m,n \geq 1\}$

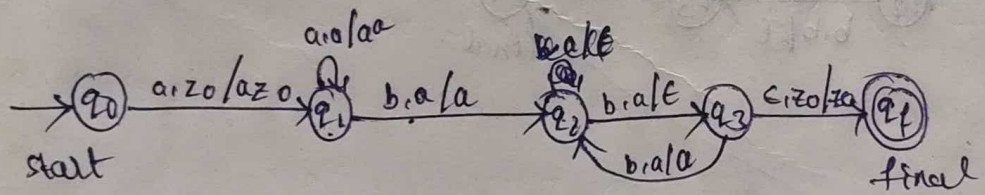


$L = \{a^n b a^n / n \geq 0\}$

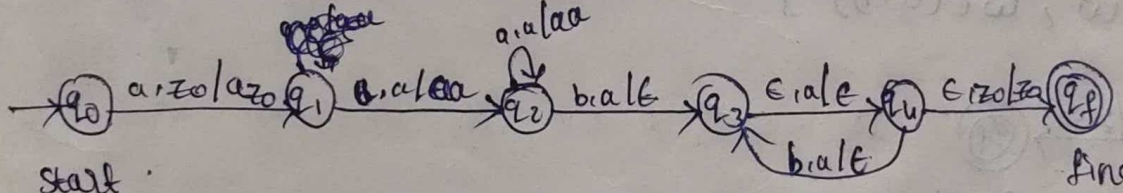


design PDA for $L = \{a^n b^{2n} / n \geq 1\}$

b — Re
2b — pop



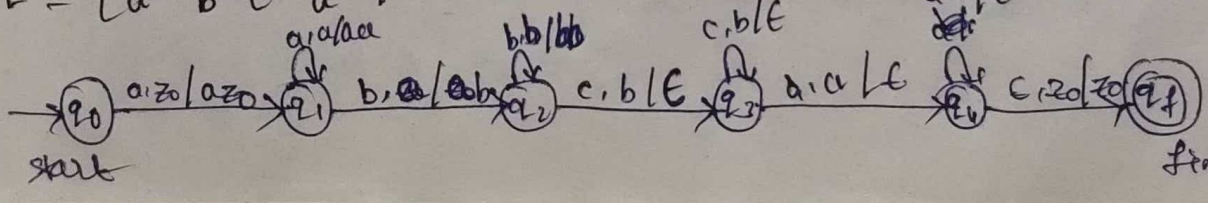
$L = \{a^{2n} \cdot b^n / n \geq 1\}$



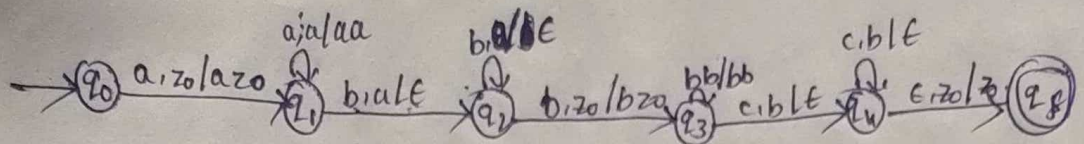
$L = \{a^n b^m c^n d^m / m,n \geq 1\}$

not possible to construct PDA.

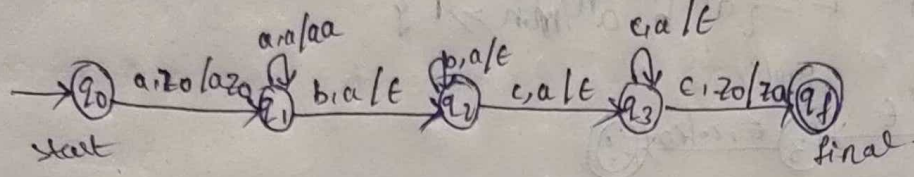
$L = \{a^n b^m c^m d^n / m,n \geq 1\}$



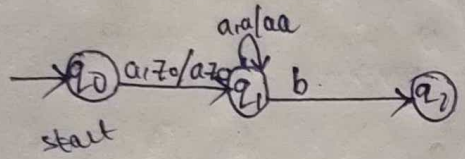
$L = \{ a^m b^{m+n} c^n / m,n \geq 1 \}$



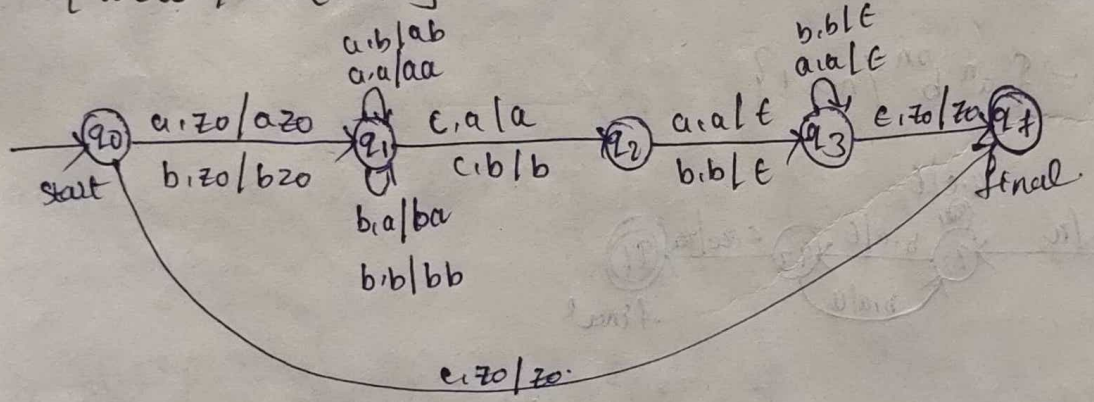
$L = \{ a^{m+n} . b^m . c^n \} / m,n \geq 1 \}$



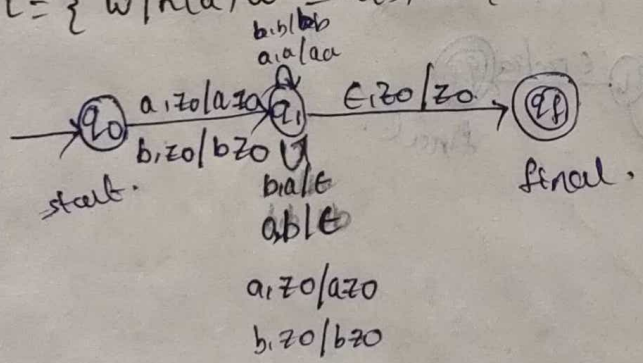
$L = \{ a^m b^n c^{m+n} / m,n \geq 1 \}$



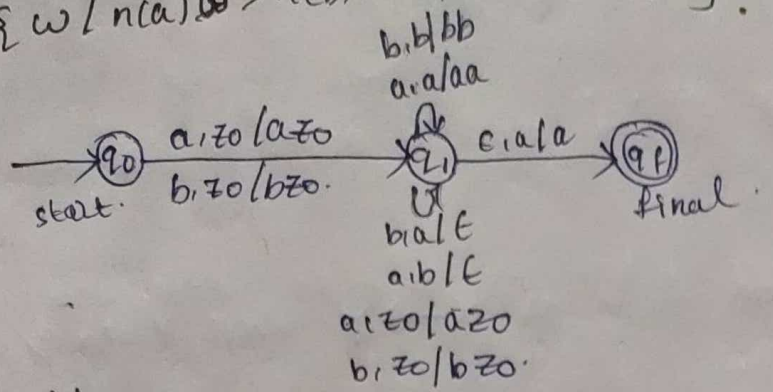
$L = \{ wcw^R / w \in (a,b)^* \}$ w^R indicates reverse of w.



$L = \{ w / n(a) w = n(b) w , w \in (a,b)^* \}$



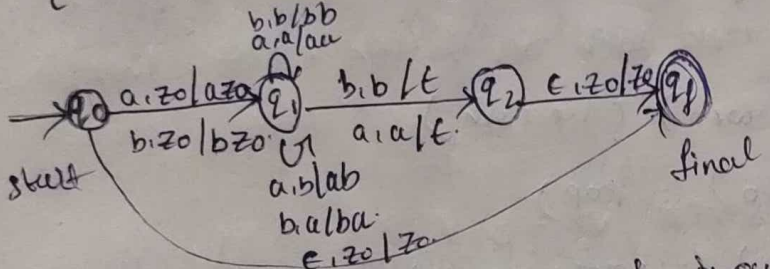
$L = \{ w / n(a) \geq n(b)\ w \ , w \in (a,b)^* \}$.



$L = \{ \omega \omega^R / \omega \in (a,b)^* \}$

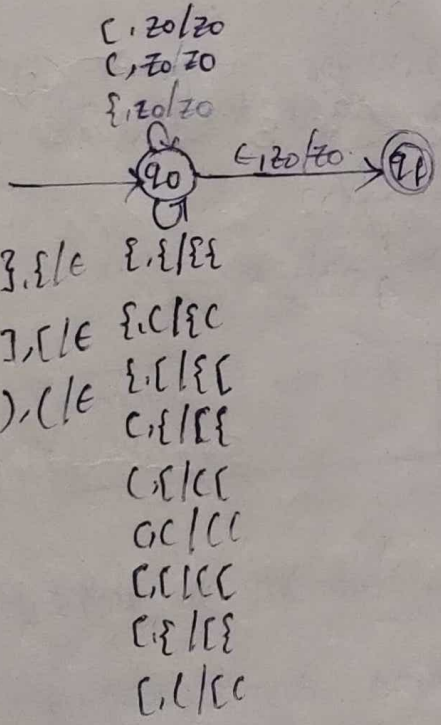
$\omega = \{ 0^i 1^j 2^k / i+j < k, \forall i,j,k \geq 1 \}$

$L = \{ \omega \omega^R / \omega \in (a,b)^* \}$



It is Non-Deterministic push down automata.

construct PDA for valid paranthesis or Dlimiter matching



H-W $L = \{ 0^m 1^n / m > n+2, m,n \geq 1 \}$

$L = \{o^{m,n} / m > n+2, m,n \geq 1\}$

Imp

Acceptance of Language by PDA :-

The Language can be accepted by PDA using two Approaches

1. Acceptance by empty state :

on reading input symbol from initial configuration for PDA. the stack gets empty.

Ex :- $(q_0, wwR, z_0) \vdash^* (q_1, wR, wz_0)$ (i.e push)

$\vdash^* (q_1, \epsilon, z_0)$ (:- POP)

$\vdash^* (q_1, \epsilon \epsilon)$ (:- empty stack)

2. Acceptance by final state :-

PDA accepts by consuming it and then understing the final state with out check i.e stack.

Ex : $(q_0, wwR, z_0) \vdash (q_1, wR, z_0)$

(q_1, ϵ, z_0)

(q_1, ϵ, z_0)

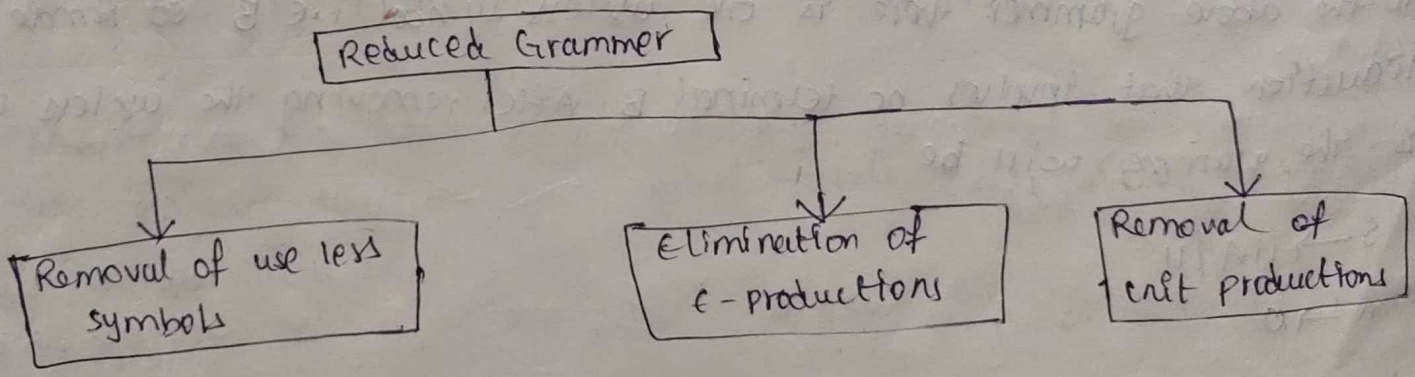
construct PDA for the language $L = \{a^n(b^m / m>n\}$ until we get the string: "aabbb"

$q_0 \rightarrow aAb$

Minimization of context free Grammer (CFG)

All grammers are not always optimized. i.e the grammer may consists of some extra symbols (non-terminals). Having extra symbols in grammer leads to increase of the length of grammer.

following diagram represents methods of Reducing Grammer:



Simplification of grammer means reduction of grammer by removing useless symbols, ε-productions & unit productions.

Properties of Reduced grammer

1. Each non-terminal & each terminal of 'G' must be appear in derivation of some word/string in 'L'
2. There should not be any production such as $X \rightarrow Y$ where X and Y are not terminals.
3. If ϵ is not in language 'L'. Then there need not be any production like $X \rightarrow \epsilon$

Removal of useless symbols

Any symbol is used if it appears on right hand side in production rule and generates terminal strings. no such derivation exist. then it is supposed to be an useless symbol.

Ex:- consider CFG. $G = (V, T, P, S)$ where $V = \{S, A, B\}$, $T = \{0, 1\}$,

$$P = \begin{cases} S \rightarrow A11B \mid 11A \\ S \rightarrow 1B \mid 11 \\ A \rightarrow 0 \\ B \rightarrow BB \end{cases}$$

In the above grammer there is one useless symbol i.e B. so Remove the production that involves no-terminal B. After removing the useless symbol 'B'. The grammer will be

$S \rightarrow 11A \mid 11$

$A \rightarrow 0$

Ex:-2 find CFG with no useless symbols Equivalent to G given below.

$S \rightarrow AB \mid CA$

$A \rightarrow a$

$B \rightarrow BC \mid AB$

$C \rightarrow aB \mid b$

In the above grammer there is only one useless symbol 'B'. After removing 'B' the grammer will be.

$S \rightarrow CA$

$A \rightarrow a$

$C \rightarrow b$

(ii) $S \rightarrow aA/bB$

$A \rightarrow aA/a$

$B \rightarrow bB$

$D \rightarrow ab/\epsilon a$

$\epsilon \rightarrow aC/d$

where 's' is start symbol.

In the above grammer there are four useless symbol 'B, C, D, E'. After removing B, C, D, E the grammer will be:

$S \rightarrow aA$

$A \rightarrow aA/a$

(iii) $S \rightarrow aA/bB/cC/a$

$A \rightarrow aB$

$B \rightarrow a/Aa$

$C \rightarrow CCD$

$D \rightarrow ddd$.

In the above grammer there are two useless symbols 'C & D'. After removing 'C & D' the grammer will be.

$S \rightarrow aA/bB/a$

$A \rightarrow aB$

$B \rightarrow a/Aa$

(iv) $S \rightarrow aS/A/C$

$A \rightarrow a$

$B \rightarrow aa$

$C \rightarrow aCb$

In the above grammer there are two useless symbols 'B & C'. After removing B & C then the grammer will be

$S \rightarrow aS/A$

$A \rightarrow a$

(v) $S \rightarrow AB$

$A \rightarrow aA/a$

$B \rightarrow bB/b$

$C \rightarrow aA/a$

$D \rightarrow bB/d$

In the above grammer there are two useless symbols 'C & D'. After removing C & D then the grammer will be

$S \rightarrow AB$

$A \rightarrow aA/a$

$B \rightarrow bB/b$

Elimination of ϵ-productions :-

production rule which involves ϵ such production is called as ϵ-production.

Ex:-1. consider the following grammer. Eliminate ϵ-production.

$S \rightarrow asa/bSb/\epsilon$

After eliminating ϵ-production the grammer will be: $S \rightarrow asa/bSb/aa/bb$

2. Eliminate ϵ-production from the following grammer $S \rightarrow xyx$

$X \rightarrow 0x/\epsilon$

$Y \rightarrow 1y/\epsilon$

After eliminating ϵ-production the grammer will be: $S \rightarrow xyx/xy/yx/y/x$

$X \rightarrow 0x/0$

$Y \rightarrow 1y/1$

3. Eliminate ϵ-production from the following grammer.

$A \to 0B1 \mid 1B1$

$B \to 0B \mid 1B \mid \epsilon$

After eliminating ϵ-production the grammer will be $A \to 0B1 \mid 1B1 \mid 01 \mid 11$

$B \to 0B \mid 1B \mid 0 \mid 1$

4. Eliminate ϵ-production from the following grammer.

$S \to a \mid Ab \mid aBa$

$A \to b \mid \epsilon$

$B \to b \mid A$

After eliminating ϵ-production the grammer will be: $S \to a \mid Ab \mid aBa \mid b \mid aa$

$A \to b$

$B \to b$

5. Eliminate ϵ-production from the following grammer.

$S \to 0S1 \mid 0B1$

$S \to 0A1$

$B \to \epsilon$

$A \to 01$

After eliminating ϵ-production the grammer will be

$S \to 0S1 \mid 0A1 \mid 01$

$A \to 01$

Removing unit productions :-

A production in which one non-terminal gives another non-terminal such production is called unit production.

Ex:- $X \to Y$, $Y \to Z$ etc

1. Remove unit production from the following grammer.

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

$A \to 0S \mid 00$

$B \to 1 \mid A$

$C \to 01$

In the above grammer there are two unit productions i.e $S \to C$, $B \to A$.
So replace 'i' with resultant grammer.

$S \to 0A \mid 1B \mid 01$
$A \to 0S \mid 00$
$B \to 1 \mid 0S \mid 00$.

2. Eliminate unit production from the following grammer.

$S \to AB$
$A \to a$
$B \to C \mid b$
$C \to D$
$D \to E \mid bc$
$E \to d \mid Ab$

In the above grammer there are three unit productions are there. they are

$C \to D$
$B \to C$
$E \to A$

After removing these three the contex will be

$S \to AB$
$A \to a$
$B \to d \mid Ab \mid bc \mid b$
$C \to d \mid Ab \mid bc$.

4. Optimize the following Grammer.
(i.e elimination of ϵ-production, unit production an useless symbols)

Ex:- $S \to A \mid 0C1$
$A \to B \mid 01 \mid 10$
$C \to E \mid CD$

there are two useless symbols i.e B, D then ϵ-production will be

$S \to 01 \mid 10$

2. $S \to AB$
$A \to a \mid E$
$A \to c \mid b$
$B \to c \mid b$
$C \to D$
$D \to E \mid bc$
$E \to d \mid Ab$

Normal forms :-

Grammer can be simplified by reducing useless symbols, ε-production and unit-production there is also need to have grammer in specific form. If any grammer represented in specific form then we need to normalize such grammer. there should be fixed no. of terminals and non-terminals. They are two important Normal form. 1. chomsky's Normal form (CNF)

2. Greibach Normal form (GNF)

1. chomsky's Normal form :-

The grammer is said to be chomsky's normal form, if all the production rules are in the form non-terminal to non terminal.

i-e non-terminal ⟶ non-terminal . non terminal

non-terminal ⟶ terminal.

Ex: consider the following grammer convert to CNF

S ⟶ aSa

S ⟶ bSb

S ⟶ a

S ⟶ b

$S \to ASA$

$S \to BSB$

$S \to a$

$b \to b$

$P \to AS$

$Q \to BS$

$\Rightarrow$

$S \to PA$

$S \to QA$

$S \to a$

$B \to b$

$S \to b$

$A \to a$

$P \to AS$

$Q \to BS$

$\therefore A \to a$

$B \to b$

2. convert the following CFG into CNF

$S \to ABA$

$A \to aA \mid \epsilon$

$B \to bB \mid \epsilon$

first eliminate ϵ-production, unit-production & useless symbols then.

$S \to ABA \mid BA \mid AA \mid AB \mid B$

$A \to aA \mid a$

$B \to bB \mid b$

$\Rightarrow$

$S \to AB\overset{X}{A} \mid BA \mid AA \mid AB \mid b\overset{X}{B} \mid b$

$A \to a\overset{X}{A} \mid a$

$B \to b\overset{X}{B} \mid b$

let $D \to BA$ then $S \to AD \mid AA \mid BA \mid AB \mid EB \mid b$.

$\epsilon \to b$

$F \to a$

$A \to FA \mid a$

$B \to \epsilon B \mid b$

$\epsilon \to b$

$F \to a$

3. convert the following grammer into CNF.

$S \to aB \mid bA$

$A \to a \mid aS \mid bAA$

$B \to b \mid aS \mid aBB$

There is no ϵ-production, unit production & useless symbols then

$S \to aB \mid bA$

$A \to a \mid aS \mid bAA$

$B \to b \mid aS \mid aBB$

$S \to DB \mid \epsilon A$

let $D \to a$

$\epsilon \to b$

$A \to a \mid DS \mid \epsilon AA$

$B \to b \mid DS \mid DBB$.

let $AA \to F$

$BB \to G$

$S \to DB \mid \epsilon A$

$A \to a \mid DS \mid \epsilon F$

$B \to b \mid DS \mid DG$.

∴ final CNF is

$S \to DB \mid \epsilon A$

$A \to a \mid DS \mid \epsilon F$

$B \to b \mid DS \mid DG$

$D \to a$

$\epsilon \to b$

$AA \to AA$

$GBB \to BB$

4. convert the following grammer into CNF

$S \to A \mid 0C1$

$A \to B \mid 01 \mid 10$

$C \to \epsilon \mid CD$

first eliminate ϵ-production, unit production & useless symbol then.

$S \to A \mid 0C1 \mid 01$

$A \to 01 \mid 10$

$\Rightarrow$

$S \to 01 \mid 10 \mid 01$

$\Downarrow$

$S \to 01 \mid 10$

$B \& D$ then

$A \to 01 \mid 10$

$C \to \epsilon$

let $0 \Rightarrow A$

$1 \to B$

$S \to AB \mid BA$

∴ final CNF : $\boxed{S \to AB \mid BA}$

5. Convert CFG to CNF

$S \to CA$

$A \to a$

$C \to aB \mid b$

first eliminate ϵ-production, unit-production & useless symbols.

there is only one useless symbol i.e B then $S \to CA$, $A \to a$, $C \to b$

There is no ϵ-production & unit-production.

∴ final CNF is $S \to CA$, $A \to a$, $C \to b$

2. Grebach Normal form (GNF)

A grammer is said to be Grebach Normal form if all production rules are in the form of non-terminal

non-terminal → terminal. Any no. of non-terminals.

Ex:- $A \to a$, $A \to aBA$, $A \to ABla \times$ Not GNF.

convert the following grammer into GNF.

$S \to CA$

$A \to a$

$C \to aB \mid b$

There is no ϵ-production, unit-production & useless symbol.

there is one useless symbol i.e B then $S \to CA$, $A \to a$, $C \to b$

∴ GNF is $S \to bA$, $A \to a$

Convert the following grammer to GNF.

$S \to ABA$

$A \to aA \mid \epsilon$

$B \to bB \mid \epsilon$.

First eliminate ϵ-production.

$S \to ABA \mid BA \mid AB \mid B \mid AA$.

$A \to aA \mid a$

$B \to bB \mid b$

There is no useless symbols.

Eliminate unit productions then $S \to ABA \mid BA \mid AB \mid AA \mid bB \mid b$

$A \to aA \mid a$

$B \to bB \mid b$.

Ans

$S \to aABA \mid aBA \mid bBA \mid bBA \mid aAB \mid aAB \mid aAA \mid aA \mid bbB \mid bb \mid b$.

$A \to aA \mid a$

$B \to bB \mid b$

Convert the following gramer to GNF

$S \to aB \mid bA$

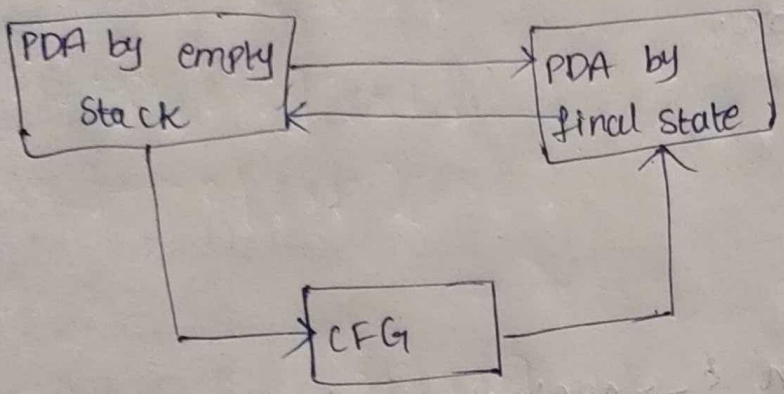
$A \to a \mid aS \mid bAA$

$B \to b \mid aS \mid aBB$.

There is no ϵ-production,

Equivalence of PDA and CFG :-

following diagram represents the Equivalence of PDA and CFG.



Convert CFG to PDA :-

1. Convert the following CFG to PDA.

$S \rightarrow 0S1 / A$

$A \rightarrow 1A0 / S / \epsilon$

Sol :- first write non-terminals $= \{S, A\}$ & terminals $= \{0, 1, \epsilon\}$

$(q, \epsilon, S) = (q, 0S1), (q, A)$

$(q, \epsilon, A) = (q, 1A0), (q, S), (q, \epsilon)$

2. convert the following CFG to PDA.

$S \rightarrow 0AA$

$A \rightarrow 1A0 / S / \epsilon$

first write non-terminals $= \{S, A\}$ & terminals $= \{0, 1, \epsilon\}$

$(q, \epsilon, S) = (q, 0AA)$

$(q, \epsilon, A) = (q, 1A0), (q, S), (q, \epsilon)$

$\delta(q, 0, 0) = (q, \epsilon)$

$\delta(q, 1, 1) = (q, \epsilon)$

Convert CFG to PDA.

$E \rightarrow I \mid \epsilon$

$E \rightarrow E+E \mid E*E \mid (E)$

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

non-terminals $= \{E, I\}$ & terminals $= \{\epsilon, a, b, 0, 1\}, +, *, (,)\}$

$\delta(q, \epsilon, E) = (q, I), (q, E), (q, E+E), (q, E*E), (q, (E))$

$\delta(q, \epsilon, I) = (q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)$

Conversion of PDA to CFG

Convert the PDA $P = (\{q, p\}, \{0, 1\}, \{z_0, x\}, q, \delta, z_0, \{p\})$ to CFG

(Q, Σ, Γ, start, stack, final)

where δ is given by: 1. $\delta(q, 0, z_0) = (q, xz_0)$

2. $\delta(q, 1, x) = (q, x)$

3. $\delta(q, 0, x) = (q, xx)$

4. $\delta(q, \epsilon, x) = (p, \epsilon)$

5. $\delta(p, \epsilon, x) = (p, \epsilon)$

6. $\delta(p, 1, x) = (p, xx)$

7. $\delta(p, 1, z_0) = (p, \epsilon)$

$S \rightarrow qz_0q \mid qz_0p$

Formula 1:

$[qz_0q] \rightarrow 0.[qxq][qz_0q]$

$[qz_0q] \rightarrow 0.[qxp][pz_0q]$

$[qz_0p] \rightarrow 0.[qxq][qz_0p]$

$[qz_0p] \rightarrow 0.[qxp][pz_0p]$

from 3 :-

$[q \times q] \Rightarrow 0 \cdot [q \times q][q \times q]$

$[q \times z_0] \rightarrow 0 \cdot [z \times q][P \times q]$

$[q \times P] \Rightarrow 0 \cdot [q \times q][q \times P]$

$[q \times P] \rightarrow 0 \cdot [q \times P][P \times P]$

formula-6 :-

$[P \times P] \Rightarrow 1 \cdot [P \times I][P \times P]$

$[A \times P] \rightarrow 1[P \times 1][q \times P]$

$[P \times q] \rightarrow 1 \cdot [P \times P][P \times q]$

$[P \times q] \rightarrow 1 \cdot [P \times q][q \times q]$

from rule number 2 :-

$[q \times q] \rightarrow 1 \cdot [q \times q]$

$[q \times P] \rightarrow 1 \cdot [q \times P]$

Rule number 4 :-

$[q \times P] \rightarrow \epsilon$

Rule No :- 4 :-

$[P z_0 P] \rightarrow 1$

Rule No :- 5 :-

$[P \times P] \rightarrow \epsilon$

convert PDA to CFG where $P=(\{P,q\},\{0,1\},\{X,Z_0\},\delta,q,Z_0,\{P\})$

where δ is 1. $\delta(q,1,Z_0)=(q,XZ_0)$

2. $\delta(q,0,X)=(P,X)$

3. $\delta(q,1,X)=(q,XX)$

4. $\delta(q,\epsilon,X)=(q,\epsilon)$

5. $\delta(P,1,X)=(P,\epsilon)$

6. $\delta(P,0,Z_0)=(q,Z_0)$

$S \rightarrow [qZ_0q][qZ_0P]$

let A B

from rule 1:-

$[qZ_0q] \rightarrow 1.[qXq][qZ_0q]$ (A → C A, C → X, A → A)

$[qZ_0q] \rightarrow 1.[qXP][PZ_0q]$ (A → D E)

$[qZ_0P] \rightarrow 1.[qXq][qZ_0P]$ (B → C B)

$[qZ_0P] \rightarrow 1.[qXP][PZ_0P]$ (B → D F)

$S \rightarrow AB$

$A \rightarrow 1.CA \mid 1.DE.$

$B \rightarrow 1.CB \mid 1.DF.$

from rule 2:-

$[qXP] \rightarrow 0.[PXP]$ (D → 0 H)

$[qXP] \rightarrow 0.[PXq]$ (D → 0 G)

$D \rightarrow 0.H$

$C \rightarrow 0.F$

from rule -4:-

$[qXq] \rightarrow \epsilon.$

from rule -5:-

$[PXP] \rightarrow \epsilon.$

pumping lemma for CFL:-

Pumping lemma for CFL is used to prove that the language is not a context free language.

Theorem:- consider 'L' is a CFL on any string 'w' w

we can break the string 'w' into 5 parts such that (i) $|vx| \geq 1$ (ii) $|vxy| \leq n$ for all $k \geq 0$ the string $uv^k wx^k y$ belongs to 'L'

Ex:- Prove the following language is not Reg context lang

$L = \{ a^n b^n c^n / n \geq 1 \}$

Let us consider give L is a CFL. Take consider of $L \geq n$ whe n is no. of strains split the string 'w' into 5 parts i.e uv^k such that (i) $|vx| \neq \epsilon$ (ii) $|uwy| \geq n$ (iii) for $k \geq 0$ the strin $uv^k wx^k y$ belongs to 'L'. consider the string $S = a^{n-1} a^1 b^{n-1}$

$w = a^{n-1} a b^{n-1} b c^n$

$= a^{n+1} b^{n+1} c^n$

$= a^{n+1} b^{n+1} c^n$

① Prove the language is not context free language.

$L = \{ a^m b^n c^m a^n / m, n \geq 1 \}$.

Closure properties of CFL :-

The following are various procos of CFL.

1. union of two text language are context free language.
2. concatination of two languages are in CFL.
3. complement of language is not in CFL
4. Intersection of two language is not is the grammer.
5. Homorghism of CFL is LNG.

Turing Machine.

Construct a Turing machine to accept $L=\{a^n b^n c^n \mid \text{where } n \geq 1\}$

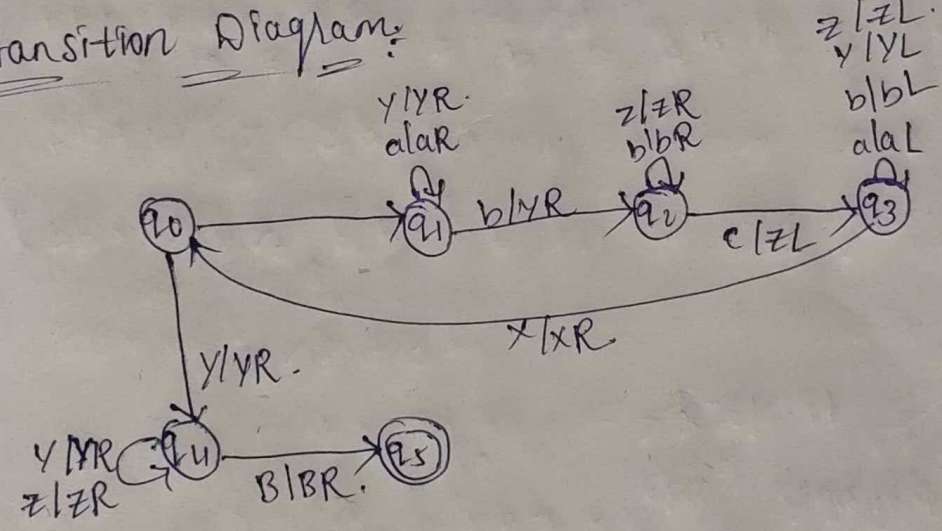
$L=\{a^n b^n c^n \mid \text{where } n \geq 1\}$.

		X		X	Y	Y	Y	Z	Z	Z		
B	a	a	a	b	b	b	c	c	c	B	B	
	↑											

state	a	b	c	x	y	z	B
q_0	(q_1, X, R)				(q_4, Y, R)		
q_1	(q_1, a, R)	(q_2, Y, R)			(q_2, Y, R)		
q_2		(q_2, b, R)	(q_3, Z, L)			(q_2, Z, R)	
q_3	(q_3, a, L)	(q_3, b, L)		(q_0, X, R)	(q_3, X, L)	(q_3, Z, L)	
q_4					(q_4, Y, R)	(q_4, Z, R)	(q_5, B, R)
q_5 ✓							

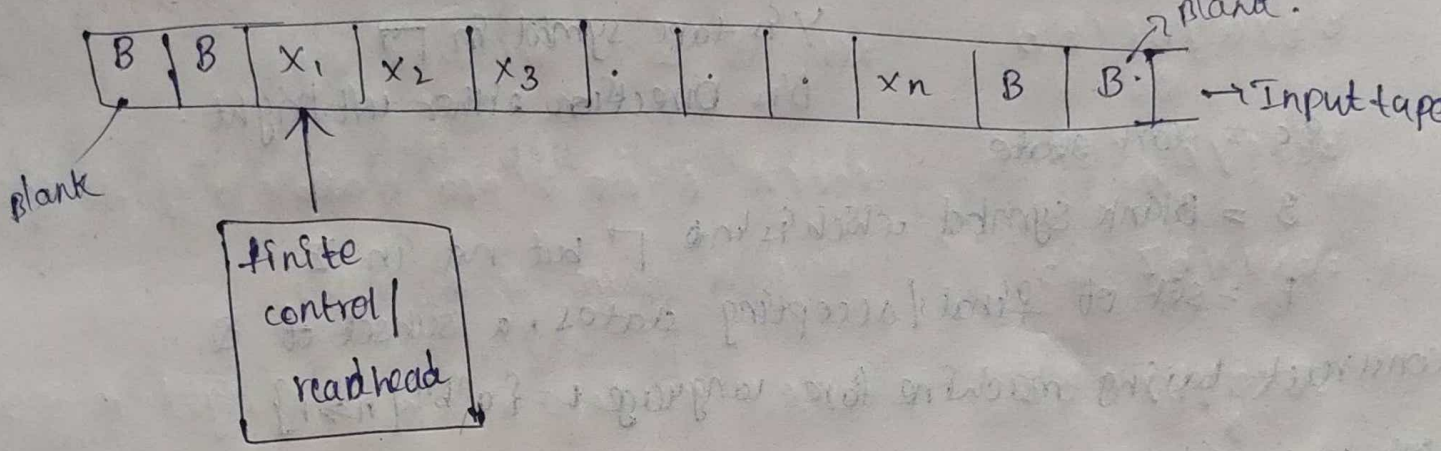
Final state

Transition Diagram:



Turing Machine

Turing machine is mathematical model which consists of an infinite length tape divided into cells on which input is given. Each cell can hold any one of finite number of cells. following diagram represents the model of Turing machine.



Input tape :- which is used to place given string. initially it is filled with special symbol called blank.

Finite control / read head :- Take head or finite control is at left most cell that force the input in one move the turing machine will

1. change the state
2. write a tape symbol - In cell scan.
3. move the tape head left or right.

Blank :- Blank is an tape symbol but not an input symbol. Initially input tape is filled with special symbol called blank.

formal Notation for turing Machine :-

formal notation for turing machine is similar to finite automata or PDA.

→ 7 tuple turing machine is described as $M=(Q, \Sigma, \Gamma, \delta, q_0, B$ whose components have following names

Q = finite

Σ = finite set of input symbols

Γ = complete set of tape symbols (i.e $\Sigma \in \Gamma$)

δ = transition function in which $\delta(q,x)$ is defined as a triple.

i.e (p, y, D) where p is next state in Q

y is tape symbol in Γ

D is Direction either left/right.

q_0 = start state

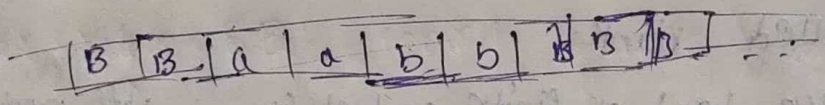
B = Blank symbol which is in Γ but not in Σ

F = set of final/accepting states, a subset of Q.

construct turing machine fore language $L=\{a^n b^n / n \geq 1\}$.

let us consider

$n=2$



let us consider.

1) B a a B B - compare and cal (x) and move right

2) B X a b B B - keep it same and move right.

3) B X a Y b B - convert 'b' into 'Y' & move left.

4) B X a Y b B - keep it same & move left in search of X.

5) B X a Y b B - keep it same, move R.

BXaYbB — convert 'a' into 'x' and move Right.

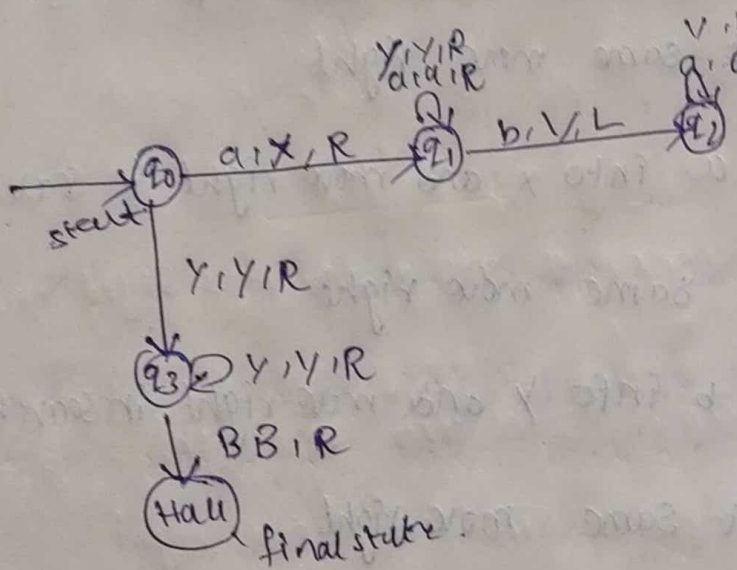
BXXYbB — keep it same, move right, search of 'b'

BXXYbB — convert into Y and move L in search of 'x'

BXXYYB — keep it same and move L.

BXXYYB — move Right.

BXXYYB — move right till B and stop.



state	a	b	X	Y	B
q_0	(q_1, X, R)	$\emptyset$	$\emptyset$	(q_3, Y, R)	-
q_1	(q_1, a, R)	(q_2, Y, L)	-	(q_1, Y, L)	-
q_2					
q_3					
halt					

- Design turing machine for $L=\{a^n b^n c^n / n \geq 1\}$

let Blank be 'B'

let n=2

B B a a b b c c B B B.

convert a into x and move Right in search of b.

B B x a b b c c B B Keep it same move right.

B B x a b b c c B B convert b into y and move right in search of

B B x a y b c c B B Keep it same and move right.

B B x a y b c c B B convert c into z and move left in search of

B B x a y b z c B B Keep it same move right.

B B x a y b z c B B convert a into x and move right in search of

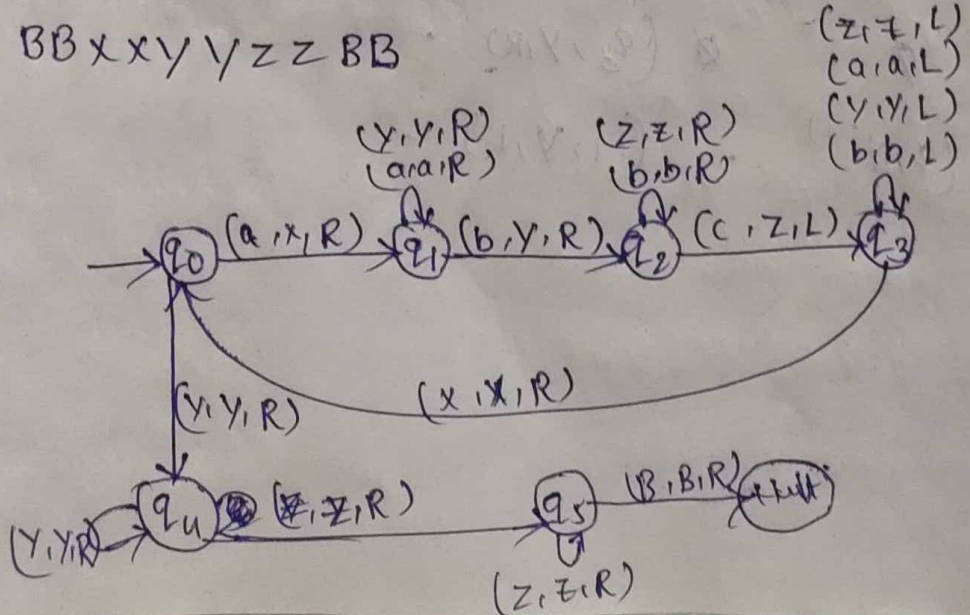
B B x x y b z c B B Keep it same move right.

B B x x y b z c B B convert b into y and move right in search of

B B x x y y z c B B Keep it same move right.

B B x x y y z c B B convert c into z and move till B and sto

B B x x y y z z B B



construct turing machine for the language $L=\{a^n b^{2n} / n \geq 1\}$

let blank be 'B'

let us consider n = '2'

BBaabbbbBB — convert a into x and move to Right in search of 'b'

BBXabbbbBB — can keep it same and move right.

BBXabbbBB — convert b into Y and move Left in search of 'X'

BBXaYbbbBB — keep it same and move right.

BBXaYbbbBB — convert a into X and move right in search of 'Y'

BBXXYbbBB — keep it same and move right.

BBXXYbbBB — convert b into Y and move left in search of 'X'

BBXXYYbBB — keep it same and move right.

BBXXYYbBB — keep it same and move right.

BBXXYYbBB — " " " "

BBXXYYbBB — " " " "

BBXXYYbBB — convert b into Y and move left in search of 'X'

BBXXYYbBB — keep it same and move right.

BBXXYYYbBB — " " "

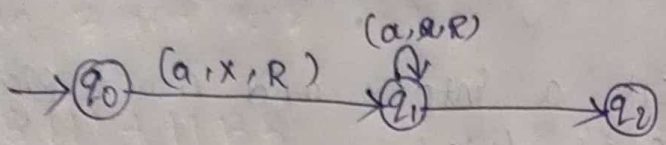
BBXXYYYbBB — " " "

BBXXYYYbBB — " " "

BBXXYYYbBB — convert b into Y and move until to get blank.

BBXXYYYYBB

BBXXYYYYBB move till B and stop.



Construct Turing machine for $L=\{a^i b^j c^i d^j / i,j \geq 1\}$

let blank be 'B'

let $i=2$, $j=3$ then

BB a a bbb C C ddd BB convert a into x and move right in search of 'c'

BBXa bbbCCdddBB keep it same and move right.

BBXabbbCCdddBB keep it same and move right.

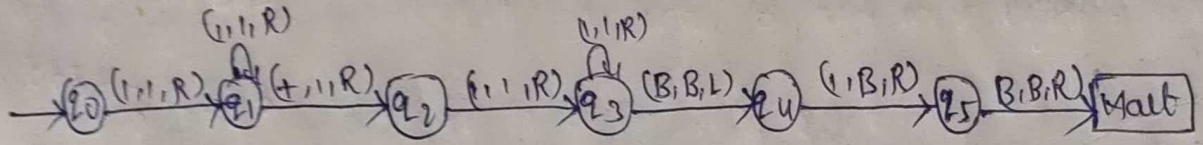
BBXabbbCCddd BB " " " "

BBXabbbCCddd BB " " " "

Design turing machine for adding of two numbers where m, n are unary numbers.

let m=3 and n=2 then

BB111+11BB	keep it same and move right.
BB111+11BB	" " "
BB111+11BB	" " "
BB111+11BB	convert + into 1 and move right.
BB1111 11BB	keep it same and move right.
BB1111 1BB	convert 1 into B and move right.
BB11111BBB	move till B and stop.



Design turing machine for subtracting of two numbers where m, n.

let m=3 and n=5

B111 −11111B (→blank)	convert '1' into blank 'B' and move right in path of 'B'
BB11 −11111B	Keep it same move right.
BB1 1−11111B	Keep it same, move right.
BB11−11111B	" " " "
BB11−11111B	" " " "

BBII - IIIIIB Keep it same move right.

BBII - IIIIIB " " "

BBII - IIIIIB " " "

BBII - IIIIIB " " "

BBII - IIIIIB Keep it same and move left.

BBII - IIIIIB move till blank 'B' and stop

BBII - IIIIIB convert 'I' into blank 'B' and move left.

BBII - IIIIBB Keep it same move left

BBII - IIIIBB " " "

BBII - IIIIBB " " "

BBII - IIIIBB " " "

BBII - IIIIBB " " "

BBII - IIIIBB " " "

BBII - IIIIBB " " "

BBII - IIIIBB " " "

BBII - IIIIBB " " movetright

BBII - IIIIBB " " "

BBII - IIIIBB convert 'I' into 'B' and move right.

BBBI - IIIIBB Keep it same & move right

BBBI - IIIIBB " " "

BBBI - IIIIBB " " "

BBBI - IIIIBB " " "

BBBI - IIIIBB " " "

BBB1 - 1111 BB
↑

BBB1 - 1111 BB
↑

BBB1 - 1111 BB
↑

BBB1 - 1111 BB
↑

BBB1 - 111 1 BB
↑

BBB1 - 111 BBB
↑

BBB1 - 111 BBB
↑

BBB1 - 111 BBB
↑

BBB1 - 111 BBB
↑

BBB1 - 111 BBB
↑

BBBB - 111 BBB
↑

BBBB - 111 BBB
↑

BBBB - 111 BBB
↑

BBBB - 111 BBB
↑

BBBB - 11 BBBB
↑

BBBB - 11 BBBB
↑

BBBB - 11 BBBB
↑

BBBB - 11 BBBB
↑

Keep it same move right.

" " "

" " move left.

" " "

convert '1' into 'B' and move left.

keep it same & move left.

" " "

" " "

" " "

" " "

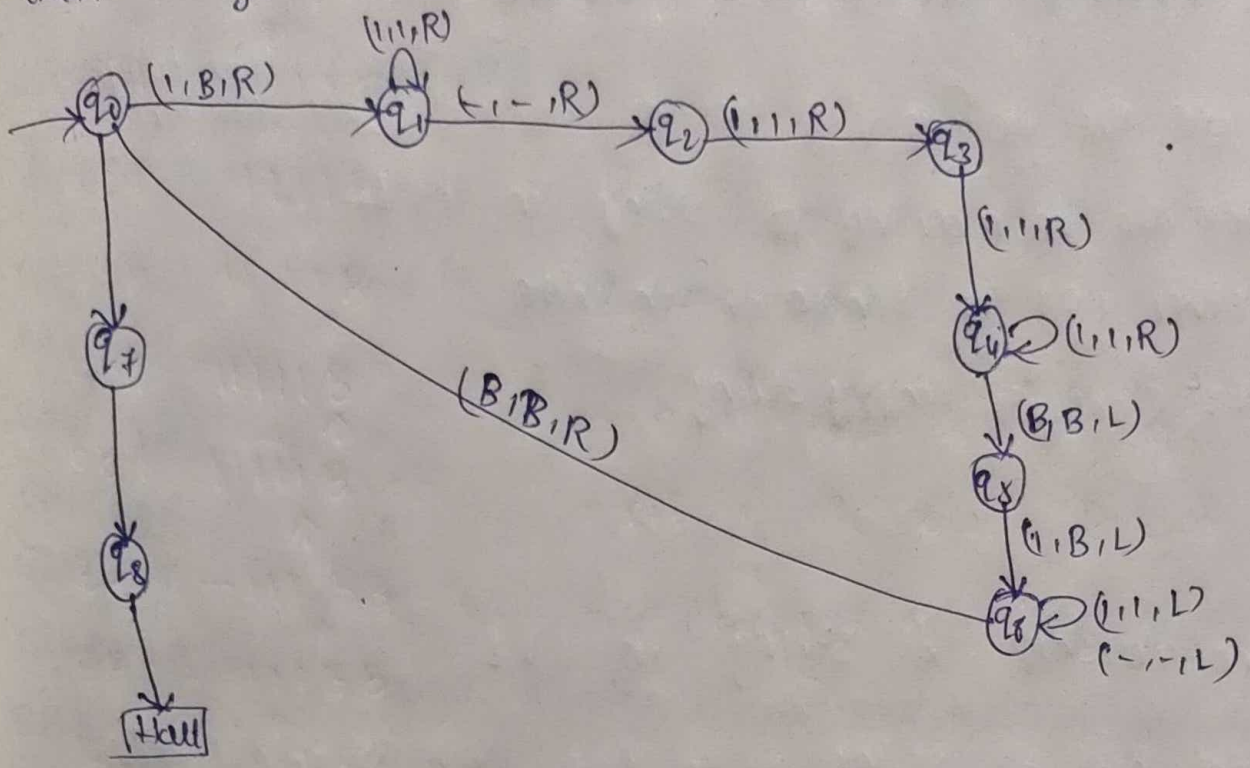
" " & move right.

convert '1' into 'B' and move right

keep it same & move right.

" " "

until ' ' get 1.



m*n

m>n

let m = 4 and n = 3 of (m-n)

B 1111 → 111B convert B '1' into Blank 'b' and move 'R'
↑

BB111 — 111B Keep it same and moveright
↑

BB111 — 111B " " " "
↑

BB111 — 111B " " " "
↑

BB111 — 111B " " "
↑

BB111 — 111B " " "
↑

BB111 — 111B " " "
↑

BB111 — 111B " " "
↑

BB111 — 111B
↑

Design turing machine for $L = \{wcw^R \mid w \in (a+b)^*\}$

Let string be BabCbaB

BabCbaB	convert 'a' into 'B' and move right in path of Blank.
BBbCbaB	Keep it same & move Right.
BBbCbaB	" " "
BBbCbaB	" " "
BBbCbaB	" " "
BBbCbaB	" " & move left in path of 'a'
BBbCbaB	convert 'a' into B and move left in path of 'B'
BBbCbBB	Keep it same & move left.
BBbCbBB	" "
BBbCbBB	" "
BBbCbBB	Keep it same & move right.
BBbCbBB	convert 'b' into 'B' and move right in path of 'B'
BBBCbBB	Keep it same and move right.
BBBCbBB	Keep it same & move right.
BBBCbBB	Keep it same & move left in path of 'b'
BBBCbBB	convert 'b' into 'B' & move lefth in path of 'B'
BBBCBBB	Keep it same & move left.
BBBCBBB	Keep it same & move right to get C and stop Procss.

Decision properties of context Free Language Grammer:-

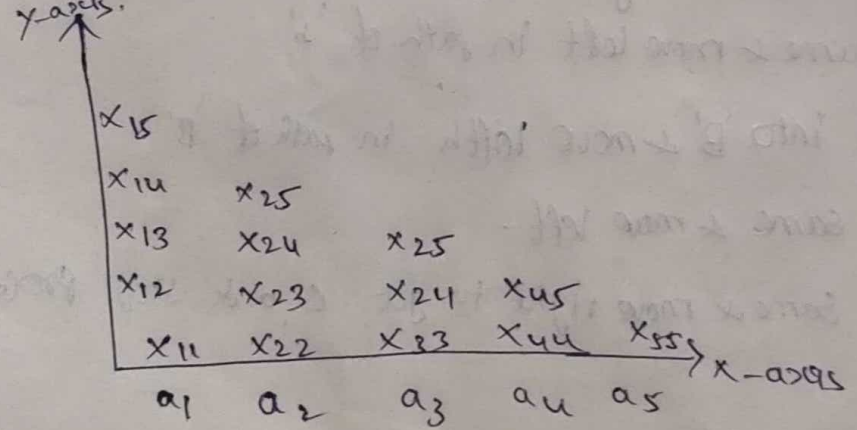
The following are the various properties of context free Language Grammer is

1. To check whether the membership of string in CFL
2. check whether CFL is finite or infinite
3. Is the CFL Generate empty string.

1. To check membership of string in CFL

To decide membership of string w in CFL. there is an efficient technique based on the idea of dynamic programming. It may also Known as Cocke-Younger-Kasami (CYK) algorithm in order of n^3, this algorithm construct a triangular table In which horizonal access corresponds to position of string

Let $w = a_1 a_2 a_3 a_4 a_5$



To fill the table move row by row upwards. Table entry is as follows: $x_{ij} = (x_{ii} x_{i+1j})(x_{ii+1}, x_{i+2j})(x_{ii+2}, x_{i+3j})$. ----

(x_{ij-1}, x_{jj})

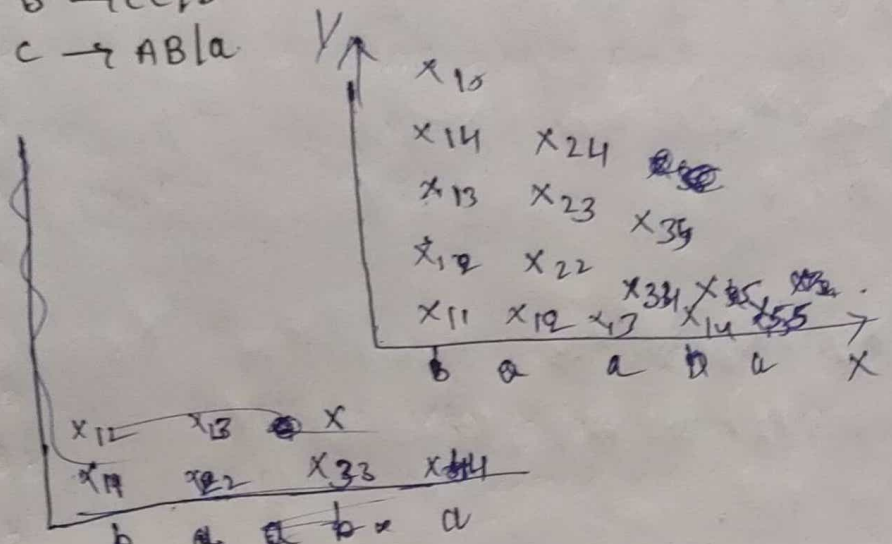
check whether the string 'baaba' is in CFL are not using CYK.

$S \to AB/BC$

$A \to BA/a$

$B \to CC/b$

$C \to AB/a$



$x_{11} = \{B\}$ $x_{12} = \{A, C\}$ $x_{13} = \{A, C\}$ $x_{14} = \{B\}$ $x_{15} = \{A, C\}$

$x_{ij} = (x_{ij}, x_{i+1,j})(x_{i,i+1}, x_{i+2,j})(x_{i,i+2}, x_{i+3,j})$ ---- (x_{ii}, x_{jj})

$x_{12} = (x_{11}, x_{22}) = \{\{B\}, \{A, C\}\} = \{(BA)(BC)\} = \{S, A\}$

$x_{23} = (x_{22}, x_{33}) = \{(A, C), (A, C)\} = \{AA, AC, CA, CC\} = \{B\}$ (CC → D)

$x_{34} = (x_{33}, x_{44}) = (AB, CB) = \{S, C\}$

$x_{45} = (x_{44}, x_{55}) = (BA, BC) = \{S, A\}$.

$x_{13} = (x_{11}, x_{23})(x_{12}, x_{33})$

$= (B, B)(\{S, A\}, \{A, C\}) = (BB) \cup (SA, SC, AA, AC)$

$= (BB, SA, SC, AA, AC) = \emptyset$

$x_{24} = (x_{22}, x_{34})(x_{23}, x_{44})(x_{13}, x_{45})$.

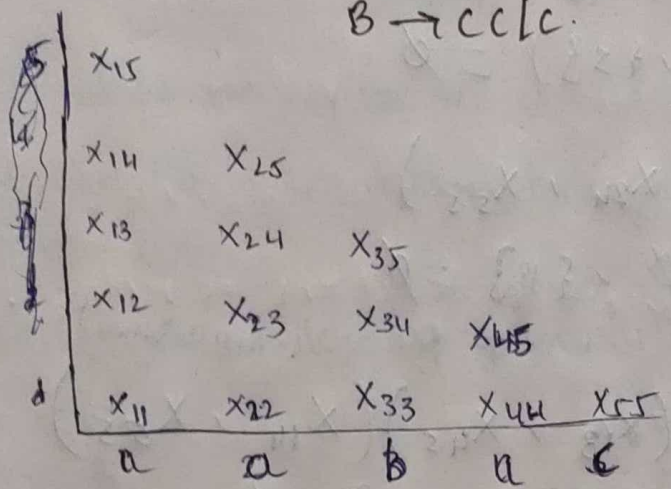
$= (B, \{S, C\})($

check whether the following string "aabac" is in CFL are not using CYK

$S \rightarrow AB/a$

$A \rightarrow BC/b$

$B \rightarrow CC/c$



step-1:-

$X_{11} = \{S\}$ $X_{22} = \{S\}$ $X_{33} = \{A\}$ $X_{44} = \{S\}$ $X_{55} = \{B\}$

step-2:-

$X_{ij} = (X_{ii}, X_{(i+1)j})(X_{i(i+1)}, X_{(i+2)j})(X_{i(i+2)}, X_{(i+3)j}) \dots (X_{i(j-1)}, X_{jj})$

$X_{12} = (X_{11}, X_{22}) = (S, S) = \emptyset$

$X_{23} = (X_{22}, X_{33}) = (S, A) = (SA) = \emptyset$

$X_{34} = (X_{33}, X_{44}) = (\{A\}, \{S\}) = (A, S) = \emptyset$

$X_{45} = (X_{44}, X_{55}) = (\{S\}, \{B\}) = (SB) = \emptyset$

step-3:-

$X_{13} = (X_{11}, X_{23}) \cup (X_{12}, X_{33})$

$= (\{S\}, \{\emptyset\}) \cup (\emptyset, \{A\}) = \emptyset$

$X_{24} = (X_{22}, X_{34}) \cup (X_{23}, X_{44}) = (\{S\}, \emptyset) \cup (\emptyset, \{S\}) = \emptyset$

$X_{35} = (X_{33}, X_{45}) \cup (X_{34}, X_{55}) = (\{A\}, \emptyset) \cup (\emptyset, \{B\}) = \emptyset$

Step-4:-

$$X_{14} = (X_{11}, X_{24})(X_{12}, X_{34})(X_{13}, X_{44})$$
$$= (\{S\}, \phi)(\phi, \phi)(\phi, \{S\}) = \phi$$
$$X_{25} = (X_{22}, X_{35})(X_{23}, X_{45})(X_{24}, X_{55})$$
$$= (\{S\}, X\phi)(\phi, \phi)(\phi, \{B\} = \phi$$

Step-5:-

$$X_{15} = (X_{11}, X_{25})(X_{12}, X_{35})(X_{13}, X_{45})(X_{14}, X_{55})$$
$$= (\{S\}, \{\phi\})(\{\phi\}, \{\phi\})(\{\phi\}, \{\phi\})(\{\phi\}, \{B\}) = \phi$$

Test CFL is finite or not

check whether the following Grammer is finite or not.

$S \rightarrow AB|a$
$A \rightarrow BC|b$
$B \rightarrow CC|C$
$C \rightarrow C$

step-1:- check whether the given grammer is remove ϵ productions, unit produc completely reduced or not.

∴ the given grammer is already reduced.

step-2:- draw directed graph from the given production rule.

from the production rule

~~$A \rightarrow BC|b$~~.

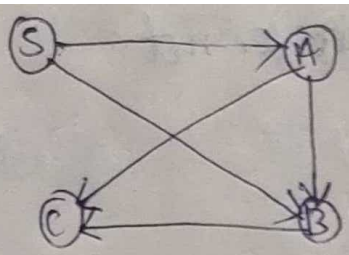
$S \rightarrow AB|a$. we will have $S \rightarrow A$ & $S \rightarrow B$ are edjes directed graph.

from the production rule.

$A \rightarrow BC|b$.

we will have $A \rightarrow B$ & $A \rightarrow C$ are edjes of directed graph.

from the production rule $B \rightarrow CC|C$ we will have $B \rightarrow C$ as the only one edje of directed graph.



Step-3:- Identify the cycles in directed graph.

the above directed graph doesn't have any cycles. so the grammer is said to be finite.

check whether the following grammer is finite or not

$S \rightarrow xs|b$

$x \rightarrow yz$

$y \rightarrow ab$

$z \rightarrow xy$

step-1:- check whether the grammer is completely reduced or not.

the grammer is already reduced.

Step-2:- Draw the directed graph from the given production rule.

from the production rule

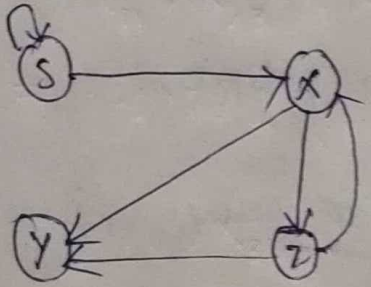
$S \rightarrow xs$ we have $S \rightarrow x$ & $S \rightarrow s$ are edges of directed graph.

from the production rule

$S \rightarrow yz$ we will have $x \rightarrow y$ & $x \rightarrow z$ are edges of directed graph.

from the production rule.

$z \rightarrow xy$ we will have $z \rightarrow x$ & $z \rightarrow y$ are edges of directed graph



Step-3:- Identify the cycles in directed graph. be non finite.

The above directed graph contains two cycles. so the grammer is said to

Check whether the following grammer is finite or not.

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

Step-1 :- check whether the given grammer is reduced or not.

here is no ϵ-productions & unit productions to remove.

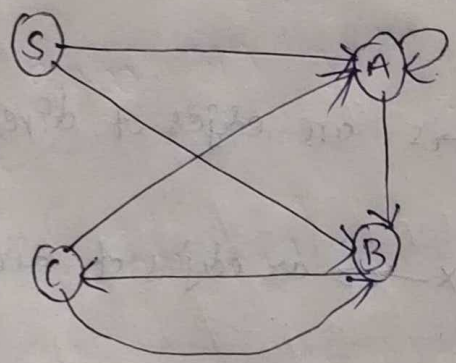
Step-2 :- draw the directed graph.

$S \rightarrow AB$ we have $S \rightarrow A$ & $S \rightarrow B$

$A \rightarrow BA$ we have $A \rightarrow B$ & $A \rightarrow A$

$B \rightarrow CC$ we have $B \rightarrow C$.

$C \rightarrow AB$ we have $C \rightarrow A$ & $C \rightarrow B$.



Step-3 :- the given grammer is infinite because the above graph form cycles.

Finite no. of strings & Infinite no. of strings

Chomsky's Hierarchy:-

It represents class of languages accepted by different machines.

Language class	Language	Machine Grammer	Machine	Example.
Type 3	Regular Language	Regular Grammer	FA	$a^* b^*$
Type 2	context free language	context free grammer	PDA	$a^n b^n$
Type 1	context sensitive language (CSL)	context sensitive grammer	Linear Bounded automata	$a^n b^n c^n$.
Type 0	Recursively executable language.	Recursively executable grammer	TM	$n!$

Undicidability and postcorrespondence problem

→ there are some problems which that are not compatable, such problems are said to be unsduable or undecidable.

→ A language L is subset of Σ^* is recursive if there exist some turing machine 'M'. That satisfies the following two condition:

→ 1. if $w \in L$ then M accepts w and halts

→ 2. if $w \notin L$ then M eventually halts without reaching an accepting state.

→ A language L is subset of Σ^* is recursively enumarable if there exist turing machine 'M'. such that $L = T(M)$

→ A problem with two answers (Yes or No) is decidable if the corresponding language is recursive. In this case the language is called decidable language.

→ A problem or language is undecidable if it is not decided.

post correspondance problem (PCP)

The undecidability of strings is determined/decided with the help of PCP

the PCP contains two list of strings that are of equal length over the input Σ. the two lists are:

$A = w_1, w_2, w_3, \ldots w_n$

$B = x_1, x_2, x_3, \ldots x_n$ then there exist a non emptyset of integers $I_1, I_2, I_3 \ldots I_n$ such that $w_{i_1}, w_{i_2}, w_{i_3} \ldots = x_{i_1}, x_{i_2}, x_{i_3} \ldots$

ex:- check the following strings are decidable or undecidable using PCP

$x = (0, 010.00, 01)$ $Y = (000, 01, 1)$

Given:

$x = (0, 01000, 01)$ $Y = (000, 01, 1)$

Pattern: 2 1 1 3 2 1 1 3

01000 0 0 01 01 000 000 1

the PCP is 2113.

check whether the following strings are decidable or undecidable using group.

$x = (b, bab^3, ba)$ $\quad y = (b^3, ba, a)$

Given

$x = (\overset{1}{b}, \overset{2}{bab^3}, \overset{3}{ba})$ $\quad y = (\overset{1}{b^3}, \overset{2}{ba}, \overset{3}{a})$

2	1	1	3		2	1	1	3
bab^3	b	b	ba		ba	b^3	b^3	a.

The PCP has the solution is 2113

check whether the following have PCP or not.

$x = (100, 0, 1)$ $\quad y = (1, 100, 00)$

Given

$x = (\overset{1}{100}, \overset{2}{0}, \overset{3}{1})$ $\quad y = (\overset{1}{1}, \overset{2}{100}, \overset{3}{00})$

1	3	1	1	3	2	2		1	3	1	1	3	2	2
100	1	100	100	100	0	0		1	00	1	1	00	100	100.

The PCP has the solution is 1 3 1 1 3 2 2.

check whether the following solution have PCP or not.

$x = (0, 1010, 01, 11)$ $\quad y = (000, 0100, 100, 110)$

Given $x = (\overset{1}{0}, \overset{2}{010}, \overset{3}{01}, \overset{4}{11})$ $\quad y = (\overset{1}{000}, \overset{2}{0100}, \overset{3}{100}, \overset{4}{110})$

The given PCP has no solution because for each x_i in X is smaller than each y_i in Y.

Check the following solution have pcp or not

x_i	y_i
01	01^2
11	10
10	1^2

the given solution has noble

check whether the following solution have PCP or not.

1. M = (abb, aa, aaa) N = (bba, aaa, aa)
2. M = (ab, bab, bbaaa) N = (a, ba, bab)

1. Given M = (a^1bb, aa^2, aaa^3) N = (bba^1, aa^2, aa^3)

pattern: 2 1 3 2 1 3

aa abb aaa aaa bba aa.

The PCP is 213.

2. It doesn't have solution.

3. x = (b^1, a^2, aba^3, bb^4) y = (ba^1, ba^2, ab^3, b^4)

pattern: 2 1 3 3 4 2 1 3 3 4

a b aba aba bb ba ba ba ab ab b.

The PCP is 12133 4

4. A = (100, 0, 1) B = (1, 100, 00)

pattern: 1 3 1 1 3 2 2 1 3 1 1 3 2 2

100 1 100 100 1 0 0 1 00 1 1 00 100 100.

The pattern of PCP is 13113 22

5. $X = (B, A, CA, ABC)$ $Y = (CA, AB, A, C)$

pattern:	2	1	3	2	4
	A	B	CA	A	ABC
	AB	CA	A	AB	C

The PCP is 2 1 3 2 4.

Programming Techniques for Turing Machine

*Programming Techniques that are used to construct an efficient Turing machine that function as a powerful as conventional computer.

→ the different techniques that are used to design TM's:

1. storage in finite control or state.
2. Multitracks or multipuletrac.
3. subroutines
4. checking of symbols.

Design Turing Machine for the language $L = \{wcw / w \in (a|b)^*\}$

Design Turing Machine for the Language $L = \{01^* + 10^*\}$.

Storing in finite loop or state

$01^* + 10^*$

State	B	0	1
q_0, B	$\emptyset$	$([q_1, 0], 0, R)$	$([q_1, 1], 1, R)$
q_1, 0	$([q_2, B], B, R)$	$\emptyset$	$([q_1, 0], 1, R)$
q_1, 1	$([q_2, 0], B, R)$	$([q_1, 1], 0, R)$	$\emptyset$
q_2, B	$([q_2, B], B, R)$	$\emptyset$	$\emptyset$

ATFL

one mark

a) Application of pumping Lemma.

b) what is homomorphism.

c) Advantages of PDA over finite automata.

d) Limitations of PDA

e) what is derivation tree?

f) properties of derivation tree.

g) Define ambiguous grammer with an example.

h) construct CFG for palindromes of odd length.

i) Give an example of PDA.

j) Disadvantages of ambiguous grammer.

k) Define unit production

l) what are useless symbols in a grammer.

m) Give an example for CFG

n) Define CFG.

- Six marks

1. a) write about any four closure properties of regular sets

b) state & prove pumping lemma for regular set. prove the language $L=\{a^m b^n \mid m>n, m,n \geq 1\}$ is not regular.

2. Define PDA. Explain its model with neat diagram.

In what way a PDA can show the acceptance of a string.

3. a) Construct CFG for $L = \{a^m b^n c^p \mid m+n=p \& p>=1\}$

b) Construct left most & right most derivations for the string aaabbabbba of the following grammer

$S \rightarrow aB \mid bA$

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

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

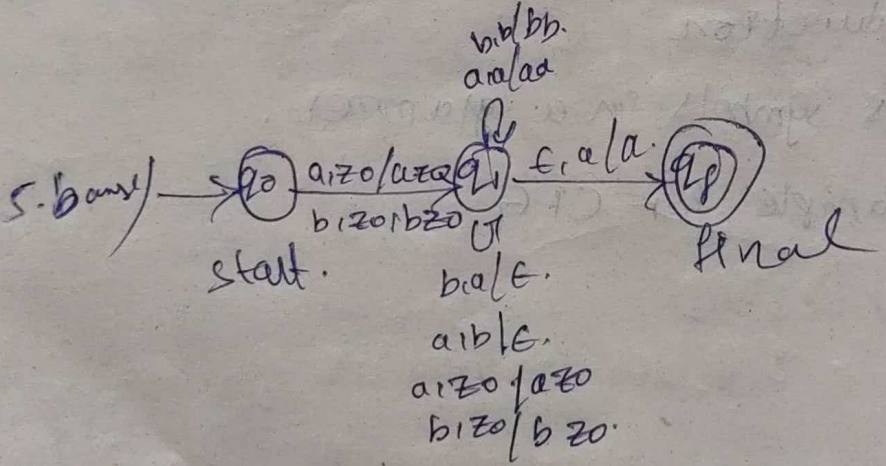
4. Construct PDA for the following

a) $L = \{ww^T \mid w \in (a+b)^*\}$

b) $L = \{w \mid n(a)_w > n(b)_w$ where $w \in (a+b)^*\}$

5. Explain in detail about Normal forms with suitable example.

6. Convert the following PDA into Grammer.



K. Sathrika
IT-A/58

(8) Define DFA
(9) Define NFA.
(10) Define FA.
(7) Applications of FA.
(3) Ardon's theorem.
(4) CFL
(5) Applications of pumping
(6) Define CNF of a CFG.
(7) Define PDA? with example.
(8) formal definition of TM.
(9) Define DCPL and DPDA.
(10) what is a role of checking off symbols in a TM?
(11) what are Recursive & Recursively Enumerable Language
(12) what are UTM or Universal Turing Machines?
(13) what is the relation between $\Sigma^* = \Sigma^+$?
(14) Σ^+ is the subset of Σ^*
(15) what is Regular expression.
(16) write the no. of states in smallest FA which accepts the language $\{x/$ of x is divisible by 3$\}$.
(17) Is $(r^*)^* = r^*$?
(18) what is meant by ambiguous grammar.
(19) Explain the term satisfiability in TM.
(20) How many ways can PDA accepts the string?
(21) why computability functions are needed in the context of TM?
(22) Define pumping lemma.
(23) Define TM.
(24) what is meant by ϵ-closure?

(20) Define a language over a set S.

(21) write the no. of states in a minimum state FA that recognizes the language represented by the regular expression $(0+1)(0+1)\cdots n$ times

(22) Define CFG

(23) IS $(r^*s^*)^* = (r+s)^*$?

(24) Define Derivation tree.

(25) Enumerate the difference b/w PDA & TM.

(26) what is useless symbol.

(27) Let S be the start symbol, A & B are non-terminals & 0 & 1 are the terminals. find the language generated by this grammar.

(28) IS Turing machine recursively enumerable?

(29) How a TM accepts a language.

1. closure properties of R.L
2. Pumping Lemma for R.L
3. construct ϵ-NFA for R.E
4. construct CFG
5. derivation tree/parse tree $\swarrow$ LMD / RMD
6. Ambiguous grammer
7. PDA → Problem or theory
8. Minimization of CFG
9. CFG to CFG