Operations on Language :- $\cap, \cup, \bar{L}, L_1 - L_2, L_1 \oplus L_2$

$L_1 = \{01, 100, 00\}$ $L_2 = \{00, 11\}$

- $L_1 \cup L_2 = \{01, 100, 00, 11\}$ atleast from L_1 or L_2
- $L_1 \cap L_2 = \{00\}$
- $(\bar{L} = \Sigma^* - L)$ $(\Sigma = \{0, 1\})$
- $\bar{L_1} = \{ \epsilon, 0, 1, 00, 01, 10, 11, 100 \ldots \}$
- $L_1 - L_2 = L_1 \cap \bar{L_2}$ ($\bar{L_2}$ is not easy to calculate)

  $= L_1 - (L_1 \cap L_2)$
- $L_1 \oplus L_2 = (L_1 - L_2) \cup (L_2 - L_1)$

  exactly from L_1 exactly from L_2

Q. $|L_1| = 10$, $|L_2| = 12$ and $|L_1 \cap L_2| = 5$

$|L_1 \cup L_2| = |L_1| + |L_2| - |L_1 \cap L_2| = 10 + 12 - 5 = 17$

$|L_1 \oplus L_2| = |L_1| + |L_2| - 2|L_1 \cap L_2| = 10 + 12 - 10 = 12$

$|L_1 - L_2| = |L_1| - |L_1 \cap L_2| = 10 - 5 = 5$

$|L_2 - L_1| = |L_2| - |L_1 \cap L_2| = 12 - 5 = 7$

$(|\Sigma^*| = \infty)$

⇒ Reversal and concatenation of strings :-

$(L^R = \{w^R \mid w \in L\})$ Reversal

- $L = \{001, 10, 11\}$

  $L^R = \{100, 01, 11\}$

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

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

- $L = (a+b)^* abb^* bc(a+ab)^*$

  $L^R = (a+ba)^* cbb^* ba(a+b)^*$

Q: Check true or false ?

$L = L^R$ False (some time true)

$|L| = |L^R|$ True

Q: Check True or False :

$\{L = L^R \mid L \text{ contain pallindromes}\}$

It is false as if $L = \{01, 10\}$ then $L^R = \{10, 01\}$ which implies $L = L^R$ and it doesn't contains pallindromes.

$L_1 \cdot L_2 = \{uv \mid u \in L_1 \ \& \ v \in L_2\}$ concatenation

eg:- $L_1 = \{100, 01, 10\}$ $\{L_2 = 01, 001\}$

$L_1 \cdot L_2 = \{10001, 100001, 0101, 01001, 1001, 10001\}$

$L_2 \cdot L_1 = \{01100, 0101, 0110, 001100, 00101, 00110\}$

so, $(L_1 \cdot L_2 \neq L_2 \cdot L_1)$ not commutative

NOTE:- $(|L_1 \cdot L_2| \neq |L_2 \cdot L_1|)$

as, $|L_1 \cdot L_2| \leq |L_1| \times |L_2|$

duplicate items are eliminated

$L_1 = \{a^n \mid n \geq 0\}$ then $L_1 \cdot L_2 = \{a^m b^n \mid m, n \geq 0\}$

$L_2 = \{b^n \mid n \geq 0\}$

$\{a^0, a^1, a^2, \ldots\}$

$\{b^0, b^1, b^2, \ldots\}$

Power of an element :-

$L^0 = \{\epsilon\}$

$L^1 = \{L\}$

$L^2 = L \cdot L$

$L^3 = L \cdot L^2$

$L^4 = L \cdot L^3$

$\emptyset^0 = \{\epsilon\}$

$L \cdot L = L^2 = \{uv \mid u \in L \text{ and } v \in L\}$

$|L|$ ← no. of strings in lang

$|\omega|$ ← length of string

$\emptyset = \{\}$ ← null set

eg:-

$L = \{10, 01\}$

$L^0 = \{\epsilon\}$, $L^1 = L = \{10, 01\}$

$L^2 = L \cdot L = \{1010, 1001, 0110, 0101\}$

$L^3 = L \cdot L^2 = \{101010, 101001, 100110, 100101, 011010, 011001, 010110, 010101\}$

$L = \{10, 01, \epsilon\}$

(1001 can belong to $L^n \mid n \geq 2$)

as $1001 \cdot \epsilon \quad L^3$

$1001 \cdot \epsilon\epsilon \quad L^4$

⋮ ⋮

⇒ Power of infinite lang :-

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

$L^m = \{a^n, n \geq 0\}$

as, $L = \{ \epsilon, a, aa, aaa, aaaa, \ldots\ldots \}$

$L = \{ \epsilon, a, aa, aaa, aaaa, \ldots\ldots \}$

- $a^m . a^n \mid m, n \geq 0$
- $a^{m+n} \mid m, n \geq 0$
- $a^n \mid n \geq 0$

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

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

⇒ L^* (Kleen closure) :-

$(L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots\ldots)$

$L^* = \{ \epsilon, 01, 10, 0101, 0110, \ldots\ldots \}$

L^+ (positive closure) :-

$$\left(L^+ = L^1 \cup L^2 \cup L^3 \dots\dots\right)$$

✓ $\epsilon \in \epsilon \in L^*$ as $L^0 \subseteq L^*$

✗ $\epsilon \in \not\epsilon \in L^+$ as if L contains ϵ then $\epsilon \in \epsilon \in L^+$

Q. Check True or False :-

(a) $\epsilon \in \epsilon \in \phi^0$ (T)

(b) $\epsilon \in \epsilon \in L^*$ (T)

(c) $\epsilon \in \epsilon \in \Sigma^*$ (T)

(d) $\epsilon \in \not\epsilon \in \Sigma^+$ (T)

(e) $\epsilon \in \not\epsilon \in L^+$ (F)

as Σ can't contains ϵ so,

$$\Sigma^+ = \Sigma^* - \epsilon$$

then Σ^+ will not contain ϵ

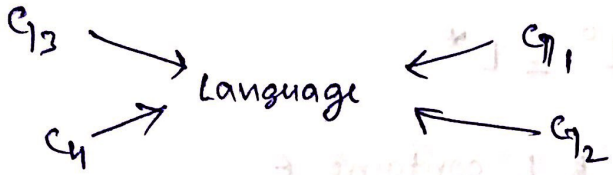
Q. check $\overline{L^*} = (\overline{L})^*$

as L^* always contains ϵ then, $\overline{L^*}$ will not contains ϵ, but $(\overline{L})^*$ will contain ϵ

so, $\left(\overline{L^*} \neq (\overline{L})^*\right)$

⇒ Grammer :- Formal representation of language

(G1) Grammer ⟶ Language (specific)



It ~~is~~ has set of production rules, through which can derive language members.

$G = \{V, T, P, S\}$

- V: set of variables (Non-terminals) (finite)
- T: set of Terminals (finite)
- P: set of Production rules (finite)
- S: start variable ($S \in V$)

eg:- In English Grammer:

Noun ⟶ dog | cat | human

Noun: variable — it can be expanded further to terminals

dog | cat | human: terminal

eg:- $S \rightarrow aS \mid \epsilon$ — Right recursion

$S \rightarrow Sa \mid \epsilon$ — left recursion

$S \rightarrow ab$ — No recursion

NOTE:- If grammer doesn't contain recursion then language must be finite.

But if grammer contains recursion then language may or may not be infinite.

L(G):-

Language defined by Grammer

$$G \rightarrow L(G)$$

eg:-

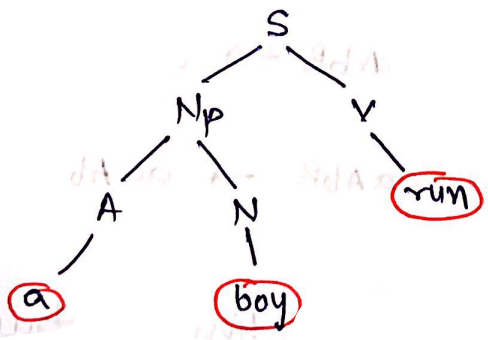
sentence — $S \rightarrow N_p V$ — verb

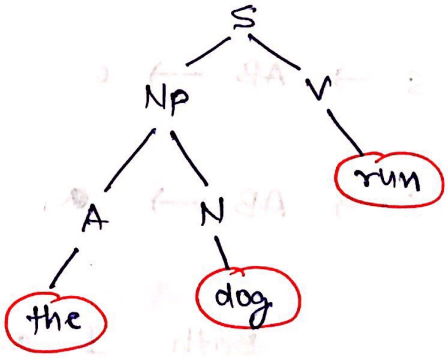
Noun phrase — $N_p \rightarrow AN$

Noun — $N \rightarrow$ boy | dog

article — $A \rightarrow$ a | the

$V \rightarrow$ run | write





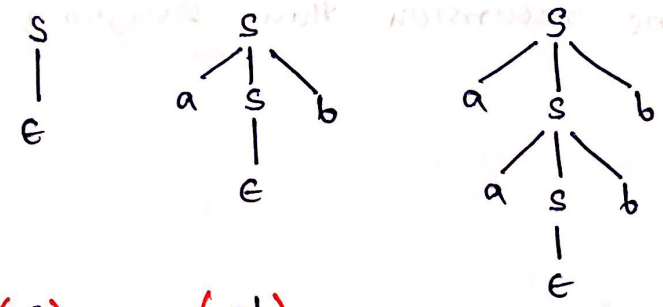
NOTE:- Compile checks grammatical errors.

a boy run ✓

a run boy ✗

eg:- Q: $S \rightarrow \epsilon$

$S \rightarrow aSb$ (recursion)



(ϵ) (ab) $(aabb)$

so, $L(G) = \{ a^n b^n \mid n \geq 0 \}$

(infinite language)

eg:- Q: $S \rightarrow AB$

$A \rightarrow aaA \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

$L(G) = (aa)^* b^*$

$= \{ a^{2m} b^n \mid m, n \geq 0 \}$

Derivation of $(\omega = aab)$:-

I: $S \rightarrow AB \rightarrow aaAB \rightarrow aaAbB \rightarrow aabB \rightarrow$ "aab"

II: $S \rightarrow AB \rightarrow AbB \rightarrow aaAbB \rightarrow aaAb \rightarrow$ "aab"

Both I and II derivation are true.

* Compiler stops when it finds a correct derivation it will not find further derivation.

Rules of derivation :-

(i) Start from production rule of start symbol 'S'.

(ii) Expand one variable at a time

(iii) Continue expansion until all terminals are left.

Sentential form :- $(V \cup T)^*$ i.e any combination of variable and terminal which can be derived from rules.

Sentence :- T^* every sentence is in sentential form But vice-versa is not possible.

NOTE:

Derivation $\longleftrightarrow$ Syntax

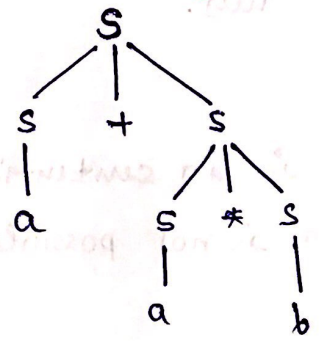
Derivation tree $\longleftrightarrow$ Semantics

- For unambiguous grammer (G) $\longrightarrow$ single derivation tree for $w \in L(G)$
- But for ambiguous grammer (G_A) $\longrightarrow$ multiple derivation tree for $w \in L(G_A)$

eg:- G: (Ambiguous grammer)

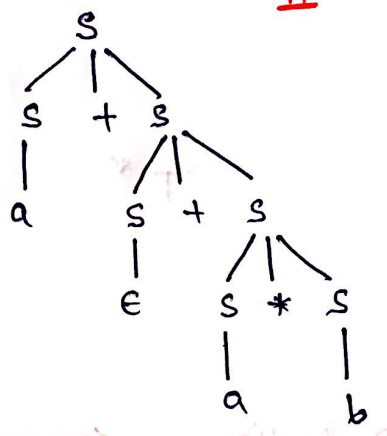
$S \rightarrow S+S \mid S*S \mid a \mid b \mid \epsilon$

I



a + (a*b)

II



a + (a * b)

NOTE:-

* G is ambiguous if for any one $w \in L(G)$ has more than one derivation tree

* G is unambiguous if all $w \in L(G)$ must have specific derivation tree

⇒ Equivalence of Grammer :-

$$G_1 \equiv G_2 \quad \text{if} \quad L(G_1) = L(G_2)$$

eg:-

G_1 : $S \to asb \mid \epsilon$

G_2 : $S \to asb \mid aasbb \mid \epsilon$

G_3 : $S \to asbb \mid \epsilon$

so, $(G_1 \equiv G_2)$ $(G_1 \not\equiv G_3)$

eg:-

G_1 : $S \to as \mid \epsilon$

G_2 : $S \to sa \mid \epsilon$

G_3 : $S \to a \mid ss \mid \epsilon$

$$(G_1 \equiv G_2 \equiv G_3)$$

Types of Grammer :-

(i) TYPE-0 :- Unrestricted grammer, phrase structure grammer, formal grammer. It's O/P is RE language

only restriction on production rule :-

(atleast one variable $\to$ Terminal | variable)

RHS RHS

eg:-

$S \rightarrow aAB \mid a$

$A \rightarrow aA \mid aB$

$\underbrace{aA}_{2} \rightarrow \underbrace{B}_{1}$ (contraction)

(ii) Type-1 :- Non-contracting grammer, Context sensitive grammer

$aA \rightarrow B$ ✗

$AB \rightarrow ab$

$A \rightarrow aB$ ✓

$S \rightarrow \epsilon$ ✓ (exception)

It's o/p is CSL (context sensitive Language)

(iii) Type-2 :- CFG (context free grammer)

Only one variable at L.H.S.

$aA \rightarrow ab$ ✗

$AB \rightarrow ab$ ✗

$A \rightarrow ab$ ✓

It's o/p is CFL.

(iv) TYPE-3 :- Regular grammer.

Right linear $V \rightarrow T^*V + T^*$

else

Left linear $V \rightarrow VT^* + T^*$

{either of the two not both}

$S \rightarrow aA \mid aB$

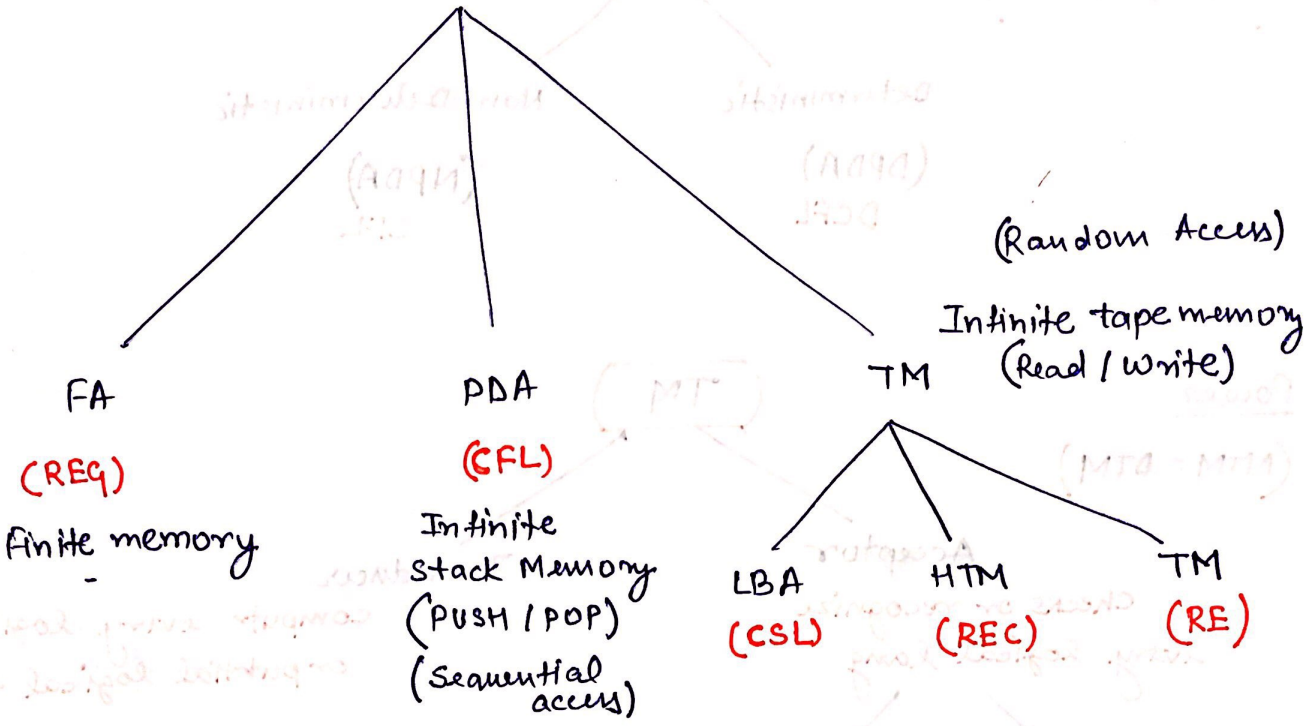
$A \rightarrow Bb \mid b$ ✗

$S \rightarrow aA \mid a$

$A \rightarrow aB \mid a$

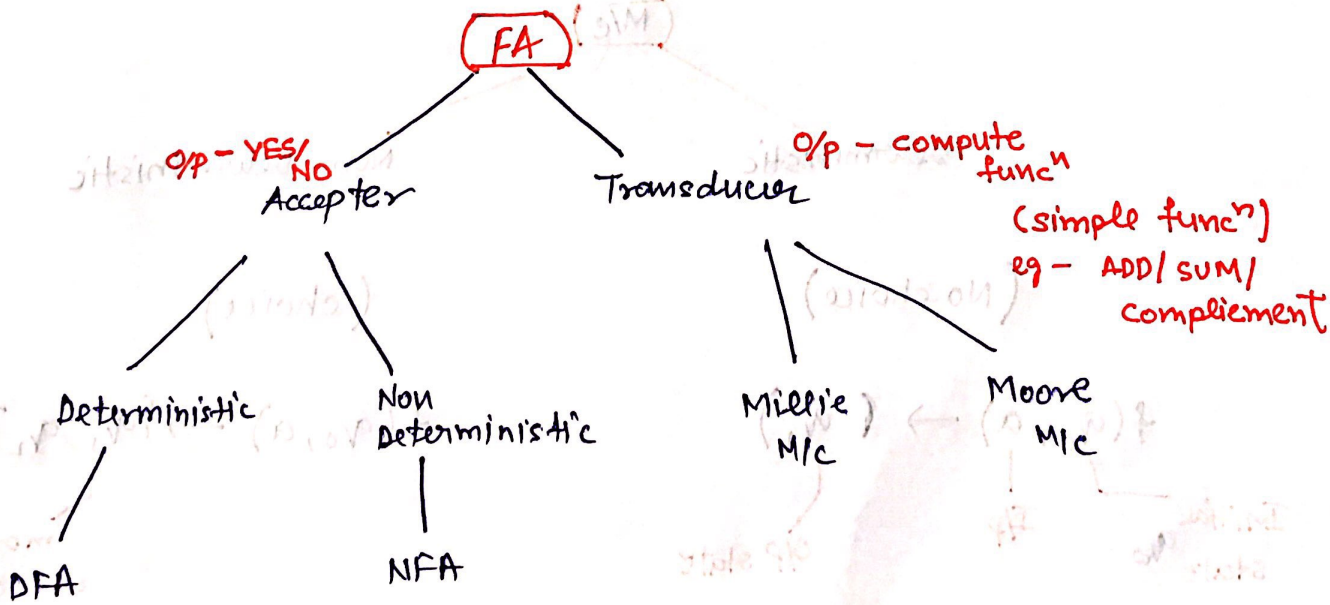
$B \rightarrow b$ ✓

⇒ Machines :-



Grammer ⟶ generate language

Machine ⟶ recognize language (ACCEPT / REJECT)

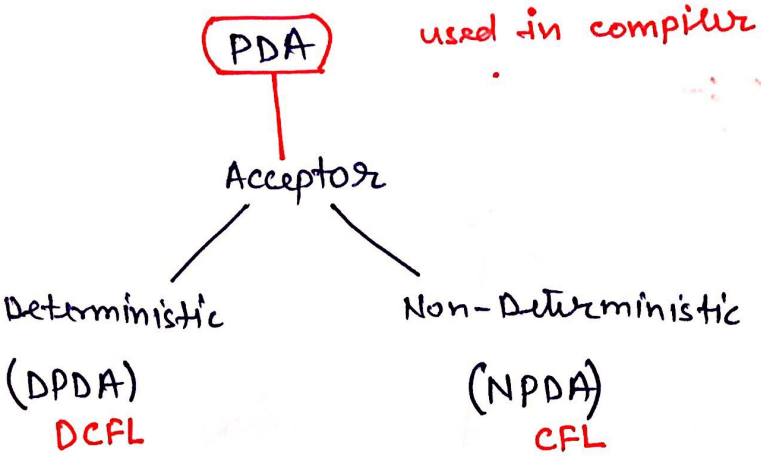


Power :- (DFA = NFA)

PDA — used in compiler (CFG)

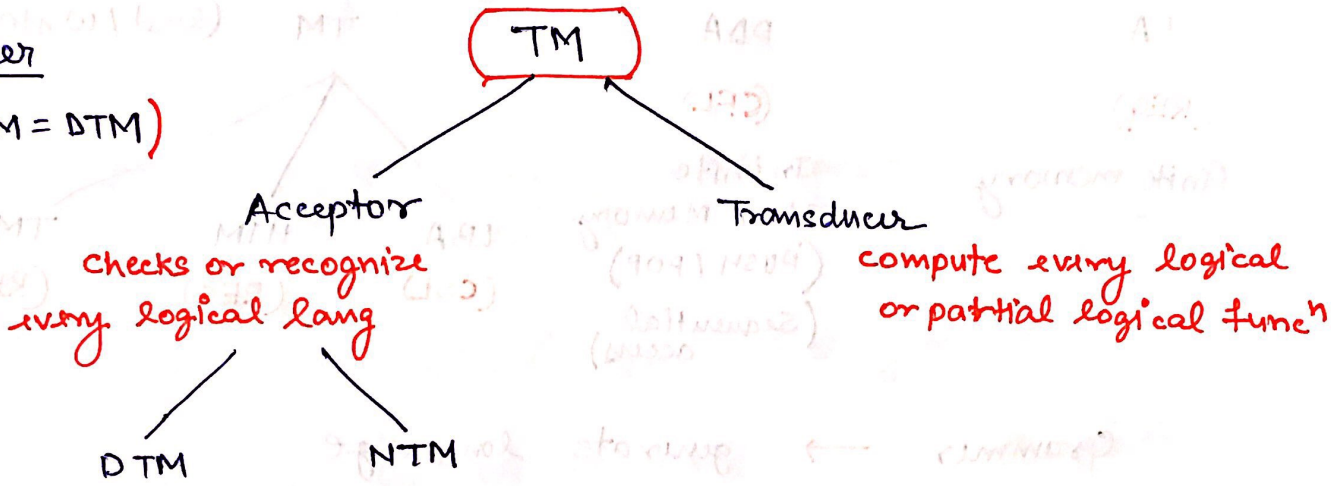
Power

(NPDA > DPDA)

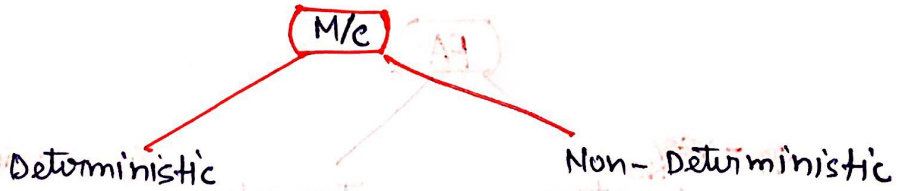


Power

(NTM = DTM)



⇒ Deterministic & Non-Deterministic M/c :-



(No choice)

$f(q_0, a) \rightarrow \{q_1\}$

Initial state — I/p — O/p state

(choice)

$f(q_0, a) \rightarrow \{q_1, q_2\}$

more than one O/p state