Date:
Page:

Theary of Computation (20CST-353)

Unit: 01

Chapter 01 - Introduction to Basic Terminology

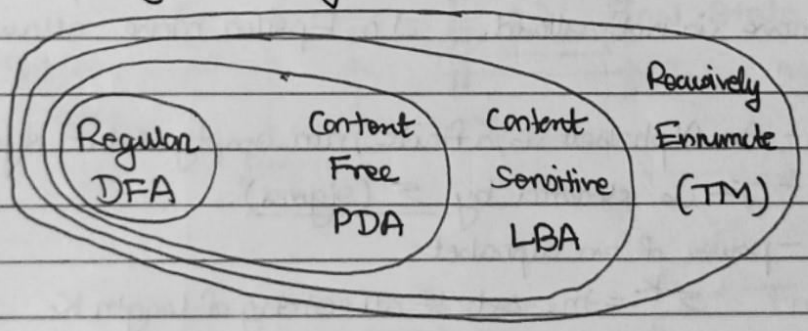
* Automata Theary - Study of abstract computing clevices or machines

* Father of modern computer Science - Alan Turing (1912-54)

* Languages - "A language is a collection of sentence of finite length all constructed from a finite alphabet of symbols".

* Grammar - "A Grammar can be regulated/regarded as a clevice that enumerates the sentence of language."

* Chomsky Hierarchy -



—Keypoint—
Languages & Grammars are coined by Chomsky (1959)

Finite Automata

* It is the simplest machine to recognise pattern.
* It takes the string of symbol as input & changes it accordingly.
* 02 states - Accept State & Reject State.
When the input string is processed successfully & the automata reached its final state, it will accept.
* Formal Defenition -
Finite Automata is a collection of 5 tuples -
Q: Finite set of states.
Σ: Finite set of input symbol
q0: Initial State
F: Final State
δ: Transition State

Date:
Page:

	DFA	NFA
1.	DFA stands for - Deterministic Finite automata.	1. NFA stands for - Non Deterministic Finite Automata.
2.	DFA can't use Empty String Transition.	2. NFA uses Empty String Transition.
3.	DFA is more difficult to construct.	3. NFA is easier to construct.
4.	Time needed for executing an input string is less.	4. Time needed for executing an input string is more.
5.	All DFA are NFA.	5. Not all NFA are DFA
6.	DFA requires more space.	6. NFA require ~~less~~ space.
7.	Dead state may be required.	7. Dead state is not required
8.	Backtracking is allowed.	8. Backtracking is not allowed.
9.	Conversion of Regular Expr. to DFA is Difficult.	9. Conversion of Regular Expr. to NFA is Simpler.
10.	Epsilon move is not allowed.	10. Epsilon move allowed.

* Alphabet - An Alphabet is a finite, non empty set of symbols.
  - It is denoted by Σ (sigma).
  - power of an alphabet:
    Σ^k = the set of all string of length k
    $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots\ldots$

* Strings - An string or word is a finite sequence of symbol chosen from Σ.
  - It is denoted by ϵ (Epsilon)

* Languages - Language "L" is a said to be an language over alphabet Σ only if $L \subseteq \Sigma^*$.
  → Reason - this because Σ^* is set of all strings.

—Key point—
Length of String denoted by "|w|"

Date:
Page:

Set Operations on Languages

* **Union -**

A String $x \in L_1 \cup L_2$, if $x \in L_1$ or $x \in L_2$.

* **Intersection -**

A String $x \in L_1 \cap L_2$, if $x \in L_1$ and $x \in L_2$.

* **Complement -**

Σ^* be the universal set.

∴ The complement is $L(bar) = \{x \in \Sigma^* \mid x \notin L\}$.

* **Reversal of Language -**

Denoted by L^R is defined as $L^R = (w^R \mid w \in L)$

* **Language concatenation -**

The concatenation of Languages L_1 & L_2 is defined as

$L_1 L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$.

Deterministic Finite Automata (DFA) -

$(Q, \Sigma, \delta, q_0, F)$ — Final State

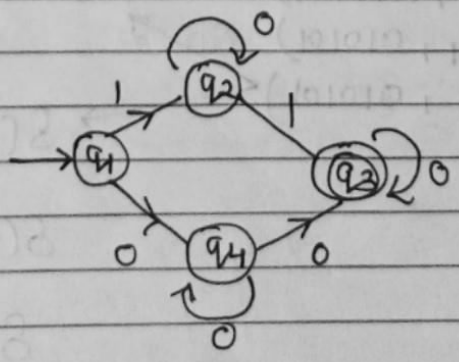
Set of States → Initial State

Transition Function → $Q \times \Sigma \to Q$

Set of inputs

Transition Diagram (TD)	Transition Table (TT)
Initial → ○	→
Final ◎	○

Ex:



Stages	Input/Output 0	1
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_3	q_4
q_4	q_4	q_4

Date:
Page:

Numerical- Acceptance of string or Not -
String - 101101

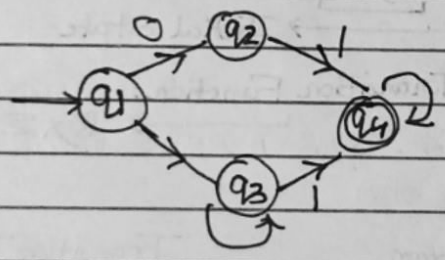
Sol- Step 1: $\delta(q_1, 101101)$ → Initial State
Step 2: $\delta(q_2, 01101)$
Step 3: $\delta(q_3, 1101)$
Step 4: $\delta(q_4, 101)$
Step 5: $\delta(q_4, 01)$
Step 6: $\delta(q_4, 1)$
Step 7: $\delta(q_4)$ → Final State

Initial State + Final State = Acceptance
Otherwise Not Acceptance

Non-Deterministic Finite Automata (NDFA)-

$(Q, \Sigma, \delta, q_0, F)$ → Final State
Set of states → Initial State
Set of Input
Transition Function
$Q \times \Sigma \to 2^Q$

Ex:



Stages	Input/Output 0	1
q_1	q_3, q_2	q_1
q_2	q_4	—
q_3	q_3	q_4
q_4	q_4	q_4

Numerical- Acceptance of string or Not -
String: 101010 1

Sol: Step 1: $\delta(q_1, 1010101)$
Step 2: $\delta(q_1, 010101)$
Step 3: $\delta(q_1, 010101)$ → $\delta(q_2, 10101)$, $\delta(q_3, 10101)$

$\delta(q_3, 10101)$
↓
$\delta(q_4, 0101)$
↓
$\delta(q_3, 1010)$

Remember: When the initial State (q_1) & final State (q_4) comes in the between the step, then the String is acceptance or otherwise not.

Formal Language –

Grammar: $G_1 \{V_N, \Sigma, P, S\}$ — Production: Set of rules

P: Set of Production
V_n: Variable
Σ: Terminal
S: Start of production

Rule of production –

Rule 01: LHS is only variable.

Rule 02: Ex: $S \to aCa$, $C \to b$ — replacement

but reverse of $b \to a$ is not allowed.

Rule 03: If Terminal is not presented in RHS then formal Language is not applicable

Key point →
* Variable: Capital ABC...
Terminal: small a, b, c + Variable
* $L(G_1)$: Language of Grammar

Numerical – If $G_1 = \{\{S\}, \{0,1\}, \{S \to 0S1, S \to \Lambda\}, S\}$

Solution – S: Starting of production – S

V_n: $\{S\}$

Σ: $\{0,1\}$

P: $S \to 0S1$

$S \to \Lambda$ – Null value

A.T.Q

Step 1: $S \to \Lambda \in L(G_1)$ (always 1st step in case of Null)

Step 2: $S \to 0S1$

$\to 00S11$ $\{\because S \to 0S1\}$

$\to 000S111$ $\{\because S \to 0S1\}$

↓ (n-1 times work)

$\therefore 0^n S 1^n$

$\to 0^n 1^n$, $n \geq 1$ Language of Grammar

$\to 0^n 1^n \in L(G_1)$ ⟶

Date:
Page:

Numerical – If $G_1 = \{S, C\}, \{a, b\}, P, S$ where P consists of $S \to aCa$

$C \to aCa \mid (a/b)$ find $L(G_1)$?

Solution–

Step 1: $S \to aCa$

Step 2: $S \to aaCaa$ ($\because C \to aCa$)

Step 3: $S \to aaaCaaa$ ($\because C \to aCa$)

↓ (n−1) time work

Step 4: a^nCa^n, $n \geq 1$

Step 5: $a^nba^n \in L(G_1)$, $n \geq 1$ ($\because C \to b$)

★ Keypoint:
$C \to aCa \mid (a/b)$
↓
$C \to aCa$
$C \to b$

Operations on Language –

I. Union
II. Concatenation
III. Transpose
IV. Intersection

All bear $L(G_1)$

Not bear $L(G_1)$

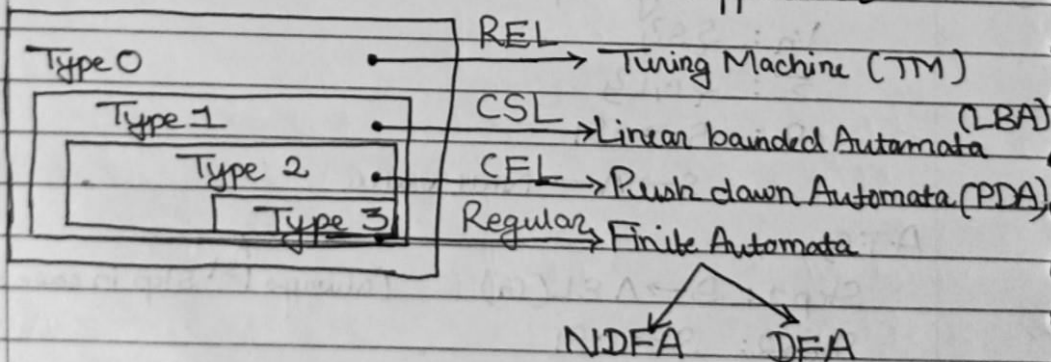
★ Grate Question Asked

★ Note point
$L(G_1) = \phi$
$\because$ RHS have only variable

Types of Language –

There are 04 types of Language are:

— Application —



★ Grate Question Asked

Other names –

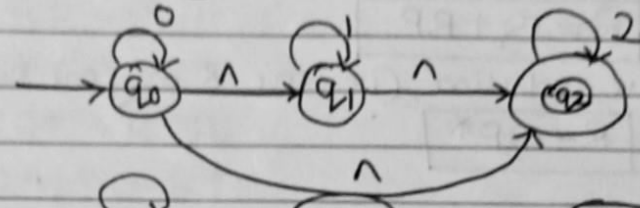
Type 0	RFL: Recursive Enumerable Language
Type 1	CSL: Content Sensitive Language
Type 2	CFL: Content Free Language
Type 3	Regular Regular Language

Null Moves (∧ moves) - 2

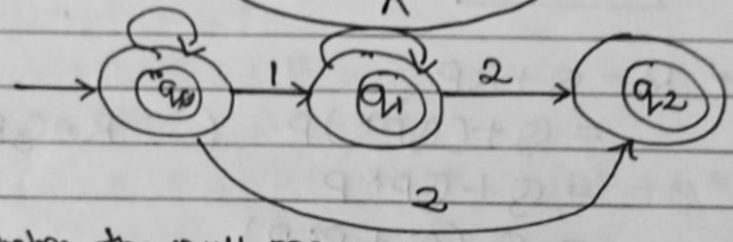
Rule 01: (u) —∧→ (v), if u: initial then make v: initial

Rule 02: Check arrow incoming to v, then replace v: 2 to null value i.e. ∧ = 2.

Rule 03: if (v): final, then make (u): final.

Numerical - 

using Null Moves (∧)?

Solution - 

remember the null moves -

i. if u: initial then make v: initial.

ii. replace the value of v with with null value.

iii. if v: final then make u: final.

Regular Expressions -

They are useful for representing certain sets of Strings in an algebraic Fashion.

Rules -

1. Any terminal Symbol (i.e. an element of Σ), ∧ & ∅ are regular Expressions.
2. The Union of two regular Expressions R_1 & R_2 written as $R_1 + R_2$ written as $R_1 | R_2$ is also regular Expression.
3. The Concatenation of two regular Expression R_1 & R_2 written as $R_1 R_2$ is also regular Expression.
4. The closure of a regular expression R, written as R^*, is also a regular Expression.

Note → $R_1 + R_2$ Union

$R_1 \cdot R_2$ Concatenation

R^* → Closure

**Note → when we take common in math like (1+_) in TOC it is (Λ+_)
Λ → Null value

Date:
Page:

**Very Imp

Arden's Theorem –

Let P and Q be two regular Expressions over Σ. If P does not contain Λ, then following equation in R, viz:

$$R = Q + RP$$

has unique solution (i.e. one & only one solution) given by $R = QP^*$

— Note Point —

- $a^* = \Lambda, +a, +aa, \ldots$
- $a^+ = a, aa, aaa, \ldots$
- $a^+ = a^* - \Lambda$

Proof →

$$R = Q + RP$$
$$= Q + (QP^*)P \quad (\because R = QP^*)$$
$$= Q + QP^*P$$
$$= Q(\Lambda + P^*P)$$
$$= QP^* \quad (\because \Lambda + R^*R = R^*)$$

Proof uniqueness →

$$R = Q + RP$$
$$= Q + (Q+RP)P \quad (\because R = Q+RP)$$
$$= Q + QP + RP^2$$
$$= Q + QP + (Q+RP)P^2 \quad (\because R = Q+RP)$$
$$= Q + QP + QP^2 + RP^3$$

↓ i times

$$= Q + QP + QP^2 \ldots QP^i + RP^{i+1}$$
$$= Q(\underbrace{\Lambda + P + P^2 \ldots P^i}_{P^*}) + RP^{i+1}$$
$$= QP^* + RP^{i+1}$$

★ We know $\Lambda + R^*R = R^*$

As,

P doesn't contain Λ, RP^{i+1} has no string of length less than $i+1$ so 'L' does not contain in set RP^{i+1}.

$$\therefore R = QP^*$$

Hence proved...

→ Always Remember

Date:
Page:

Identities for Regular Expressions

I_1: $\emptyset + R = R$

I_2: $\emptyset R = R\emptyset = \emptyset$

I_3: $\wedge R = R \wedge = R$

I_4: $\wedge^* = \wedge$ and $\emptyset^* = \wedge$

I_5: $R + R = R$

I_6: $R^* R^* = R^*$

I_7: $RR^* = R^* R$

I_8: $(R^*)^* = R^*$

*I_9: $\wedge + RR^* = R^* = \wedge + R^* R$

I_{10}: $(PQ)^* P = P(QP)^*$

I_{11}: $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$

*I_{12}: $(P+Q)R = PR + QR$ and
$R(P+Q) = RP + RQ$

→ Most Time used identities

Numerical- Prove:

$$(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1) = 0^*1(0+10^*1)^*$$

Solution- LHS: $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

$= (1+00^*1)(0+10^*1)^*$ ($\because \wedge + RR^* = R^* = \wedge + R^*R$)

$= (\wedge + 00^*)1(0+10^*1)^*$ ($\because (P+Q)R = PR+QR$, $R(P+Q) = RP+RQ$)

$= 0^*1(0+10^*1)^*$ ($\because \wedge + RR^* = R^* = \wedge + R^*R$)

$= RHS$

$LHS = RHS$

Hence proved.

Date:
Page:

VIMP

Kleene's Theorem-

The class of regular sets over Σ is the smallest class R containing $\{a\}$ for every $a \in \Sigma$ and closed under union, concatenation & closure.

Proof →

The set $\{a\}$ is represent by the regular Expression a.

So,

$\{a\}$ is regular for every $a \in \Sigma$.

As the class of regular sets is closed under union, concatenation and closure. R is contained in the class of regular sets.

Let,

L be a regular set

Then

$L = T(M)$ for some DFA

$\therefore$

$M = (\{q_1, \dots q_m\}, \Sigma, \delta, q_0, F)$

$\therefore$

$$L = \bigcup_{j=1}^{n} P_1^m f_j$$

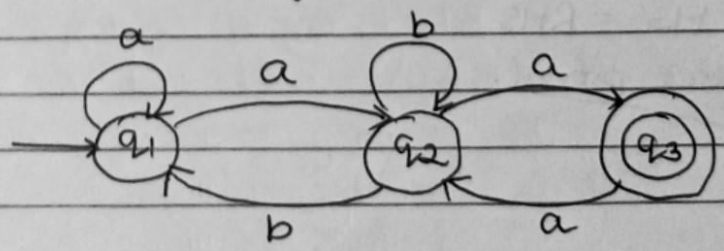
where $F = \{q_{f_1} \dots q_{f_n}\}$ &

$P_{1 f_j}^m$ obtained by applying union, contenation closure to slingletons in Σ.

Thus

L is in R.

Numerical- Prove that the strings are $(a + a(b+aa)^*b)^* a(b+aa)^*a$



Solution- Write the states from incoming arrows are coming to the states.

$\therefore q_1 = q_1a + q_2b + \wedge \rightarrow$ (I) Null comes when the intial comes always.

$q_2 = q_1a + q_2b + q_3a \rightarrow$ (II)

$q_3 = q_2a \rightarrow$ (III)

put the value of eq (III) in eq (II)

$\Rightarrow q_2 = q_1a + q_2b + q_2aa$

$q_2 = q_1a + q_2(b + aa)$ (R = Q, P)

by applying Arden Theorem $\therefore R = Q + RP$, $R = QP^*$

$\Rightarrow q_2 = q_1a(b+aa)^* \rightarrow$ (IV)

put the value of eq (IV) in eq (I)

$\Rightarrow q_1 = q_1a + q_1a(b+aa)^*b + \wedge$

$q_1 = q_1(a + a(b+aa)^*b + \wedge)$ (R = R, P, Q)

by applying Arden Theorem $\therefore R = Q + RP$, $R = QP^*$

$\Rightarrow q_1 = \wedge(a + a(b+aa)^*)b)^* \rightarrow$ (V)

put the value of eq (III) in eq (V)

$\Rightarrow q_3 = q_2a$

$q_3 = q_1a(b+aa)^*a$

$q_3 = \wedge(a + a(b+aa)^*b)^* a(b+aa^*)a$ (R = Q, P)

by applying Arden Theorem $\therefore R = Q + RP$, $R = QP^*$

$\Rightarrow q_3 = \wedge(a + a(b+aa)^*b)^*a \rightarrow$ (VI)

put the value of eq (VI) in eq (V)

$\Rightarrow q_2 = q_1(a + a(b+aa)^*b + \wedge)$

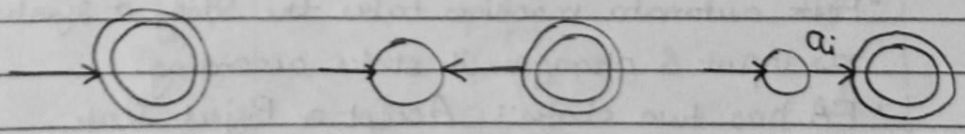
$q_2 = \wedge(a + a(b+aa)^*b)^*(a + a(b+aa)^*(b+\wedge))$ (R = Q, P, R)

$\Rightarrow q_2 = (a + a(b+aa)^*b)^*a(b+aa)^*a$

$\therefore$ The q_3 is a final state, the set of strings recognised by the graph is given by

$(a + a(b+aa)^*b)a(b+aa)^*a$.

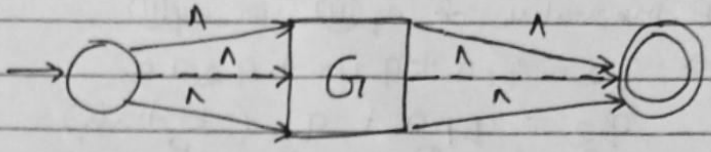
Transition System & Regular Expressions-



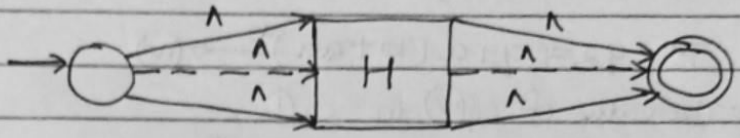
$R = \wedge$ $R = \phi$ $R = a_i$

Date:
Page:

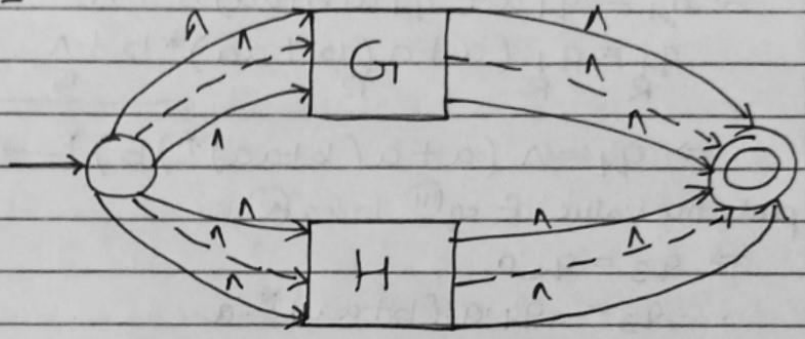
For P –



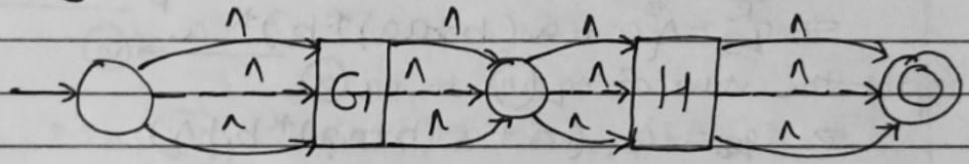
For Q –



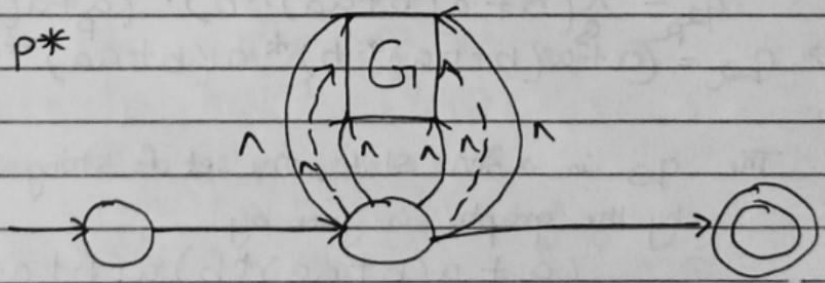
For P+Q –



For PQ –



For P*



Finite State Machine –

- It is used to recognise pattern.
- Finite automata machines takes the String of symbol as input & changes it states according.
- FA has two states : Accept or Reject State.
- Advantages – I. They are Flexible
  II. Low processor overhead.
  III. Easy determination of readability of a state.

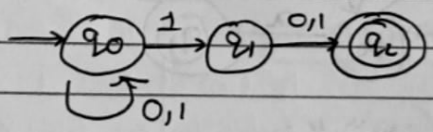
Date:
Page:

- Disadvantages - I. The implementations of huge system using FSM is hard for managing without any idea or design
  II. Not applicable for all domains.
  III. The order of state conversions are inflexible.

- Limitations of Finite State Machine -
  I. FSM level properties.
  II. State properties.
  III. Resolution Action properties
  IV. Transition properties
  V. Event Type properties
  VI. Event content properties.

Conversion of NFA to DFA →

Q.1) NFA of all binary strings in which 2nd last bit is 1



Sol:

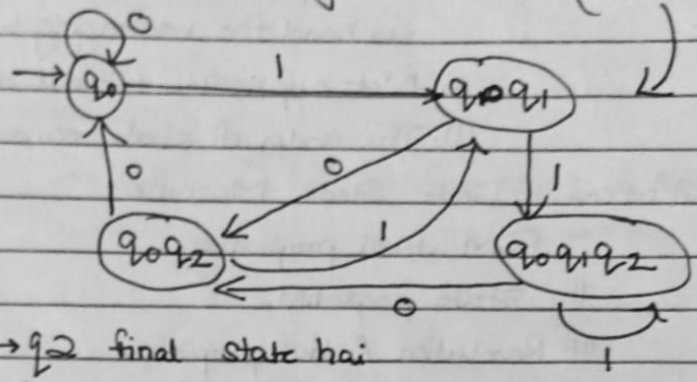
Step1: Draw Transition table

	0	1
→q0	q0	q0q1
q1	q2	q1
final q2	-	-

Step2:

	0	1
→q0	q0	q0q1
q0q1	q0q2	q0q1, q2
q0q2	q0	q0q1
q0q1q2	q0q2	q0q0, q2

Step 3: Draw Transition Diagram {DFA}

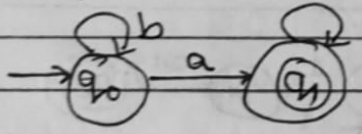


NFA → q_2 final state hai
∴ DFA → q_2 bi final State hai

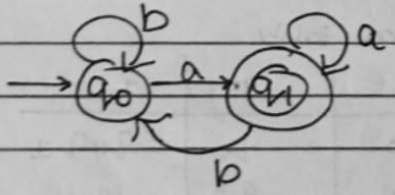
Construction of DFA -

Q.1) Const. a DFA which accepts of all strings containing 'a' & end with 'a' ?

Sol: (1) {a, aa, aa, ... } containing 'a'

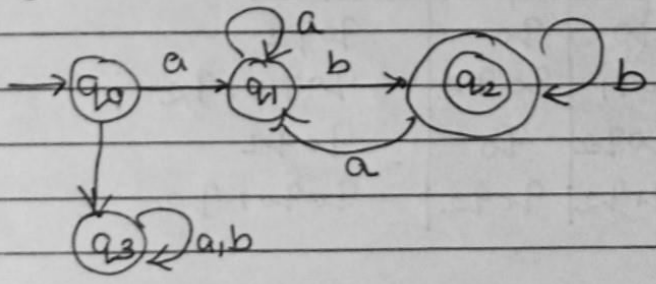


(ii) Ends with "a":



Q.2) Const. a DFA which accepts of all strings ~~starts~~ with 'a' & ending with 'a' ?

Sol- String- {ab, aabb, ... }



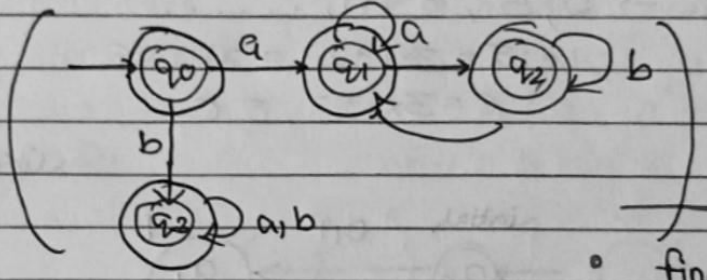
Q.3) Construct a DFA which accept a lang. of all Strings not starting with 'a' (A) & not ending with 'b' (B) ?

Sol- using Demorgan's Law

$$(A \cup B)^c = A^c \cap B^c$$

$$= (\text{not starting with A})^c \cap (\text{not ending with B})^c$$

$$= \text{Starting with A and ending with B.}$$

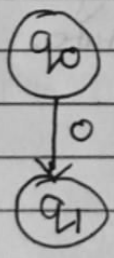


Trick

- final → Non final
- Non final → final.

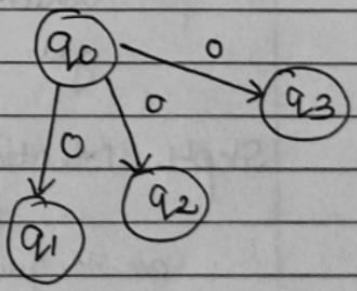
Identification of NDFA & DFA

DFA



if we have only single move is called DFA.

NDFA



if we have multiple choices of move is called NDFA.

Date:
Page:

W
IMP
Numerical-
Sol:

Construct FA $|W| = 2 \bmod 3$.

Step 1: $|W| = (2) \bmod (3)$ → no. of states
(2 → initial, n)
→ Starting table of given no & add n in series.

i.e-

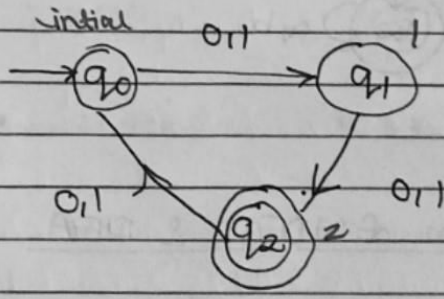
Series → 2, 5, 8, 11, ...

i.e: $5 = 3\times1+2 = 3+2 = 5$

$6 = 3\times2+2 = 8$

Step 2:

like = n=2
2 → final



Step 3: Make initial point & whatever the "n" is given which initial & start counting "n" times to get your final point or state.

Step 4: Mention 5 tuples of FA:

$\{Q, \Sigma, \delta, q_0, F\}$

acc. to. question

$= (\{q_0, q_1, q_2\}, \{0,1\}, \delta = Q\times\Sigma = Q, q_0, q_2)$

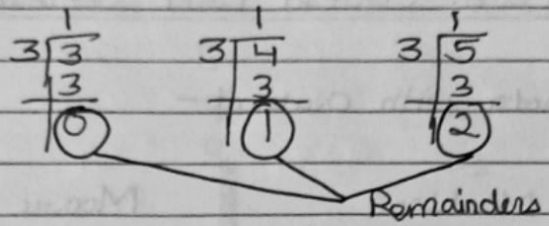
↓ set of states ↓ set of input ↓ Transition Func^n ↓ initial state ↓ Final State

Date:
Page:

Numerical - Construct a DFA which accepts a language of all binary strings divisible by 3 over $\Sigma(0,1)$.

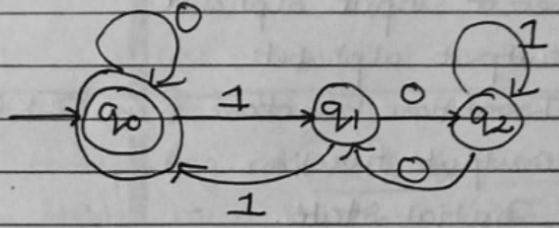
Sol-

Step1:



q_0 0
q_1 1
q_2 2

Step2:



Date:
Page:

Unit : 02

Chapter 2.1. Properties of Regular Set
Chapter 2.2. Finite Automata with Output
Chapter 2.3. Context Free Grammars

Finite Automata with Output -

	Mealy Machine	Moore Machine
1. Definition:	A Mealy Machine and Moore Machine has 06 tuple - $\{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$ Q: Finite set of states Σ: set of input alphabet Δ: output alphabet δ: Transition Function from $\Sigma \times Q$ into Q. λ: output function. q_0 : Initial state.	
2. Output function:	value of output function $z(t)$ is present state $q_1(t)$ and the present input $x(t)$. i.e. $z(t) = \lambda(q_1(t), x(t))$.	2. value of output function is depends upon present state. i.e. $z(t) = \lambda(q_1(t))$.
3.	If input changes, output also changes.	3. If input changes, output does not change.
4.	Less number of states are required.	4. More states are required.
5.	Asynchronous output generation.	5. Synchronous output & state generation.
6.	Output is placed on transition.	6. Output is placed on state.
7.	Difficult to Design.	7. Easy to Design.

Current Next State
Corresponding Previous State

Page:

Conversion Moore to Mealy Machine -

Present State	Next State a=0	Next State a=1	output (λ)
→ q_1	q_1	q_2	0
q_2	q_1	q_3	0
q_3	q_1	q_3	1

Corresponding

Sol:

Present State	a=0 Next State	a=0 Output	a=1 Next State	a=1 Output
→ q_1	q_1	0	q_2	0
q_2	q_1	0	q_3	1
q_3	q_1	0	q_3	1

Current State

→ The rows corresponding to q_2 & q_3 are identical.
So we can delete one of the two states i.e. q_2 or q_3.
We delete q_3.

Present State	a=0 Next state	a=0 Output	a=1 Next State	a=1 Output
→ q_1	q_1	0	q_2	0
q_2	q_1	0	q_2	1

Conversion Mealy to Moore Machine -

Present State	Next state a=0	Output	Next state a=1	Output
→ q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Sol:

Present State	a=0 Next state	a=1 Next State	output
→q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_{403}	q_{291}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Derivation Tree

A Derivation Tree (also called as "Parse tree") for content free grammar (CFG), $G = (V_N, \Sigma, P, S)$

where V_N: Variable

P: set of Production

Σ: Terminal

S: Starting of Production.

is a tree satisfying the condition -

1. Every vertex has a label which is a variable or terminal or Λ.
2. Root has label S.
3. Label of an internal vertex is a variable.
4. If vertices $n_1, n_2 \ldots n_k$ written with labels $X_1, X_2 \ldots X_n$ are sons of vertex n with label A, then $A \to X_1 X_2 \ldots X_n$ is a production in P.
5. A vertex n is a leaf if its label is $a \in \Sigma$ or Λ n is the only son of its father if its label is Λ.

Example:

Let $G = (\{S, A\}, \{a, b\}, P, S\}$ where P consist $S \to aAS \mid a \mid SS$, $A \to SbA \mid ba$.

String: "aabbaa". Construct Derivation Tree?

Sol: $S \to aAS$ $\because S \to aAS$

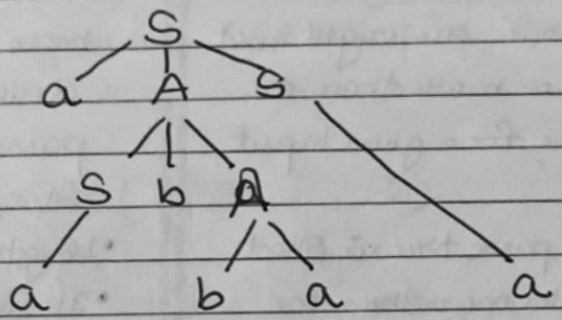
$\to aSbAS$ $\because A \to SbA$

$\to aabAS$ $\because S \to a$

$\to aabbaS$ $\because A \to ba$

$\to aabbaa$ $\because S \to a.$

Parse Tree or Derivation Tree:



Derivation Tree
(02 Types)

Left Most
A Derivation $A \overset{*}{\Rightarrow} w$ is called Left most Derivation if we apply a production only to the leftmost variable at every time.

Ex: Show that a+a*b

$S \to S+S$, $S \to S*S$, $S \to a/b$

Sol:

$S \to \overleftarrow{S*S}$ $(S \to S+S)$

$\to \overleftarrow{S+S}*S$ $(S \to a)$

$\to a+\overleftarrow{S*S}$ $(S \to a)$

$\to a+a\overleftarrow{*S}$ $(S \to b)$

$\to a+a*b$

Right Most
A Derivation $A \overset{*}{\Rightarrow} w$ is called Right Most derivation if we apply a production only to the right most variable at every time.

Ex:

$S \to \overrightarrow{S+S}$ $(S \to S*S)$

$\to S+\overrightarrow{S*S}$ $(S \to b)$

$\to \overrightarrow{S+S*b}$ $(S \to a)$

$\to S+\overrightarrow{a*b}$ $(S \to a)$

$\to a+a*b$

hence proved.

Date:
Page:

Types of Grammar

Ambiguous Grammar

- A grammar said to be Ambiguous Grammar if there exists more than one left most derivation or more than one right most derivation or more than one parse tree for a given input String.
- Length of parse tree is short
- It generates more than one parse tree.
- It contains Ambiguity.
- Amount of non terminals is less.
- Ambiguous Grammar occurs if there exists more than one derivation tree.

Example-

$S \rightarrow S+S$

$S \rightarrow S*S$

$S \rightarrow S$

$S \rightarrow a$

Unambiguous Grammar

- A grammar is said to be Unambiguous if it does not contain more than one left most derivation or more than one right most derivation or more than one parse tree for a given input String.
- Length of parse tree large
- It generates only one parse tree.
- It does not contain anything
- Amount of non terminals is more.
- Unambiguous Grammar occurs if there exists only one & only one derivation tree

Example-

$X \rightarrow AB$

$A \rightarrow Aa/a$

$B \rightarrow b$

Date:
Page:

Chomsky Hierarchy -

Unrestricted Grammar ⟶ Type 0
Content Sensitive Grammar ⟶ Type 1
content Free Grammar ⟶ Type 2
Regular Grammar ⟶ Type 3

Context free Grammar (CFG) -

Content Free grammar (CFG) is a set of recursive rewriting rules or productions used to generate pattern of strings.

$$G_1 = (V, T, P, S)$$

where,

V : Final set of a non-terminal symbol. (like A, B, C etc).
T : Final set of a Terminal symbol. (like a, b, 1, 2 etc)
P : Set of Production rules
S : Start symbol.

Date:
Page:

Minimisation of Deterministic Finite Automata

Remember-

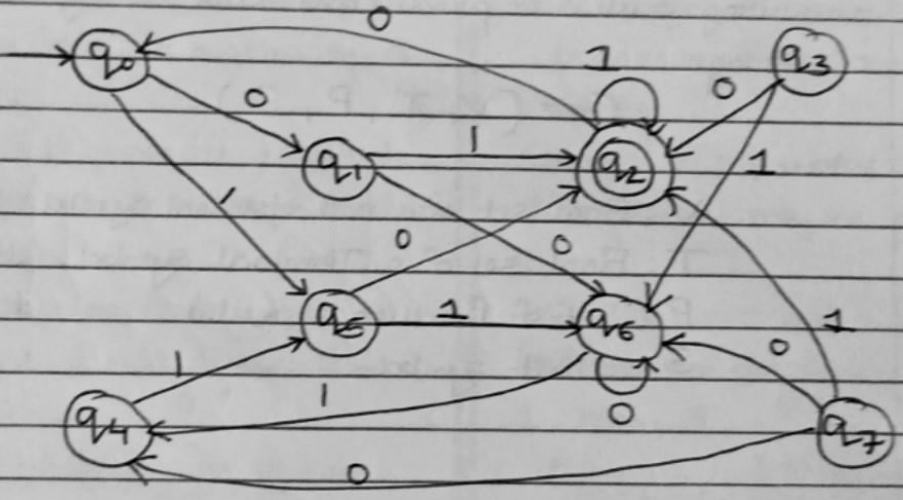
Q_1^0 : final State

Q_2^0 :

$$\boxed{Q_2^0 = Q - Q_1^0}$$

Q_1^1

Ques-1) Construct a minimum state Automation equivalent to the finite Automation?



Solution:

Step 1: Draw Simple Transition Table

State	0	1
→ q_0 initial	q_1	q_5
q_1	q_6	q_2
(q_2) final	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

Step 2: Construction of π_0.

By definition of 0-Equivalence $\pi_0 = \{Q_1^0, Q_2^0\}$ where Q_1^0 is final states and $Q_2^0 = Q - Q_1^0$

$\therefore Q_1^0 = F = \{q_2\}$

$Q_2^0 = Q - Q_1^0$

So,

$\pi_0 = \{\{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}\}$

Step 3: Construction of π_{k+1} from π_k

$q_2 = q_3$ & q_5 0 output.
$q_2 = q_1$ & q_7 1 output.

$\pi_1 = \{\{q_2\}, \{q_3, q_5\}, \{q_1, q_7\}, \{q_0, q_4, q_6\}\}$

$\pi_2 = \{\{q_2\}, \{q_3, q_5\}, \{q_1, q_7\}, \{q_0, q_4\}, \{q_6\}\}$

$\pi_3 = \pi_2$ ∴ further uplift not possible.

Step 4: Construction of minimum automation.

Transition Table

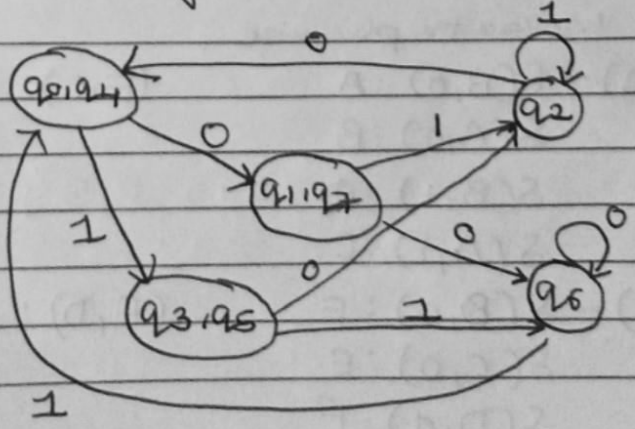
State	0	1
→$\{q_0, q_4\}$	$\{q_1, q_7\}$	$\{q_3, q_5\}$
$\{q_1, q_7\}$	$\{q_6\}$	$\{q_2\}$
$\{q_2\}$	$\{q_0, q_4\}$	$\{q_2\}$
$\{q_3, q_5\}$	$\{q_2\}$	$\{q_6\}$
$\{q_6\}$	$\{q_6\}$	$\{q_0, q_4\}$

Rough work

	0	1
q_0	q_1	q_5
q_4	q_2	q_5
q_6	q_6	q_1

same output input in one set

Transition Diagram



Minimum State Automation.

Date:
Page:

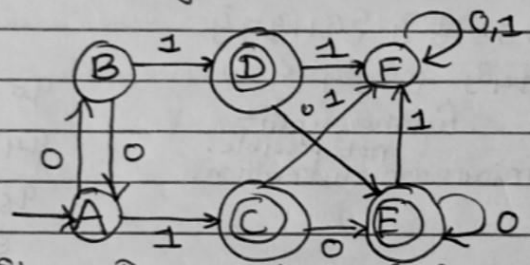
Myhill – Nerode Theorem

Steps-

1. Draw a table for all pair of states (P, Q)
2. Mark all pair where P∈F & Q∉F
3. If there are any unmarked pair (P, Q) such that {δ(P, x), δ(Q, x)} is marked then mark [P, Q] where 'x' is an input symbol.
4. Combine all the unmarked pair & make them a single state in the minimized DFA.

★ one is Final State another is non-final State you have to mark

Qus-1.



Sol: Step1: Draw a table 6x6

	A	B	C	D	E	F
A						
B						
C	✓	✓				
D	✓	✓				
E	✓	✓				
F	✓	✓	✓	✓	✓	

Step2: Mark

Step3: Marked the pair wise.

(B,A) – δ(B,0) : A
δ(A,0) : B
δ(B,1) : D
δ(A,1) : C

(D,C) – δ(D,0) : E
δ(C,0) : E
δ(D,1) : F
δ(C,1) : F

(E,C) : δ(E,0) : E
δ(C,0) : E
δ(E,1) : F
δ(C,1) : F

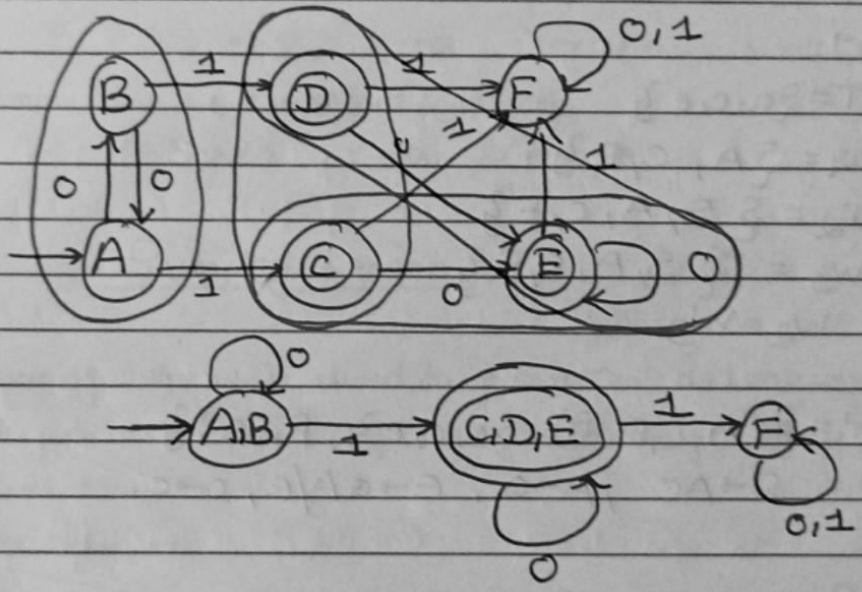
(E,D) : δ(E,0) : E
δ(D,0) : E
δ(E,1) : F
δ(D,1) : F

(F,A) : $\delta(F,0) : F$
$\delta(A,0) : B$
$\delta(F,1) : F$
$\delta(A,1) : C$
↓
{F, C} : marked
then {F, A} : also marked

(F,B) : $\delta(F,0) : F$
$\delta(B,0) : A$
$\delta(F,1) :$
$\delta(B,1) :$
↓
{F, A} : marked hai
then {F, B} : also marked

Step4: combine all the Unmarked pair state in a single state.

{ (A,B) (D,C) (E,C) (E,D) }



Minimised DFA.

Date:
Page:

Removal or Reduction of Grammar

Ques- 1) Find a reduced grammar equivalent to the grammar to the grammar G, having production rules - P: S→AC/B,
A→a ,
C→c|BC,
E→aA/e?

Sol- Grammar = {Vn, T, P, S}

Vn: Non-terminal symbol (A, B, C...)
T: Terminal Symbol (a, b, 1, 2, ...)
P: Production rule
S: Start.

Phase 1:

T = {a, c, e}

W_1 = {A, C, E}

W_2 = {S, A, C, E}

W_3 = {S, A, C, E}

∴ $W_2 = W_3$

G' = {(A, C, E, S), {a, c, e}, P, (S)}

P = S→AC , A→a , E→aA/e, C→c.

Phase 2:

Y_1 = {S}

Y_2 = {S, A, C}

Y_3 = {S, A, C, a, c}

Y_4 = {S, A, C, a, c}

G'' = {(A, C, S), (a, c), P, S}

P = S→AC , A→a , C→c

Simply, we are doing Backtracking in the unit Productions.

Removal of Unit Productions

Q.1) Remove Unit Productions from the Grammar whose production rule is given by P: $S \to XY$,
$X \to a$,
$Y \to Z/b$,
$Z \to M$, $M \to N$, $N \to a$?

Sol: Since $N \to a$, we add $M \to a$
P: $S \to XY$, $X \to a$, $Y \to Z/b$, $Z \to M$, $M \to a$, $N \to a$
Since $M \to a$, we add new production $Z \to a$
P: $S \to XY$, $X \to a$, $Y \to Z/b$, $Z \to a$, $M \to a$, $N \to a$
Since $Z \to a$, we add new production $Y \to a$
P: $S \to XY$, $X \to a$, $Y \to a/b$, $Z \to a$, $M \to a$, $N \to a$.
Remove the unreachable symbols-
P: $S \to XY$, $X \to a$, $Y \to a/b$

Pumping Lemma

* Pumping lemma is used to prove that a language is not regular.
* It gives a method for pumping (generating) many sub strings from a given string.
* Theorem-
For any regular language L, there exists an integer P, such that $\forall$ w in L, $|W| \geq P$.
We can break w into three strings $w = xyz$ i.e.
  i. $|xy| \leq P$
  ii. $|y| > 0$
  iii. $xy^i z \in A,L \; \forall \; i \geq 0$.
  ↳ i: Sequence of Symbols.

SA,L: Regular Langua

* Application of Pumping Lemma-
  - It is to be applied to show that certain languages are not regu
  - It should never be used to show a language is regular.

Another Definition of Pumping Lemma Theorem -

If A is a Regular Language, then A has a Pumping Length 'P', such that any string 'S' where $|S| \geq P$ may be divided into 3 parts $S = xyz$ such that the following conditions must be true -

1. $xy^iz \in A$ for every $i \geq 0$
2. $|y| > 0$
3. $|xy| \leq P$

To prove that a language is not Regular using Pumping Lemma - (we prove using Contradiction) -

1. Assume that A is Regular.
2. It has to have a pumping length (say P).
3. All strings longer than P can be pumped $|S| \geq P$.
4. Now find a string 'S' in A i.e. $|S| \geq P$.
5. Divide S into xyz.
6. Show that $xy^iz \notin A$ for some i.
7. Then consider all way that S can be divided into xyz.
8. Show that none of these can satisfy all the 3 pumping conditions at the same time.
9. S cannot be pumped == "CONDRADICTION".

Que-1) Using Pumping Lemma prove that language - $A = \{a^n b^n \mid n \geq 0\}$ is not regular?

Sol- Assume that A is Regular

Pumping length (P) = 7

String (S) = $a^P b^P$

Divide S into XYZ

S = aaaaaaabbbbbbb

Case I: Y is in the "a" part.

aa aaaaa bbbbbbb
x y z

case II: Y is in the "b" part

aaa aaa a bbb bbbb b
x y z

case III: Y is in the "a" and "b" part

aaaaaaa a bb bbbbbb
x y z

consider p = 02

for case I: ~~xy~~ $xy^1z = xy^2z$

aa aaaaa a aaaabbbbbbb

a:11 ≠ b:7

case II: $xy^1z = xy^2z$

aaaaaaa abbbbbbbbbbb

a:7 ≠ b:11

case III: $xy^1z = xy^2z$

aaaaa aabbaabb bbbbb

$|xy| \leq p \quad \therefore p = 7.$

for case I : 6 ≤ 7

case II : 13 ≤ 7

case III : 9 ≤ 7

Date:
Page:

Removal of Null Production

Que-1) Remove Null Production from the following Grammar-

$S \rightarrow ABAC$

$A \rightarrow aA/\epsilon$

$B \rightarrow bB/\epsilon$

$C \rightarrow c$?

In a CFG, a Non Terminal Symbol 'A' is a nullable variable if there is a production $A \rightarrow \epsilon$ or there is a derivation starts at "A" and leads to ϵ.

Sol- Steps-

1. To remove $A \rightarrow \epsilon$, look for all productions whose right side contains the A.
2. Replace each occurrence of A in each of these production with ϵ.
3. Add the resultant productions to the Grammar.

$A \rightarrow \epsilon$, $B \rightarrow \epsilon$.

Firstly: To eliminate $A \rightarrow \epsilon$

$S \rightarrow ABAC/ABC/BAC/BC$.

$A \rightarrow aA/a$

$B \rightarrow bB/\epsilon$

$C \rightarrow C$

Secondly: To eliminate $B \rightarrow \epsilon$

$S \rightarrow ABAC/ABC/BAC/BC/AAC/AC/C$

$A \rightarrow aA/a$

$B \rightarrow bB/b$

$C \rightarrow c$.

∴

We have removed all the nooll production so none of the non terminal symbols are giving Epsilon "ϵ"

∴

It simplified the CFG.

Chomsky Normal Form (CNF)-

A context Free grammar G_1 is in Chomsky normal form if every production is of the form $A \to a$ or

$A \to BC$ &

$S \to \Lambda$ is in G_1 if $\Lambda \in L(G_1)$. When Λ is in $L(G_1)$, we assume that S does not appear on the RHS of any production.

Example- Consider G_1 whose productions are

$S \to AB / \Lambda$,

$\Lambda \to a$,

$B \to b$, Then G_1 is in Chomsky Normal.

Steps to convert a given CFG to Chomsky Normal Form -

Step1. If the start Symbol S occurs on some right side, create a new start symbol 'S' & new Production $S' \to S$.

Step2. Remove Null Production.

Step3. Remove Unit Production.

Step4. Replace each production $A \to B_1 \dots B_n$ where $n > 2$ with $A \to B_1 C$ where $C \to B_2 \dots B_n$.

Repeat this step for all productions having two or more symbols RHS

Step5. If the right side of any Production is in the form $A \to aB$ where

a : is terminal

A, B : are non terminals.

Then the production is replaced by the $A \to XB$ and $X \to a$.

Repeat this step for every production which is in the form of $A \to aB$.

Que1) Convert the following CFG to CNF:

P : $S \to ASA / aB$

$A \to B / S$

$B \to b / \in$

Sol: I. Since S appears in RHS, we add a new state "S'"

P: S' → S, S → ASA/aB, A → B/S, B → b/ε.

II. Remove the null production:

B → ε & A → ε

After removing B → ε

P: S' → S, S → ASA/aB/a, A → B/S/ε, B → b

removing A → ε

P: S' → S, S → ASA/aB/a/AS/SA/S,

A → B/S, B → b.

III. Remove unit production:

S → S, S' → S, A → B and A → S.

After removing S → S

P: S' → S, S → ASA/aB/~~AS~~/a/AS/SA, A → B/S,

B → b.

removing S' → S

P: S' → ASA/aB/a/AS/SA,

S → ASA/aB/a/AS/SA, A → B/S, B → b

removing A → B

P: S' → ASA/aB/a/AS/SA

S → ASA/aB/a/AS/SA, A → b/S, B → b

removing A → S

P: S' → ASA/aB/a/AS/SA

S → ASA/aB/a/AS/SA

A → b/ASA/aB/a/AS/SA

B → b.

IV. Now find out the productions that has more than Two variables in RHS,

S' → ASA, S → ASA & A → ASA

(ASA = X)

After removing those

P: S' → AX/aB/a/AS/SA

S → AX/aB/a/AS/SA

A → b/Ax/aB/a/AS/SA

B → b, X → SA.

V. Now change the productions $S' \to aB$, $S \to aB$ and $A \to aB$

P: $S' \to AX/YB/a/AB/SA$,
$S \to AX/YB/a/AS/SA$,
$A \to b/AX/YB/a/AS/SA$,
$B \to b$,
$X \to SA$,
$Y \to a$.

Greibach Normal Form (GNF)-

A context Free grammar is in the Greibach Normal Form if every production is of the form $A \to a\alpha$ $\{\alpha \in V_N\}$ & $a \in \Sigma$
$S \to \Lambda$ is in the Gr if $\Lambda \in L(G_1)$. When $\Lambda \in L(G_1)$ we assume that S does not appear on the RHS of any production.

Example- Gr given by $S \to aAB/\Lambda$
$A \to bC$
$B \to b$
$C \to c$ is in GNF.

Depends Upon two Lemmas are-

- **Lemma 1:**
Let $G_1 = (V_N, \Sigma, P, S)$ be CFGr. Let $A \to BY$ be an Λ production in P. Let B-production be $B \to \beta_1/\beta_2/\cdots/\beta_n$.
$\therefore G_2 = (V_N, \Sigma, P, S)$ is an context free grammar equivalent to G_1.

- **Lemma 2:**
Let $G_1 = (V_N, \Sigma, P, S)$ be CFG. Let set Λ production be
$A \to A\alpha_1/A\alpha_2 \cdots A\alpha_r/\beta_1/\cdots/\beta_s$.
Let Z be a new variable.
New production
(I) Λ productions in P_1 are $A \to \beta_1/\cdots/\beta_s$, $A \to \beta_1 Z/\beta_2 Z/\cdots/$
(II) Z productions in P_1 are $Z \to \alpha_1/\cdots/\alpha_r$, $Z \to \alpha_1 Z/\alpha_2 Z/\cdots/$
(III) Production for other variable are as in P.
Then CFG1 is equivalent to G_1.

In General Way

$A \to b$

$A \to bC_1C_2 \dots C_R$

where $A, C_1, C_2 \dots$ are non terminals

b - Terminals

Steps to convert a given CFG to GNF-

Step1. Check if the given CFG has any Unit production or Null prod & remove if there are any.

Step2. Check whether the CFG is already in CNF and convert it to CNF if it is not.

Step3. Changes the names of the Non Terminal Symbols into some A_i in ascending order of i.

Ques 1) Convert grammar $S \to CA/BB$

$B \to b/SB$

$C \to b$

$A \to a$ into GNF?

Sol- Step 1 & Step 2 is applicable.

Step 3: Replace: $S \to A_1$, $C \to A_2$, $A \to A_3$, $B \to A_4$ } randomly assign numbers but in ascending order

So, new production

$A_1 \to A_2A_3 / A_4A_4$

$A_4 \to b / A_1A_4$

$A_2 \to b$

$A_3 \to a$

Step4. Alter the rules so that the non-terminals are in ascending order, such that - If production is in the form

$A_i \to A_jX$

then $i < j$ & should never be $i \geq j$

$A_4 \to b / A_1A_4$

$A_4 \to b / A_2A_3A_4 / A_4A_4A_4$

$A_4 \to b / bA_3A_4 / A_4A_4A_4$

$A_4 \to b / bA_3A_4 / A_4A_4A_4$ → Left Recursion.

Step5. Remove Left Recursion.

$A_4 \rightarrow b \mid bA_3A_4 \mid A_4A_4A_4$

(Left Recusion)

$Z \rightarrow A_4A_4Z \mid A_4A_4$ where Z: new variable

$A_4 \rightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z \mid$

now,

$A_1 \rightarrow A_2A_3 \mid A_4A_4$

$A_4 \rightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

$Z \rightarrow A_4A_4 \mid A_4A_4Z$

$A_2 \rightarrow b$

$A_3 \rightarrow a$.

} again checking $A_i \rightarrow A_jx$ where $i \leq j$

$A_1 \rightarrow bA_3 \mid b \mid bA_3A_4 \mid bZ \mid bA_3A_4ZA_4$.

(b from A_4, bA_3A_4 from A_4, bZ from A_4)

$A_4 \rightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

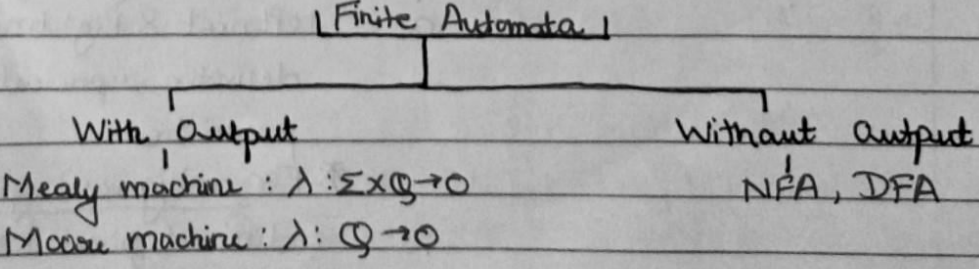
$Z \rightarrow bA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_3A_4ZA_4 \mid$
$bA_4Z \mid bA_3A_4A_4Z \mid bZA_4Z \mid bA_3A_4ZA_4Z$.

$A_2 \rightarrow b$

$A_3 \rightarrow a$

Finite Automata -

- It have with output machines do not have State.
- It is simplest machine to recognise patterns.
- It has 5 tuples - $\{Q, \Sigma, q, F, \delta\}$.



Right & Left
Date: Leftmost
Page: Same parse tree

Unambiguous

Q-1) Consider the production $S \to aB/bA$
$A \to aS/bAA/a$
$B \to bS/aBB/b$, obtain for the String aaabbabbba. Find.

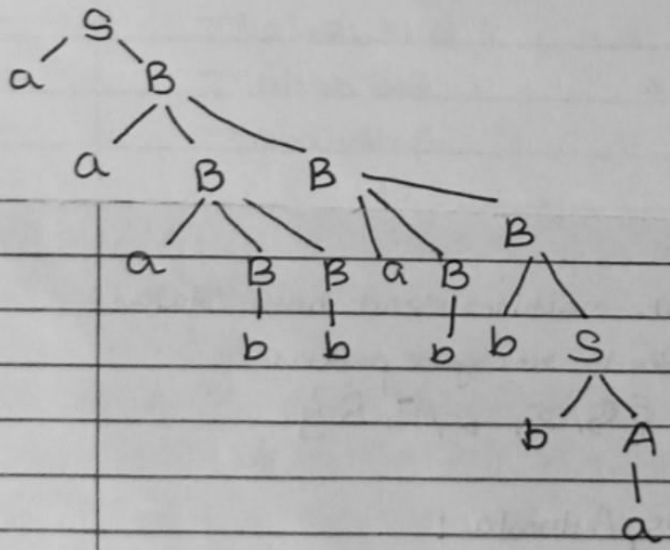
(I) Leftmost derivation:

$S \Rightarrow aB$
$\to aaBB$ ($\because B \to aBB$)
$\to aaaBBB$ ($\because B \to aBB$)
$\to aaabBB$ ($\because B \to b$)
$\to aaabbB$ ($\because B \to b$)
$\to aaabbaBB$ ($\because B \to aBB$)
$\to aaabbabB$ ($\because B \to b$)
$\to aaabbabbS$ ($\because B \to bS$)
$\to aaabbabbbA$ ($\because S \to bA$)
$\to aaabbabbba$ ($\because A \to a$)

(II) Rightmost derivation:

$S \to aB$
$\to aaBB$ ($\because B \to aBB$)
$\to aaBaBB$ ($\because B \to aBB$)
$\to aaBaBbS$ ($\because B \to bS$)
$\to aaBaBbbA$ ($\because S \to bA$)
$\to aaBaBbba$ ($\because A \to a$)
$\to aaBabbba$ ($\because B \to b$)
$\to aaaBBabbba$ ($\because B \to aBB$)
$\to aaaBbabbba$ ($\because B \to b$)
$\to aaabbabbba$ ($\because B \to b$)

(III) Parse Tree



Notes →

* For unambiguous grammar, Leftmost & Rightmost derivation represent Same parse tree.
* For ambiguous grammar, different parse tree.

Q.2) Show that id * id can be generated by two distinct leftmost derivation in the grammar. $E \to E+E / E*E / (E) / id$.

Sol- (1) $E \to E+E$
$\to id+E$
$\to id+E*E$
$\to id+id*id$

$E \to E*E$
$\to E+E*E$
$\to E+E*id$
$\to E+id*id$
$\to id+id*id$

$\therefore$ id + id * id can be generated by two distinct LMD.

Q.3) Construct a CFG for $L = \{a^n b^n, n \geq 1\}$

Sol-

n	$a^n b^n$	string
1	$a^1 b^1$	ab
2	$a^2 b^2$	aabb
3	$a^3 b^3$	aaabbb

So, $L = \{ab, aabb, aaabbb, \ldots\}$

Since least string is ab

$S \to ab$

If a is generated, then its corresponding b has to generated

$S \to aSb$

$\therefore G = \{V, \Sigma, P, S\}$
$= \{(S, a, b), (a, b), P, S\}$
$P = \{ S \to aSb / ab \}$.

Q.4) The set of Regular Language are closed under -

(a) Union: $L_1 = \{a, b\}$ $L_2 = \{c, d\}$: $L_1 \cup L_2 = \{a, b, c, d\}$
(b) Concatenation: : $L_1 L_2 = \{ac, ad, bc, bd\}$.
(c) Kleenestar: $L^* = \{x_1 \ldots x_n / n \geq 0\}$: $L^* = \{\epsilon, a, b, aa, ab, \ldots\}$.
(d) Kleene plus: $L^+ = \{x_1 \ldots x_n / n \geq 0\}$: $L^+ = \{a, b, aa, ab, \ldots\}$
(e) Intersection: $L_1 = \{a, b\}$ $L_2 = \{c, a\}$: $L_1 \cap L_2 = \{a\}$
(f) Diffrence: $L_1 = \{a, b, c\}$ $L_2 = \{b, c\}$: $L_1 - L_2 = \{a\}$
(g) Complement: $\Sigma^* - L = \{x / x \in \Sigma^* \forall x \notin L\}$: $\Sigma^* L = \{\epsilon, a, b, ab, \ldots\}$.

Q.5)

CNF	GNF
I. Starting Symbol generate ϵ. Ex: $A \to \epsilon$	I. Starting Symbol generate ϵ. Ex: $A \to \epsilon$
II. Non terminal generate 2 non terminal. $S \to AB$	II. Non terminal generate terminal. $A \to a$
III. Non terminal generate terminal. $S \to a$	III. Non terminal generate terminal followed by any non terminal. $S \to aASB$.

Q 6) Using pumping Lemma, show that the fallowing sets are not regular.

(a) $L=\{a^n b^{2n} / n>0\}$: L is not regular by contradiction

If L is regular we apply pumping lemma

$|xy| \leq n$, $|y|>0$ & $xz \in L$.

$|xy| \leq n$, $xy = a^m$ & $y = a^l$ where $0<l \leq n$

So $xz = a^{n-l} b^{2n} \in L$, contradiction $n-l \neq n$

Thus L is not regular.

(b) $L = \{a^n b^m / 0<n<m\}$: n be no. of states

$w = a^n b^m$, $m>n$

By applying pumping lemma

So, $a^{n-l} a^{lk} b^m \in L$ $\forall k>0$

$n-l+lk > m$, This is contradiction

Hence L is not regular.

Unit : 03

Chapter 3.1: Context Sensitive Language

Chapter 3.2: Rushdawn Automata

Chapter 3.3: Turing Machines

Rushdawn Automata (PDA)-

- Rushdawn Automata is a finite Automata with extra memory called Stack which helps Rushdawn automata to recognise Context Free language.
- It has 07 tuple are-

$$PDA - \{Q, \Sigma, \Gamma, q_0, Z_0, F, \delta\}.$$

where:

Q: set of states

Σ: set of input states.

Γ: Rushdawn symbol.

q_0: Initial State.

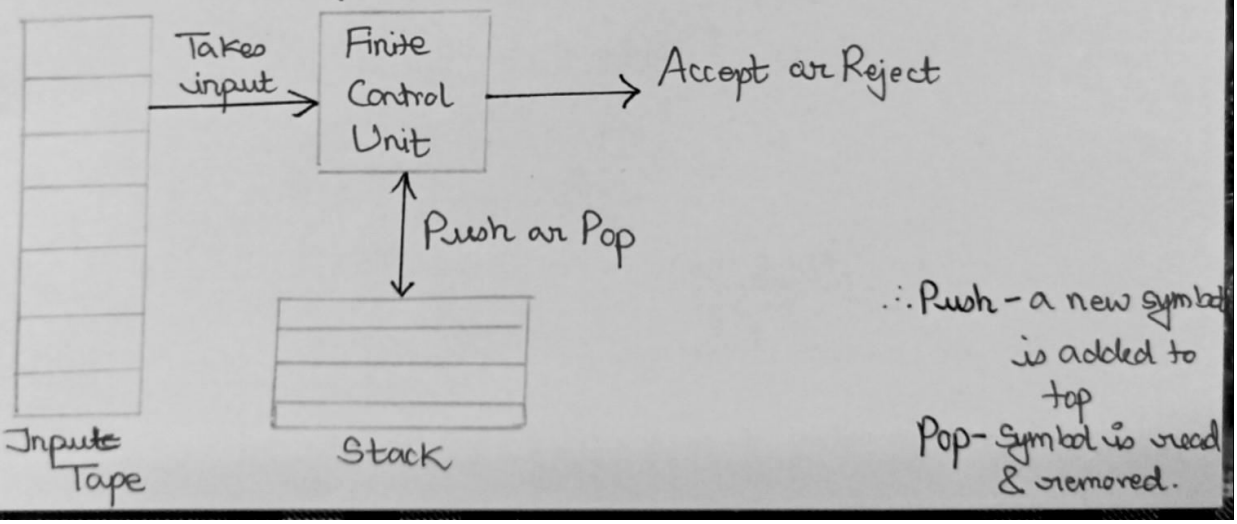
Z_0: Initial Symbol.

F: Set of Final States.

δ: Transition Function

$$\delta = Q \times \{\Sigma \cup \{\Lambda\}\} \times \Gamma$$

- Basically a pushdown Automation is –
  "Finite State Machine" + "a Stack".
- A Pushdown Automation has 03 components –
  An Input Tape / A control Unit / A stack with infinite size.
- A stack head scans the top symbol of the Symbol.
- A PDA may or may not read an input Symbol but it has to read the top of the stack in every Transition.



∴ Push – a new symbol is added to top
Pop – Symbol is read & removed.

Rules –

Rule 1: $\delta(q_0, a, z_0) = \{(q_0, az_0)\}$
similarly $\delta(q_0, b, z_0) = \{(q_0, bz_0)\}$.

Rule 2: $\delta(q_0, a, a) = \{(q_0, aa)\}$
similarly $\delta(q_0, b, a) = \{(q_0, ba)\}$.

Rule 3: $\delta(q_0, a, b) = \{(q_0, ab)\}$
similarly $\delta(q_0, b, b) = \{(q_0, bb)\}$

Rule 4: $\delta(q_0, c, a) = \{(q_1, a)\}$
similarly $\delta(q_0, c, b) = \{(q_1, b)\}$

Rule 5: $\delta(q_1, a, a) = \delta(q_1, b, b) = \{(q_1, \Lambda)\}$.

Rule 6: $\delta(q_1, \Lambda, z_0) = \{(q_f, z_0)\}$.

Q-1) $A = (\{q_0, q_1, q_f\}, \{a, b, c\}, \{a, b, z_0\}, \delta, q_0, z_0, \{q_f\})$ is a pda with string is $(q_0, bacab, z_0)$?

Sol – using Rules –

$\delta(q_0, bacab, z_0)$
$\delta(q_0, acab, bz_0)$
$\delta(q_0, cab, abz_0)$
$\delta(q_1, ab, abz_0)$
$\delta(q_1, b, bz_0)$
$\delta(q_1, \Lambda, z_0)$
$\therefore \delta(q_1, \Lambda, z_0) = \{(q_f, z_0)\}$.

Q-2) Construct a pda A equivalent to following content Free Grammar : $S \to 0BB$

$B \to 0S \mid 1S \mid 0$ and,

Test whether 0^4 is in $N(A)$?

Sol – Define A:

$A = (\{q\}, \{0, 1\}, \{S, B, 0, 1\}, \delta, q, S, \Gamma)$

δ is defined by :

R_1 : $\delta(q, \Lambda, S) = \{(q_0, 0BB)\}$

R_2 : $\delta(q, \Lambda, B) = \{(q_0, 0S), (q_0, 0), (q_0, 1S)$

R_3: $\delta(q,0,0) = \{(q,\Lambda)\}$

R_4: $\delta(q,1,1) = \{(q,\Lambda)\}$

Then

$(q, 010^4, S)$

$\rightarrow (q, 010^4, 0BB)$

$\rightarrow (q, 10^4, BB)$

$\rightarrow (q, 10^4, 1SB)$

$\rightarrow (q, 0^4, SB)$

$\rightarrow (q, 0^4, 0BBB)$

$\rightarrow (q, 0^3, BBB)$

$\rightarrow (q, 0^3, 000)$

$\rightarrow (q, \Lambda, \Lambda)$

$\therefore$

$010^4 \subseteq N(A)$

Applications of Pushdown Automata –

There are many applications of PDA are:

1. Used in the Syntax Analysis phase.
2. Implementations of stack applications.
3. Used in evaluations of the arithmetic Expressions.
4. Used for solving the Tower of Hanoi Problem.
5. Used in XML parsing.
6. Used in automatic theorem proving & formal verification.
7. Used in formal verification of hardware & software system.
8. Building compilers & interpreters.
9. Parsing context free grammars.
10. Modeling network problems.

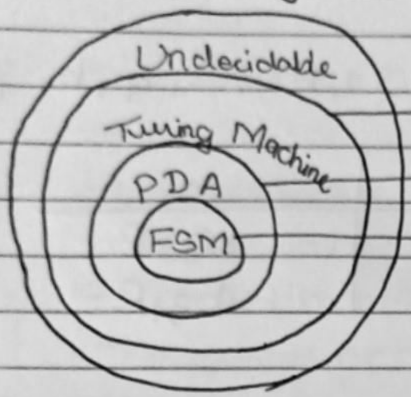
Note → The Pushdown Automata (PDA) has 3 components –

i. An input tape.

ii. A Finite Control unit.

iii. A stack with infinite size.

Turing Machine



→ Recursively Enumerable Languages
→ Context Free languages
→ Regular Languages.

- Turing Machine was invented by Alan Turing in 1936 and it is used to accept Recursive Enumerable Languages.
- A Turing Machine can be defined as a set of 7 tuples –

$$T.M = \{Q, \Sigma, \Gamma, \delta, q_0, b, F\}$$

where:

Q: Finite set of states
Σ: set of input states
Γ: Tape alphaset / Symbol.
δ: Transaction Function: $\delta = Q \times \Gamma \times \{L, R\}$
q_0: Initial state
b: Blank symbol.
F: Final states.

- Applications –
  1. Used to solve the recursively enumerable problem.
  2. Used for knowing complexity Theory.
  3. Used for neural networks implementation.
  4. Used in Robotics Applications.
  5. Used in the Implementation of artificial intelligence.
  6. For understanding complexity Theory.
  7. Used in digital circuit.
  8. Used in computational biology.

Q 1) Design a Turing machine to recognise all strings consisting of an even no. of 1s.

Sol- $M = (\{q_1, q_2\}, \{1, b\}, \{1, b\}, \delta, q_1, b, \{q_1\})$

where δ is defined -

Present State	1
→ q_1	bq_2R
q_2	bq_1R

$q_1 11 \rightarrow bq_2 1 \rightarrow bbq_1$ as q_1 is an accepting state

$\therefore$

$q_1 111 \rightarrow bq_2 11 \rightarrow bbq_1 1 \rightarrow bbbq_2$. as M halts and q_2 is not an accepting state

$\therefore$ 111 is not accepted by M.

Content Sensitive Grammar -

CSG has 4 tuples -

$G = \{N, \Sigma, P, S\}$ where

N: set of non-terminal symbols.

Σ: set of terminal symbols

P: set of production.

S: Start symbol.

Content Sensitive Language -

The language that can be defined by content sensitive grammar is called CSL. Properties of CSL are -

- Union, intersection & concatenation of two CSL is content sen
- Complement of CSL is content sensitive.

Example -

Consider CSG,

$S \rightarrow abc / aAbc$

$Ab \rightarrow bA$

$Ac \rightarrow Bbcc$

$bB \rightarrow Bb$

$aB \rightarrow aa / aaA$

Sol- $S \to aAbc$

$\to abAc$

$\to abBbcc$

$\to aBbbcc$

$\to aaAbbcc$

$\to aabAbcc$

$\to aabbAcc$

$\to aabbBbccc$

$\to aabBbbccc$

$\to aaBbbbccc$

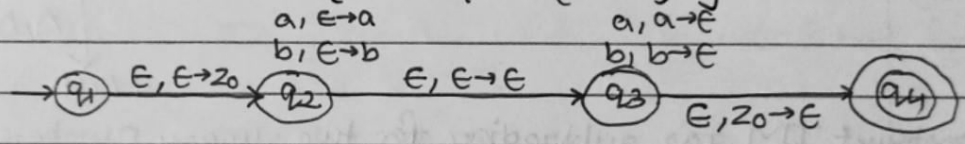
$\to aaabbbccc$

∴ The language generated by this grammar is $\{a^n b^n c^n \mid n \geq 1\}$.

Q.1) Construct PDA that accepts even palindromes of the form:

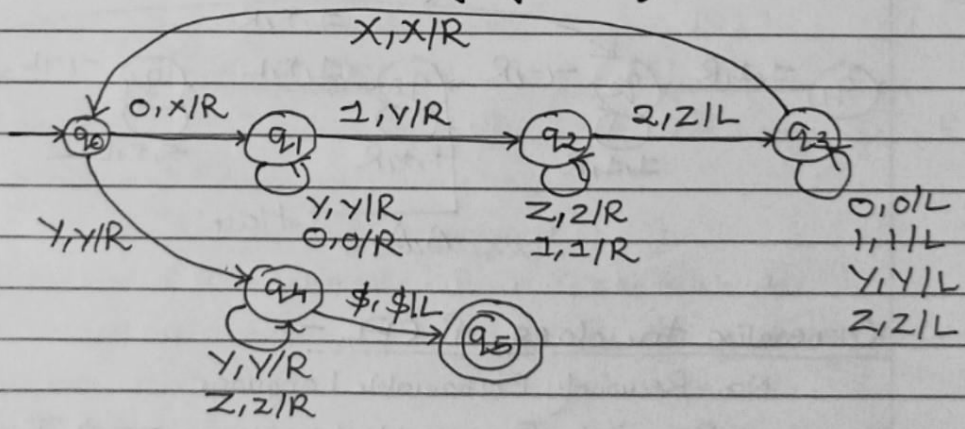
$L = \{ww^R \mid w = (a+b)^+\}$.

Sol-



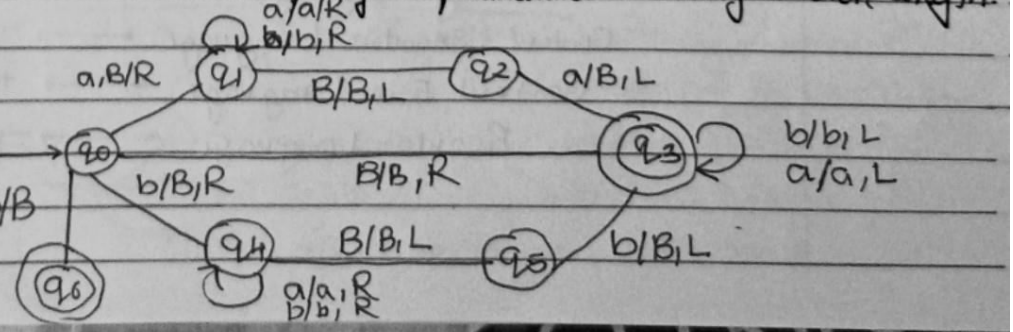
Q.2) Construct a TM for a language $L = \{0^n 1^n 2^n\}, n \geq 1$.

Sol-



Q.3) Construct a TM for checking the palindrome of string of even length?

Sol-

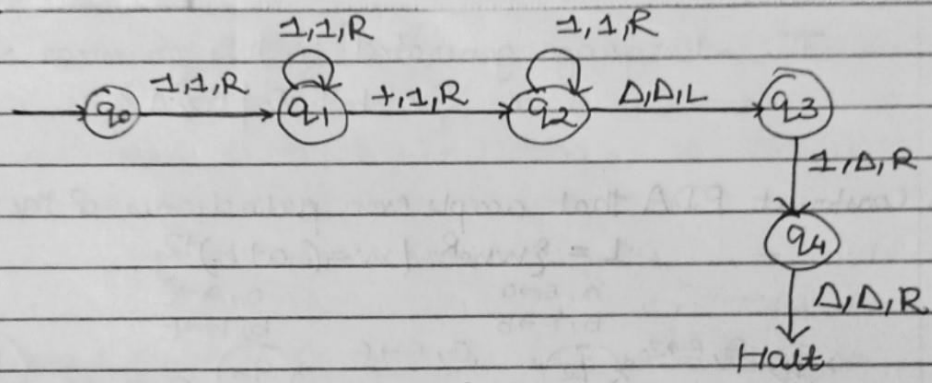


Q.4) Construct a TM for checking the palindrome of string of odd Length.

Sol-

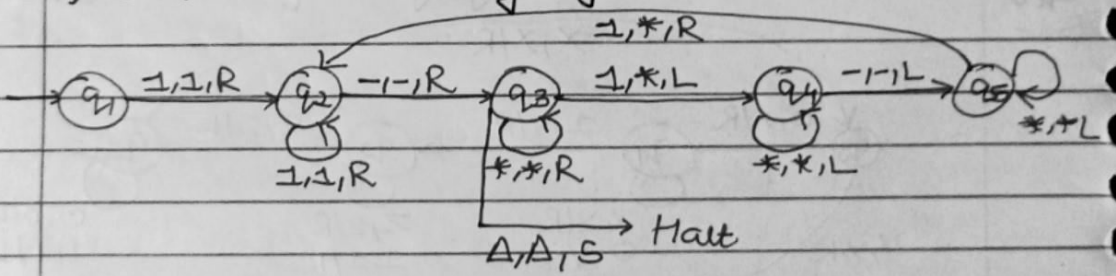
Q.5) Construct TM for addition function for the unary number.

Sol-



Q.6) Construct TM for subtraction for two unary numbers $f(a-b)=c$ where a is always greater than b.

Sol-



Generalise the class of CFL -

Non-Recursively Enumerable Language

Recursively Enumerable Language ⟶ Type 0

Recursively Languages

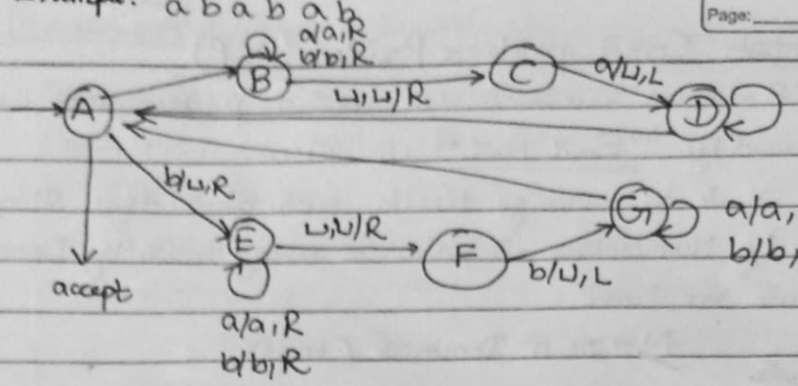
Context Sensitive Language ⟶ Type 1

Context Free Language ⟶ Type 2

Regular Language ⟶ Type 3

Q.7) Construct a TM that accepts Even Palindrome over the alphabet $\Sigma = \{a, b\}$.

Ans- Example: a b a b a b



Recursive Language

- A language "L" is said to be Recursive Language "if exists a turing machine which accept all the strings in "L" & reject all the string not in "L"."

Recursively Enumerable Language

- A Language "L" is said to be recursively Enumerable Language "if there exists a Turing machine which will accept & halt for the input strings which are in "L"."

Decidable Language

- A language "L" is decidable if it is a recursive Language. All decidable Languages are recursive languages.

Partially Decidable Language

- A language "L" is said partially Decidable Language if L is a recursively Enumerable Language.

Undecidable Language

- A language is undecidable if it is not decidable.
- An Undecidable Language may sometimes be partially decidable but not decidable.

Universal Turing Machine

The Language $A_{TM} = \{ \langle M, w \rangle / M$ is a Turing machine & M accepts $w \}$. is Turing Recognisable.

- M accept w : Alg. will Halt & Accept
- M Reject w : Alg. will Halt & Reject
- M Loops on w : Algo. will not Halt.

Post Correspondence Problem (PCP)

- The PCP is an undecidable decision problem that was introduced by "Emil Post" in 1946.
- **Aim:** It is to arrange tiles in such order that String made by Numerators is same as String made by Denominator.

In this we have

N → no. of Dominos (tiles).

Example → I st method

Dominos:

$\frac{B}{CA}$	$\frac{A}{AB}$	$\frac{CA}{A}$	$\frac{ABC}{C}$	Numerators / Denominators

We need to find a sequence of domains such that the top & bottom Strings are same?

Dominos:

A B CA A ABC → ABCAAABC

AB CA A AB C → ABCAAABC

II nd method

Another way of representing the PCP →

	A	B	
1	1	111	$\frac{1}{111}$
2	10111	10	$\frac{10111}{10}$
3	10	0	$\frac{10}{0}$

Dominos: (2) (1) (1) (3)

A 10111110 } Top and
B 10111110 } down are same

Q:1) Does PCP with two lists $x=(b, bab^3, ba)$ & $y=(b^3, ba, a)$

Sol: (2) (1) (1) (3)

$x_2\ x_1\ x_1\ x_3 = y_2\ y_1\ y_1\ y_3$

$bab^3\ b\ b\ ba = ba\ b^3\ b^3\ a$

$bab^3bbba = bab^3b^3a$

	1	2	3
x	b	bab^3	ba
y	b^3	ba	a

then

Yes, this PCP has a solution.

Q.1) Let $G_1 = (\{S, A\}, \{0,1,2\}, P, S)$, where P consists of $S \to OSA_12$, $S \to 012$, $2A_1 \to A_12$, $1A_1 \to 11$. Show that $L(G_1) = \{0^n 1^n 2^n / n \geq 1\}$

Sol- $S \to 012$ is a production so $012 \in L(G_1)$

$S \to OSA_12$

$\Downarrow$ (n-1) time

$\to O^{n-1} S(A_12)^{n-1}$

$\to O^{n-1} 012(A_12)^{n-1}$ $(\because S \to 012)$

$\to O^{n-1} A_1^{n-1} 2^n$ $(\because 2A_1 \to A_12)$

$\to O^n 1^n 2^n$

$\therefore O^n 1^n 2^n \in L(G_1)$

Q.2) Construct a PDA 'A' accepting $L = WCW^T$ where $W \in (a,b)^*$ by final state?

Sol- Let $WCW^T \in L$

$W = a_1 a_2 \ldots a_n$ where a_i is either a or b.

$W^T = a_n a_{n-1} \ldots a_2 a_1$

$(q_0, WCW^T, Z_0) \to (q_0, a_1 a_2 \ldots a_n CW^T, Z_0)$

$\downarrow$

$(q_0, CW^T, a_n \ldots a_2 a_1 Z_0)$ Rule(1)

$\downarrow$

$(q_1, a_n = a_2 a_1, a_n - a_2 a_1 Z_0)$ Rule(4)

$\downarrow$

$(q_1, \wedge, Z_0)$ Rule(5)

$\downarrow$

(q_f, Z_0)

$\therefore WCW^T \in T(A)$

Q.3) Design a TM to recognise string consist of even no. of 1s?

Sol: Let assume M is Turing Machine

$M = (\{q_1, q_2\}, \{1, b\}, \{1, b\}, \delta, q_1, b, \{q_1\})$

Present State	Tape symbol (1)
$\to (q_1)$	bRq_2
q_2	bRq_1

$q_1 11 \to bq_2 1 \to bbq_1$ $\therefore$ Acceptable by even no. of 1's.

Date:
Page:

Language Acceptability

Q:1. Consider the TM as given below -

Present State	Tape Symbol 0	1	x	y	b
→ q_1	xRq_2				bRq_5
q_2	$0Rq_2$	yLq_3		yRq_2	
q_3	$0Lq_4$		xRq_3	yLq_3	
q_4	$0Lq_4$		xRq_1		bRq_6
q_5				yRq_5	
((q_6))					

Sol- Describe the process for - (I.) 011
(II.) 001
(III.) 0011

(I) 011

$q_1011 \rightarrow xq_201 \rightarrow q_3xy1 \rightarrow xq_5y1 \rightarrow xyq_51$

As transition $\delta(q_5, 1)$ is not defined, M halts
So string 011 is not accepted.

(II) 001

$q_1001 \rightarrow xq_201 \rightarrow x0q_21 \rightarrow xq_30y \rightarrow q_4x0y$

$xxyq_2 \leftarrow xxq_2y \leftarrow xq_10y$

M halts. As q_2 is not an accepting state,
So 001 is not accepted by M.

(III) 0011

$q_10011 \rightarrow xq_2011 \rightarrow x0q_211 \rightarrow xq_30y1 \rightarrow q_4x0y1$

$xxq_3yy \leftarrow xxyq_21 \leftarrow xxq_2y1 \leftarrow xq_10y1$

$xq_3xyy \rightarrow xxq_5yy \rightarrow xxyq_5y \rightarrow xxyyq_5b \rightarrow xxyybq_6$

M halts. As q_6 is an accepting State
So 0011 is accepted by Machine (M).

VV IMP

Date:
Page:

→ 112233

Q 2) Design TM to recognise the language $\{1^n 2^n 3^n, n \geq 1\}$.

Sol- 1. q_1 as initial state. Read/write head scan leftmost 1 and replace it by "b", and move towards Right and enter q_2.

2. On scanning leftmost '2', R/W head replaced 2 by b & moves to Right & enter in q_3.

3. on scanning leftmost '3', R/W head replaced 3 by b & moves to Right & m enter in q_4.

4. After scanning the right most 3, R/W moves to the left until it finds the leftmost '1'. As the result, the leftmost 1,2,3 are replaced by b.

	Present State	Input Tape Symbol 1	2	3	b
— fixed part to remember	→ q_1	$\underline{bRq_2}$			
	q_2	$1Rq_2$	$\underline{bRq_3}$	$\underline{bRq_4}$	bRq_2
	q_3		$2Rq_3$	bRq_4	bRq_3
	q_4			$3Lq_5$	bLq_1
	q_5	$1Lq_6$	$2Lq_5$		bLq_5
	q_6				bRq_1
	(q7)				—

$q_1 112233 \rightarrow bq_2 12233 \rightarrow blq_2 2233 \rightarrow blbq_3 233$

$\downarrow$

$blbq_5 2b3 \leftarrow blb2q_5 b3 \leftarrow blb2bq_4 3 \leftarrow blb2q_4 33$

$\downarrow$

$blq_5 bl \rightarrow bq_5 lb2b3 \rightarrow q_6 blb2b3 \rightarrow bq_1 lb2b3$

$\downarrow$

$bbbbq_3 b3 \leftarrow bbbq_2 2b3 \leftarrow bbq_2 b2b3$

$\downarrow$

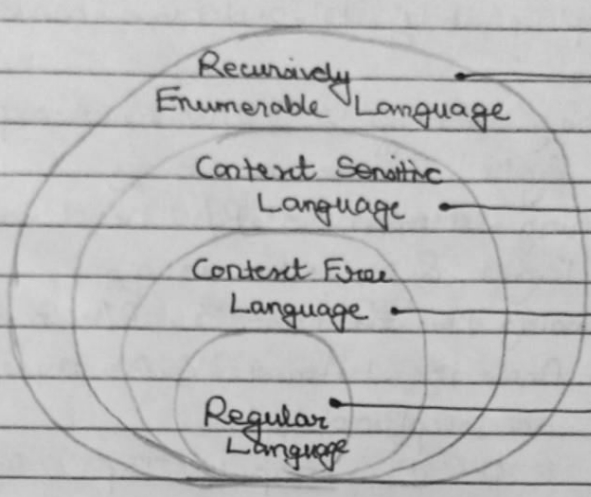
$bbbbbq_3 3 \rightarrow bbbbbbq_4 b \rightarrow bbbbbq_7 bb$

$\therefore$ 112233 is accepted.

Date:
Page:

Chomsky Classification of Language

			Automation / Application:
REL	Type 0	Recursively Enumerable Language	→ Turing Machine
CSL	Type 1	Context Sensitive Language	→ Linear Bounded Automation (LBA)
CFL	Type 2	Context Free Language	→ Push down Automation (PDA)
RL	Type 3	Regular Language	→ Finite Automata



Relation → $T_0 \subseteq T_1 \subseteq T_2 \subseteq T_3$

More Explanation →

Type 0: Unrestricted Grammar / Language :

$\phi A \psi \rightarrow \phi \alpha \psi$ ⟶ Production

Type 1: Context Sensitive Grammar / Language :

$\phi A \psi \rightarrow \phi \alpha \psi$ ⟶ Production

$\alpha \neq \Lambda$ production of Λ's not permitted

Type 2: Context Free Grammar / Language:

$A \rightarrow \alpha$

where $A \in V_n$ (variable) ⟶ Production

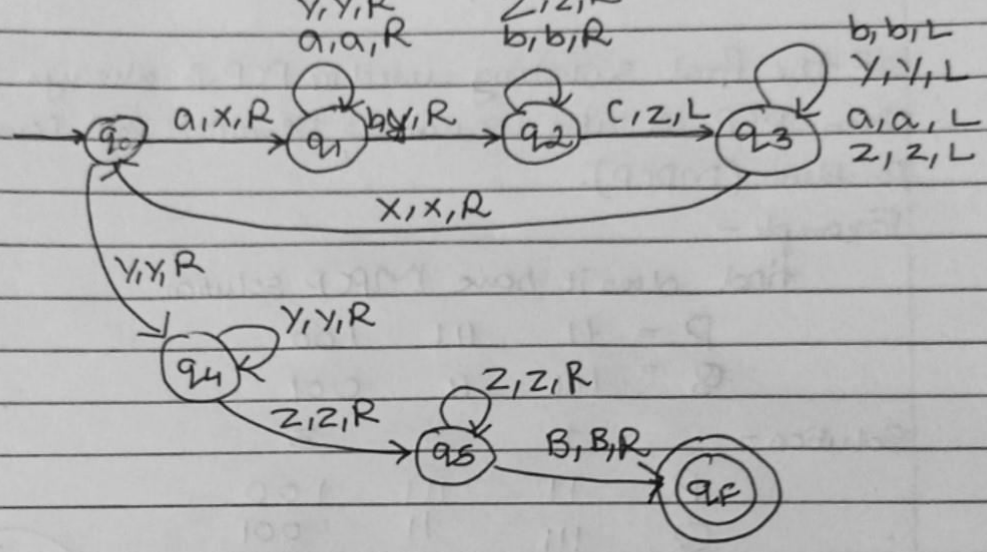
$\alpha \in (V_n \cup \Sigma)^*$ (variable ∪ Terminal)

Type 3: Regular Grammar / Language :

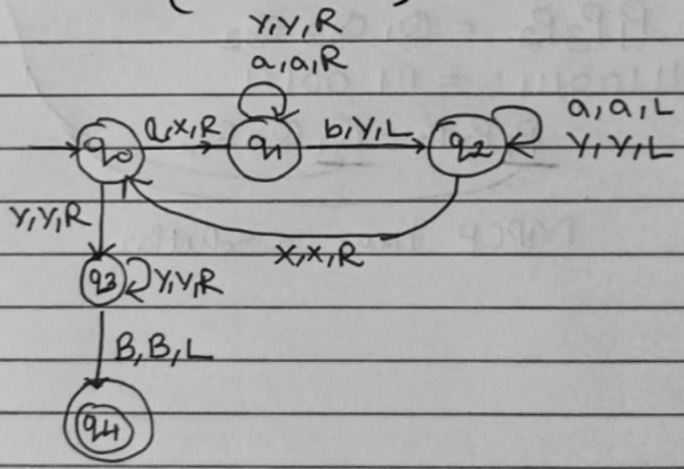
$S \rightarrow \Lambda$ ⟶ Production

↳ but S is not comes RHS of any production.

Construct TM for $a^n b^n c^n \mid n \geq 1$ →



Construct TM for $\{a^n b^n \mid n \geq 1\}$ →



Construct PDA for accepting the language $L = \{a^n b^m \mid n, m \geq 1\}$.

$L = \{aab, abb, aaab, abbb, \ldots\}$

Stack (S) →

$\delta(q_0, a, Z) \rightarrow (q_0, aZ)$

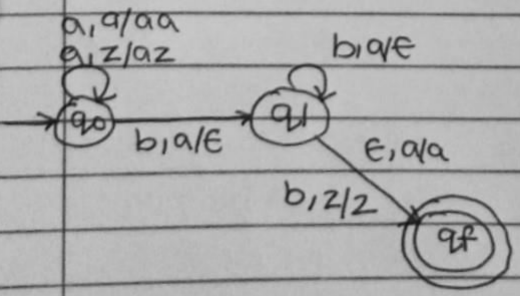
$\delta(q_0, a, a) \rightarrow (q_0, aa)$

$\delta(q_0, b, a) \rightarrow (q_1, \epsilon)$

$\delta(q_1, b, a) \rightarrow (q_1, \epsilon)$

$\delta(q_1, \epsilon, a) \rightarrow (q_f, a)$

$\delta(q_1, b, Z) \rightarrow (q_f, Z)$



Modified Post Correspondence Problem (MPCP) →

If the first substring used in PCP is always x_1 & y_1 then PCP is also known as Modified Post Correspondence Problem (MPCP).

Example –

find does it have MPCP solution

P = 11 111 100

Q = 111 11 001

Solution –

	1	2	3
P	11	111	100
Q	111	11	001

$P_1 P_3 P_2 = Q_1 Q_3 Q_2$

11100111 = 111 00111

$\because P_1 P_3 P_2 = Q_1 Q_3 Q_2$

$\therefore$ MPCP have a solution

$P_1 = Q_1$ is called MPCP rest of all is same as PCP.

Moore Machine to Mealy →

Present State	Next State a=0	Next State a=1	Output
→q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

Sol–

Mealy Machine –

Present State	Next State a=0 State	Next State a=0 output	Next State a=1 State	Next State a=1 output
→q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

Mealey to Moore Machine →

Present State	Next State a=0 State	Next State a=0 output	Next State a=1 State	Next State a=1 output
→q_1	q_2	Z_1	q_3	Z_1
q_2	q_2	Z_2	q_3	Z_1
q_3	q_2	Z_1	q_3	Z_2

Sol–

Moore Machine →

Present State	Next State a=0	Next State a=1	Output
→q_1	q_{21}	q_{31}	∧
q_{21}	q_{22}	q_{31}	Z_1
q_{22}	q_{22}	q_{31}	Z_2
q_{31}	q_{21}	q_{32}	Z_1
q_{32}	q_{21}	q_{32}	Z_2

Arden's Theorem: $R = Q + RP$ for uniqueness $R = QP^*$.

Kleene's Theorem: $L = \bigcup_{j=1}^{n} P_1^m f_j$, L is in R.

Fastrack Revision All Care Concepts's Tuple →

(1) Finite Automata →
- Deterministic FA → $\{Q, \Sigma, \delta, q_0, F\}$ ∴ $\delta : Q \times \Sigma \to Q$
- Non Deterministic FA → $\{Q, \Sigma, \delta, q_0, F\}$ ∴ $\delta : Q \times \Sigma \to 2^Q$

] without output

- Mealy Machine → λ = present state × present input
- Moore Machine → λ = present state

] with output

(2) Mealy Machine → $\{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$ ∵ λ = present state, present input

Moore Machine → $\{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$ ∵ λ = present state

(3) Derivation Tree → $G = (V_n, \Sigma, P, S)$.

(4) Context Free Grammar → $G = (V, T, P, S)$

(5) Pushdown Automata → $\{Q, \Sigma, \gamma, q_0, Z_0, F, \delta\}$. ∵ $\delta = Q \times (\Sigma \cup \{\wedge\}) \times \gamma$

(6) Turing Machine → $\{Q, \Sigma, T, q_0, \delta, b, F\}$ ∵ $\delta = Q \times T \times \{L, R\}$

...ping Lemma:

$S = xyz$ →
- I. $xy^i z$, $i \geq 0$
- II. $|xy| \leq p$
- III. $|y| > 0$

...omsky Classification of Language:

$\phi A \psi \rightarrow \phi \alpha \psi$ ← Type 0 → Recursively Enumerable Language → Turing Machine

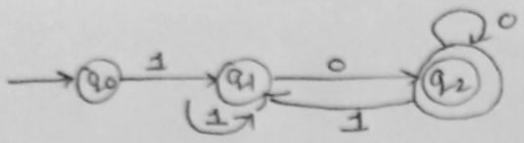
$\phi A \psi \rightarrow \phi \alpha \psi$ $\because \alpha \neq \wedge$ ← Type 1 → Content Sensitive Language → Linear Bounded A...

$A \rightarrow \alpha$, $A \in V_n$, $\alpha \in (V_n \cup \Sigma)^*$ ← Type 2 → Content Free Language → Push down Auto...

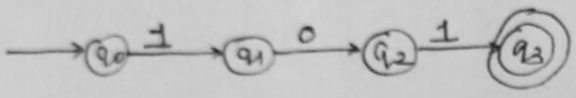
$S \rightarrow \wedge$ ← Type 3 → Regular Language → Finite Automata

Design a DFA with $\Sigma = \{0,1\}$ accepts those strings —

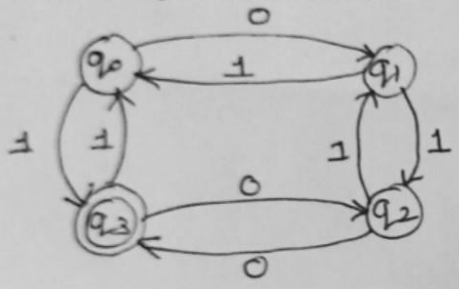
1. Starts with 1 & ends with 0.



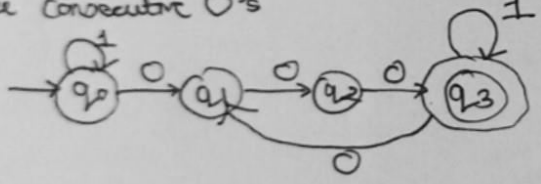
2. Accepts the only input 101.



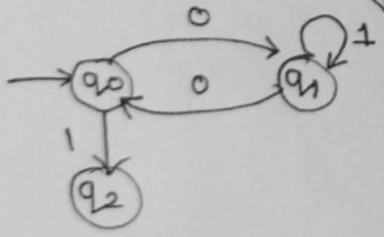
3. Even no. of 0's and 1's.



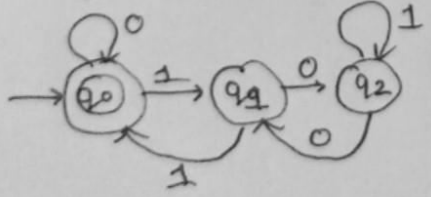
4. Three consecutive 0's



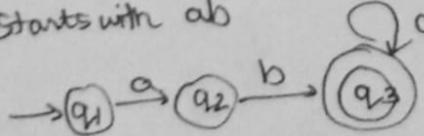
5. Even no of 0's followed by single 1.



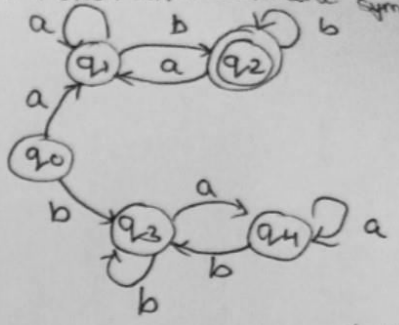
6. Divisible by 3 over $\Sigma\{0,1\}$



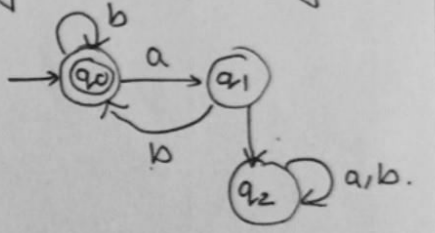
7. Starts with ab



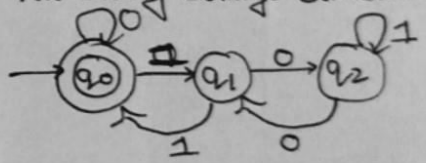
8. Starts & ends with different symbol



9. every "a" is followed by "b"



10. All binary strings divisible by 3



Present State	Tape Symbol				
	0	1	x	y	b
→ q_1	xRq_2				bRq_5
q_2	$0Rq_2$	yLq_3		yRq_2	
q_3	$0Lq_4$		xRq_5	yLq_3	
q_4	$0Lq_4$		xRq_1		
q_5				yRq_5	bRq_6
(q_6)					

(I) 011 →

$q_1011 \rightarrow xq_211 \rightarrow q_3xy1 \rightarrow xq_5y1 \rightarrow xyq_51 \rightarrow$ HALT

(II) 0011 →

~~$q_10011 \rightarrow 0q_2011 \rightarrow 00q_211 \rightarrow 0q_30y1$~~

$q_10011 \rightarrow xq_2011 \rightarrow x0q_211 \rightarrow xq_30y1 \rightarrow q_4x0y1$

$\downarrow$

xq_10y1

$\downarrow$

xxq_2y1

$\downarrow$

$xxyq_21 \leftarrow xxq_3yy \leftarrow xq_3xyy \leftarrow q_3xxyy$

$\downarrow$

xxq_5yy

$\downarrow$

$xxyyq_5b$

$\downarrow$

$xxyybq_6$

M halts as q_6 is an accepting state.

So I/p string 0011 is accepted.