Simplification of CFG:

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

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

if $L(G) = L(G')$

* G & G' are equivalent

① Remove useless symbols.

$$L(G) = \{ w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

↳ symbol does not take part in this ⇒ useless symbol.

Useless Symbol:

(i) Unreachable symbol
(ii) Non-generating

Unreachable symbol → terminal / non-terminal

$$A \Rightarrow \alpha B \beta$$

$$\alpha, \beta \in (V \cup \Sigma)^*$$

$$S \overset{*}{\Rightarrow} w$$

A symbol that is A

(2) Non-generating : A symbol that cannot be generated by using one or more production rules.

$S \to AB$
$A \to a$
$S \to a$.

B ⇒ non-generating
~~A ⇒ unreachable.~~

$S \to asb / \varepsilon$
$A \to aBb / b$
$B \to b$

A, B ⇒ unreachable

- A symbol is called not generating if it can be written as $A \overset{*}{\Rightarrow} w \in \Sigma^*$

Identification of Non-generating Symbols:

Base Case :- $\pi_0 = \{S\}$

$= \{$ all the symbols in RHS of production rules $\}$

~~Unreachable~~:

t cannot be
more production
rule.

ng

achable

ng if it

ods:

in RHS of
}

① Unreachable:

Basic Case:

$\Pi_0 = \{S\}$

$A \in \Pi_0$ & $A \to \alpha \in P$

every symbol B in α
is add to Π_0

$\Pi_1 = \Pi_0 \cup \{B\}$ Unreachable $= V - \Pi_1$

Non-generating:

Basic - Case:

$\Pi_0 = \{A\}$ $A \overset{*}{\Rightarrow} w \in \Sigma^*$

Index:

$B \to \alpha$, every non-terminal in α
is already in Π_0 & B is
not in Π_0.

$\Pi_1 = \Pi_0 \cup \{B\}$

$\Pi' = V - \Pi_1$

$S \to AB$
$A \to a$
$S \to a$

Non-generating

{B}

remove $S \to AB$.

Unreachable:

{A}. (after removal of $S \to AB$)

$S \to a$

Identification of non-generating:

$S \to AB/Bc/CA$

$A \to bBa$

$B \to b/bb$

$C \to CEd$

$E \to cCb$

C, E → non generating.

After simplificatn.

$S \to AB$

$A \to bBa$.

$B \to b/bb$

$S \rightarrow AB|BA$

$A \rightarrow bBb$

$B \rightarrow cd$

$E \rightarrow aCb$

$C \rightarrow ab$

$\pi_0 = \{S\}$

$\pi_0 = \{S, A, B\}$

$\pi_0 \rightarrow$

$S, A, B \Rightarrow$ reachable

$C, E \Rightarrow$ non-reachable.

After simplification:

$S \rightarrow AB|BA$

$A \rightarrow bBb.$

$B \rightarrow cd$

② Removal of ε-productions:

$A \rightarrow \varepsilon$ is know as ε-productions

$L(G') = L(G) | \varepsilon$

$C \to AB$

$C \to A$

$A \to \varepsilon$

$B \to \varepsilon$

$C \xrightarrow{*} \varepsilon$ is called nullable.

Identification of Unreachable:

Q1 'S → nullable AS/AB

$A \to \varepsilon$

$B \to \varepsilon$

D → ~~Dε~~

Q2

$S \to ABC$

$A \to Bc/a$

$B \to bAc/\varepsilon$

$C \to cAB/\varepsilon$

Q1

$A \to \varepsilon$

$B \to \varepsilon$ are removed.

$S \to aS/AB$

$A \to a/\varepsilon$

$B \to b/\varepsilon$

$D \to d$

$S \to aS/A/B$

$A \to a/aB/a$

$B \to b$

$D \to d$

$S \to ABC$

$A \to Bc/$

Q2

$S \to A/B/C/AB/BC/AC$

$A \to B/C/a.$

$B \to bA/bC$

$C \to CA/CB$

③ Removal of Unit Production:

$\boxed{A \to B}$ unit production

$A, B \in V$

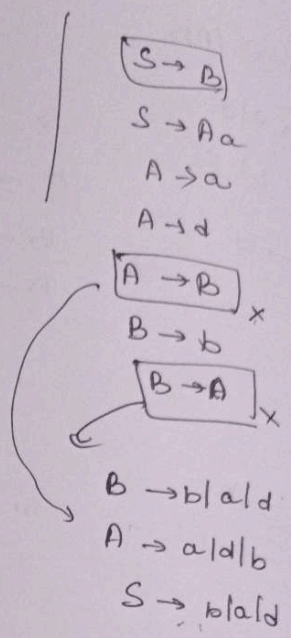
$S \to B$

$B \to a$

$S \to Aa|B$

$A \to a|d|B$

$B \to b|A$



Simplification of CFG:

1. Removal of ε-production
2. Removal of unit production
3. Removal of useless symbol

} safe order

Normal Form:

1. Chomsky Normal Form (CNF):

A grammar is said to be in CNF if

$A \to BC$

$A \to a$

$B, C, A \in V$

$a \in \Sigma$

if a given CFG, & and its corresponding lang $L(G)$, if $\varepsilon \in L(G)$

CFG:
G, $L(G)$
if $\varepsilon \in L(G)$

CNF:
G', $L(G')$

$L(G) = L(G') \cup \varepsilon$

a) a string w of length 'n' can be derived in atmost "$2n-1$" steps.

b) CYK $\Rightarrow O(n^3)$ { algo to check whether the given language is of the given string }.

Ex:

$S \to aSb$

$S \to ab$

$S \to ASB$

$A \to a$

$B \to b$

$S \to ABAB$

$S \to Z_1 B$

$Z_1 \to AS$

Q. $S \to ASB$

$A \to aAS \mid aASA \mid a \mid \varepsilon$

$B \to SbS \mid A \mid bb$

~~$S \to ASB$~~

① if start symbol appears on RHS of any production rule of CFG, then add S_0

$S_0 \to S$

$S \to ASB$

$A \to aASA$

$A \to a$

$\boxed{A \to \varepsilon}$

$B \to SbS$

$\boxed{B \to A}$

$B \to bb$

② Apply safe order of simplificatn steps.

$\boxed{A \to \varepsilon}$

$\boxed{B \to \varepsilon}$

Step 1:

$S_0 \to S$

$S \to AS \mid SB \mid S \mid ASB$

$A \to aAS \mid aSA \mid$ ~~aAA~~ $aS \mid a \mid aASA$

~~$A \to a$~~

$B \to SbS \mid bb \mid A$

Chitti

$S_0 \to ASB \mid SB$

$S \to ASB \mid SB \mid AS$

$A \to aASA \mid a \mid aSA \mid aS$

$B \to SBS \mid bb \mid aASA \mid a \mid aSA \mid aAS \mid aS$

$S_0 \to z_1B \mid SB \mid AS$

$S \to z_1B \mid SB \mid AS$

$A \to z_2A \mid z_3A \mid z_4z_1 \mid z_4S \mid a$

$B \to z_5S \mid z_6z_6 \mid z_2A \mid z_3A \mid z_4z_1 \mid z_1S \mid a$

$z_1 \to AS$

$z_2 \to z_1z_4$

$z_3 \to z_4S$

$z_4 \to a$

$z_5 \to Sz_6$

$z_6 \to b$

CNF: ⇒ use this for parsing purpose

$A \to BC$
$A \to a$

$B, C \in V$

$a \in \Sigma$

① Put $S_0 \to S$, if S appears ~~on~~ RHS of any P rule.

② Apply safe order (ε-prod → Unit-prod → useless)

③ Shortening of RHS

$X \to X_1 X_2 \cdots X_n$, $n \geq 3$

add $Z_{e_1} \to X_1 X_2$

$X \to Z_{e_1} X_3 \cdots X_n$ } & repeat until $n \geq 2$

④ Add non-terminals for terminals.

Ex $A \to aB$. ⇒ $A \to ZB$
$Z \to a$

Greibach

Greibach Normal Form :

$$A \rightarrow aB_1B_2 \cdots B_n, \quad n \geq 1$$
$$A \rightarrow a$$

$|w| = n$

No. of steps to derive a string of length $= n$

① For Every terminal introduce a new non-terminal

Eg:→ $S \rightarrow aSb$
$S \rightarrow ab$

$S \rightarrow ASB$
$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b$

② Introduce ordering among non-terminals

$S \rightarrow ASB$ (A is coming before AB)

$S \leftarrow A_1$
$A \leftarrow A_2$
$B \leftarrow A_3$

$A_1 \rightarrow A_2A_1A_3$
$A_1 \rightarrow A_2A_3$
$A_2 \rightarrow a$
$A_3 \rightarrow b$

③ Use property ① & property ②, and convert the rules in such a way that all rules in you get either GNF or ~~not~~ in GNF the following form.

$$A_i \rightarrow A_j \alpha \quad , \quad j > i$$

④ Prop 1:

G_1: $A \rightarrow \alpha_1 B \alpha_2$

$B \rightarrow B_1 | B_2 -- | B_n$

G_1': $A \rightarrow \alpha_1 B_1 \alpha_2 | \alpha_1 B_2 \alpha_2 | --- | \alpha_1 B_n \alpha_2$

Prop 2:

G_1:

$A \rightarrow A\alpha_1 | A\alpha_2 | --- | A\alpha_n$

$A \rightarrow B_1 | B_2 -- | B_m$ } $m+n$

G_1':

$A \rightarrow B_1 --- | B_m$

$A \rightarrow B_1 Z --- | B_m Z$

$Z \rightarrow \alpha_1 | --- | \alpha_n$

$Z \rightarrow \alpha_1 Z | \alpha_2 Z | --- \alpha_n Z$

(16) Convert Ai rules in GNF.

$A_1 \to A_2A_1A_3$ | $A_1 \to aA_1A_3$

$A_1 \to A_2A_3$ | $A_1 \to aA_3$

(17) Convert

$Z_e \to A_1$ to $Z_e \to aA_1A_3$

~~$S \to SS / AE aSB$~~

Ans

$S \to SS / aSb / ab$

$S \to SS / ASB / AB$
$A \to a$
$B \to b$
} step 1

$S = A_1$
$A = A_2$
$B = A_3$

$S \to$

$A_1 \to A_1A_1$
$A_1 \to A_2A_1A_3$
$A_1 \to A_2A_3$
$A_2 \to a$
$A_3 \to b$
} step 2

Prop. 2:

$A_1 \to A_1A_1$ (A: A_1, α_1)

$A_1 \to A_2A_1A_3 / A_2A_3$ (B: B_1, B)

$\Rightarrow$

$A_1 \to A_2A_1A_3 / A_2A_3$
$A_1 \to A_2A_1A_3Z / A_2A_3Z$
} step 3

$Z \to A_1$
$Z \to A_1Z$

$A_1 \rightarrow aA_1A_3 / aA_3$

$A_1 \rightarrow aA_1A_3Z / aA_3Z$

$Z \rightarrow A_1$

$Z' \rightarrow A_1 Z$

A

$A_1 \rightarrow aA_1A_3 \mid aA_3$

$A_1 \rightarrow aA_1A_3Z \mid aA_3Z$

$Z \rightarrow aA_1A_3 \mid aA_3 \mid aA_1A_3Z \mid aA_3Z$

$Z \rightarrow aA_1A_3Z \mid aA_3Z \mid aA_1A_3ZZ \mid aA_3ZZ$

$A \rightarrow \alpha_1 A \alpha_2$ — simple recursion

$A \rightarrow A\alpha_1$ — left recursion

$A \rightarrow \alpha_1 A_1$ — right recursion

CYK : To test Membership :

↙

Cocke - Younger - Kasami

Given a CFG, G and a string x, is $x \in L(G)$?

↳ Membership test

$L(G) = L(G')$

G' ⇒ CNF / GNF / simplified version of G

Dynamic Pro

↳

x_{ij}

x_{ij} is

i^{th} po

Ex abk

$V_{ij} = \{ A$

set of n

$x_{13} = \{$

=

→ $\alpha_1 A \alpha_2$

simple recursion

→ $A\alpha_1$

left recursion

→ $\alpha_1 A_1$

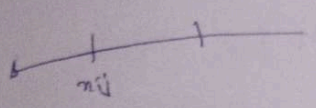
right recursion

$\in L(G)$?

of G

Dynamic Programming :

↳ used for optimisation purpose.



x_{ij} is the part of x which starts from i^{th} position of x & its length is j.

Ex: abba → x.

x_{22}

x_{ij}

$x_{14} = 4 = |x|$

$V_{ij} = \{ A \in V \mid A \overset{*}{\Rightarrow} x_{ij} \}$

set of non-terminals that can drive x_{ij}.

$x_{13} = \{ x_{11}\ x_{27}\ x_{31} \}$

$= \{ x_{12}\ x_{31} \}$

CYK Algorithm:

To solve decision Prob

$V_{ij} = \{A \in V / A \xrightarrow{*} a_{ij}\}$

$x \in L(G)$ iff V_{1n} contains "S"

Important Algo:-

for $i \leftarrow 1$ to n

$V_{ij} = \{A / A \to a$ in P having the symbol $x\}$

for $j \leftarrow 2$ to n

for $i \leftarrow 1$ to $n-j+1$

$V_{ij} \leftarrow \phi$

for $k \leftarrow 1$ to $j-1$

$V_{ij} \leftarrow V_{ij} \cup \{A / A \to BC$, where $B \in V_{ik}$ & $C \in V_{(i+k)(j-k)}\}$

Ex:

G:

$S \to AB/BC$

$A \to BA/a$.

$B \to CC/b$

$C \to AB/a$.

$x = baaba$.

$|x| = 5$

$V = \{S, A, B, C\}$

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

$S = \{S\}$

n = baaba.

Single/length of 1 = 5

length of 2 = 4

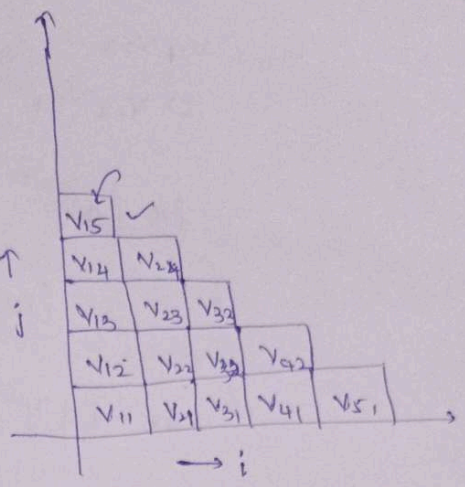
length of 3 = 3

length of 4 = 2

" " 5 = 1

15.

$= \frac{n(n+1)}{2}$



$V_{12} = V_{11} V_{21}$

$V_{11} = \{B\}$ $V_{12} = \{A, S\}$

$V_{21} = \{A, C\}$ $V_{22} = \{B\}$

$V_{31} = \{A, C\}$ $V_{32} = \{S, A\}$

$V_{41} = \{B\}$ $V_{42} = \{A, S\}$

$V_{51} = \{A, C\}$

$V_{12} = V_{11} V_{21}$ $i=1, j=2$ $K=1$

$= \{BA, BC\} = \{A, S\}$

$V_{22} = V_{21} V_{31}$ $i=2$ $j=2$ $k=1$

$= \{A, C\}\{A, C\} = \{AA, AC, CA, CC\} = \{B\}$

$V_{32} = V_{31} V_{41}$

$= \{A, C\}\{B\} = \{AB, CB\} = \{S, C\}$

$V_{42} = V_{41} V_{51}$

$= \{B\}\{A, C\}$

$= \{BA, BC\} = \{A, S\}$

Pushdown Automata: (PDA)

DFA/NFA :⟶ regular language.

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

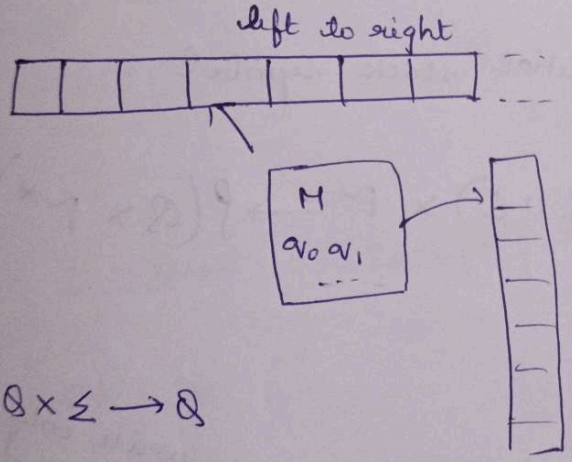
↳ limited amount of memory

stack

↓

DS: LIFO.

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



$\delta : Q \times \Sigma \rightarrow Q$

FA → decide the next state depending on current input & current state.

PDA → decide the next state depending upon & auxillary memory.

$$M = (Q, \Sigma, \delta, q_0, F, \Gamma, z_0)$$

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ finite set of alphabets

$q_0 \rightarrow$ start state

$F \rightarrow$ finite set of final states

F - cannot be empty

There cannot be more than one start state in NFA, DFA & PDA

$\Gamma \rightarrow$ finite set of stack symbols

$z_0 \rightarrow$ initial stack symbol

$$\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow P(Q \times \Gamma^*)$$

$W = aabb$

$\delta(q_0, a, z_0) = \{(q_0, Az_0)\}$ ← write only the top 2 elements of stack ← pushed 'a' into the stack

$\delta(q_0, a, A) = \{(q_0, AA)\}$

$\delta(q_0, b, A) = \{(q_1, \varepsilon)\}$ ↳ poped 'a' from the stack

$\delta(q_1, b, A) = \{(q_1, \varepsilon)\}$

$z_0 \rightarrow$ indicates empty stack

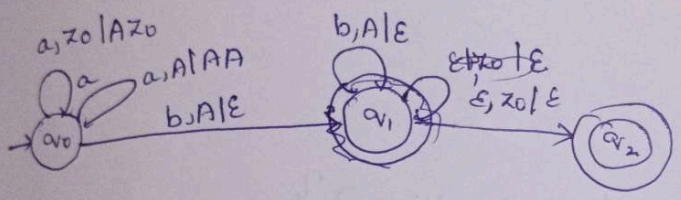
$\varepsilon \rightarrow$ indicates poping

$\delta(q_1, \epsilon, z_0) = \{(q_2, \epsilon)\}$ ← Accept state

$q_1 \in F$ → now there is nothing in the stack

$M \Rightarrow Q = \{q_0, q_1\}$ $\quad q_f = q_2$

$\Gamma = \{A, B, z_0\}$ $\quad q_0 = \{q_0\}$



PDA

$L = \{ w c w^R \mid w \in \{a,b\}^* \}$

NPDA ≈ CFG

DPDA ⊊ CFG

⊆ CFL

a b c b a

a c a

∉

~~b~~ b c b

Languages expected by DPDA

↳ LR-lang

$\delta(q_0, a, z_0) = \{(q_0, A z_0)\}$

$\delta(q_0, a, A) = \{(q_0, AA)\}$

$\delta(q_0, a, B) = \{(q_0, BA)\}$

$\delta(q_0, b, z_0) = \{(q_0, B z_0)\}$

$\delta(q_0, b, A) = \{(q_0, BA)\}$

$\delta(q_0, b, B) = \{(q_0, BB)\}$

$\delta(q_0, c, A) = \{(q_1, A)\}$

$\delta(q_0, c, B) = \{(q_1, B)\}$

$\delta(q_0, c, z_0) = \{(q_1, z_0)\}$

$\delta(q_1, a, A) = \{(q_1, \varepsilon)\}$

$\delta(q_1, b, B) = \{(q_1, \varepsilon)\}$

$\delta(q_1, \varepsilon, z_0) = \{(q_f, \varepsilon)\}$

HW ① NWR

② $\{w \in \Sigma^* / H_a(w) > H_b(w)\}$

① WWR

1. $\delta(q_0, a, z_0) = \{(q_0, Az_0), (q_1, z_0)\}$
2. $\delta(q_0, b, z_0) = \{(q_0, Bz_0), (q_1, z_0)\}$
3. $\delta(q_0, a, A) = \{(q_0, AA), (q_1, AA)\}$
4. $\delta(q_0, a, B) = \{(q_0, AB), (q_1, B)\}$
5. $\delta(q_0, b, A) = \{(q_0, BA), (q_1, A)\}$
6. $\delta(q_0, b, B) = \{(q_0, BB), (q_1, B)\}$
7. $\delta(q_0, \varepsilon, z_0) = \{(q_1, z_0)\}$
8. $\delta(q_0, \varepsilon, A) = \{(q_1, A)\}$
9. $\delta(q_0, \varepsilon, B) = \{(q_1, B)\}$
10. $\delta(q_1, a, A) = \{(q_1, \varepsilon)\}$

⑪ $\delta(q_1, b, B) = \{(q_1, \varepsilon)\}$

⑫ $\delta(q_1, \varepsilon, z_0) = \{(q_f, \varepsilon)\}$

② $\{w \in \Sigma^* \mid n_a(w) > n_b(w)\}$

① $\delta(q_0, a, z_0) = \{(q_0, Az_0)\}$

② $\delta(q_0, a, A) = \{(q_0, AA)\}$

③ $\delta(q_0, b, A) = \{(q_1, \varepsilon)\}$

④ ~~$\delta(q_1, \varepsilon, A) = \{(q_f, A)\}$~~

⑤ ~~$\delta(q_0, \varepsilon, A) = \{(q_f, A)\}$~~

⑥ ~~$\delta(q_0, \varepsilon, z_0) = \{(q_1, z_0)\}$~~

⑦ $\delta(q_0, b, z_0) = \{(q_0, Bz_0)\}$

⑧ $\delta(q_1, a, A) = \{(q_1, A)\}$

$\delta(q_0, b, A) = \{(q_0, \varepsilon)\}$

$\delta(q_0, b, B) = \{(q_0, BB)\}$

$\delta(q_0, a, B) = \{(q_1, \varepsilon)\}$

aaab

baaa

b

a, B

a
a
b

$L = \{a^i b^j c^k \mid i = j \text{ or } k \leq j \leq 2k\}$

Acceptance of PDA:

① Empty store : I/P exhausted and stack is empty

$$L(M) = \{ w \in \Sigma^* / \delta(q_0, w, z_0) \vdash^* (p, \lambda, \lambda) \}$$

② Final state : After reading all the symbols, some symbols are left in stack and M/C has reached to final state.

$$L(M) = \{ w \in \Sigma^* / \delta(q_0, w, z_0) \vdash^* (q_f, \lambda) \}$$
$$\lambda \in \Gamma$$

DFDA vs PDA:

DPDA:

① $\forall q \in Q$, $z_0 \in \Gamma$ if $\delta(q_0, \varepsilon, z_0)$ is non-empty then $\delta(q_0, a, z_0)$ should be empty.

② for $\delta(q_1, a, z) \rightarrow$ only one choice.

stack is

$(q_0, w, z_0) \vdash^* (p, \epsilon)$

symbols,
ε in stack
ed to final

$\vdash^* (q_f, \epsilon)\}$

empty

Pumping lemma for CFL:

Let 'L' be a CFL if $\forall p \geq 0$, s.t $\exists w \in L$ s.t $|w| \geq p$
for all partition of $w = uvxyz$

$|vxy| \leq p$

$|vy| \geq 0$

$\exists i \geq 0$, $uv^i xy^i z \notin L$ $\Rightarrow$ L is not a CFL

$L = \{a^n b^n c^n, n \geq 1\}$ is not CFL.

$\forall p \geq 0$.

$w = a^p b^p c^p$ $\quad |w| = 3p$

$w = a^p b^{p-t-l} b^l b^t c^p$

$w = a^p (b^{p-t-l})^i b^l (b^t)^i c^p$ $\quad t, l \geq 0$

$i = 0$.

$w = a^p b^l c^p$

$w \notin L$

$\therefore$ L is not a CFL.

Turing Machine:

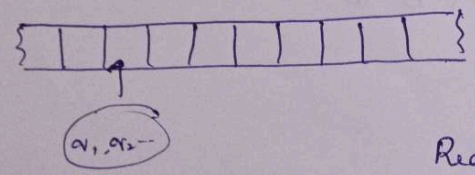
1936

- An abstract computational procedeere, that takes some input and gives some output.

- Procedure → undecidable
  Algorithm → decidable.

- Turing Machine acts → Acceptor
  → Input/Output device

Turing Machine as Acceptor:



⇒ ∞ length tape

Readonly from tape → PDA

Movement of head L→R — PDA

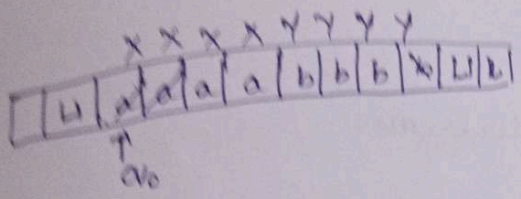
Turing Machine:
Read & Write on Tape
Movement of ~~tape~~ head from L→R & R→L.

$M = (Q, \Sigma, \delta, q_0, \Gamma, \Gamma^*, L)$

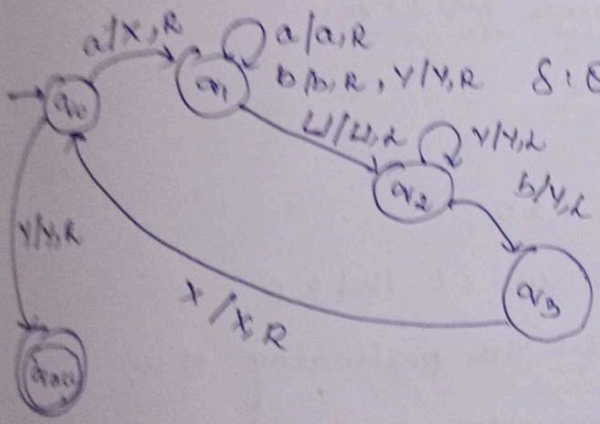
↑ Tape symbol which includes Σ

↳ Number of Γ^*, i.e. $\sqcup$ & Σ

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

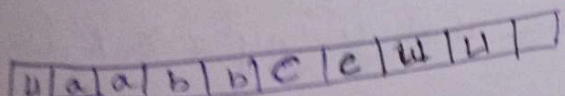


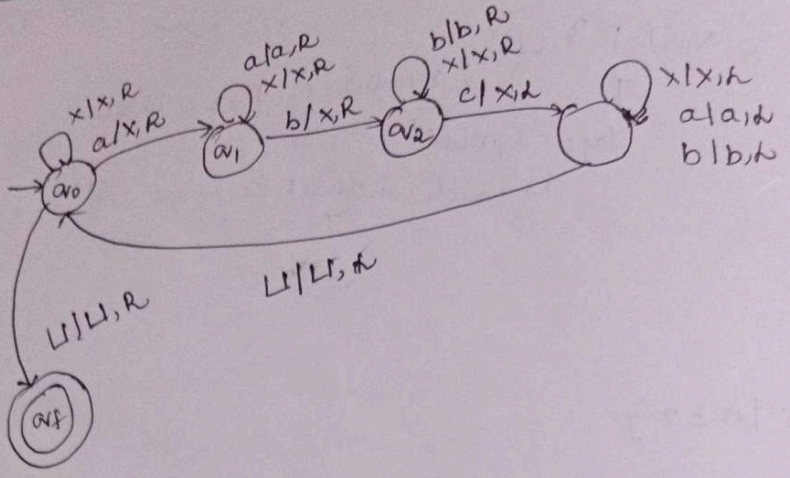
$\delta : Q \times \Sigma \rightarrow Q \times \Sigma$



$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

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





L = set of all palindromes

Pumping lemma for CFL:

Two player game:

P_1: choose $p \geq 0$

P_2: choose w, s.t $|w| \geq p$

P_1: decides the partioning of w

$w = uvxyz$

$|vxy| \leq p$ & $|vy| \geq 0$

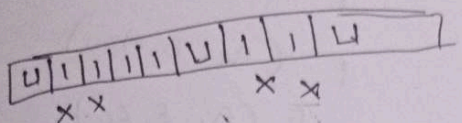
P_2: select the value of i

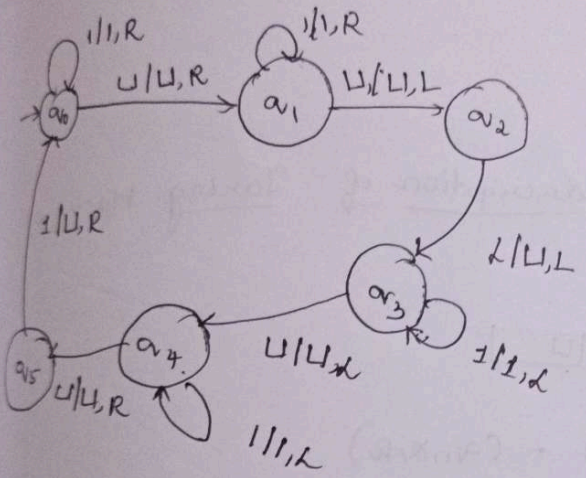
such that $uv^ixy^iz \notin L$ (win)

Turing Machine as Input / Output:

$f(x,y) = x - y$

①





Instantaneous discription of PDA:

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

$\delta(q_0, a, z_0) = (q_0, AZ_0)$

$\delta(q_0, a, A) = (q_1, AA)$

$\delta(q_0, b, A) = (q_1, \varepsilon)$

$\delta(q_1, b, A) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, z_0) = (q_2, \varepsilon)$

$w = a^2b^2$

$(q_0, aabb, z_0) \vdash (q_0, abb, Az_0)$

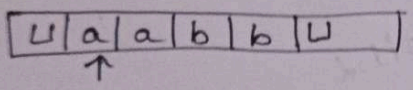
$\vdash (q_0, bb, AAz_0)$

$\vdash (q_1, b, Az_0)$

$\vdash (q_1, \varepsilon, z_0)$

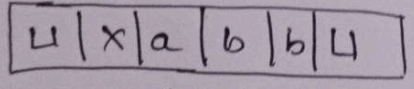
$\vdash (q_1, \varepsilon)$

Instantaneous description of Turing Machine:



① $\delta(q_0, a) = (q_1, X, R)$

②



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

③