SIMATS
School of Engineering

Theory of Computation

Computer Science and Engineering

Saveetha Institute of Medical And Technical Sciences,Chennai.

MATHEMATICAL PRELIMINARIES:

INTRODUCTION TO FORMAL PROOF:

Proof:- A proof is a conveneing argument that some statement is true.

*** Deductive Proof :- (Direct Proof):**

- Sequence of statements whose truth leads us from some initial statements called hypothesis to a conclusion statement.

Ex:
Prove that of $x \geq 4$ then $2^x \geq x^2$.

Proof:- when $x = 4$ then

$$2^x = 2^4 = 16$$
$$x^2 = 4^2 = 16$$
$$\Rightarrow 2^x = x^2.$$

As x grows larger than 4, 2^x doubles each time x increase of one.

$\therefore$ Each time x increases above 4, 2^x grows more than x^2.

Hence Proved.

ADDITIONAL FORM OF PROOFS:

*** Proof by Contrapositive :-**

- The contrapositive of a statement if H then C is if not C then not H.

To prove a statement it is enough to prove contrapositive.

Ex:
Prove that for any integer i, j and n if $i * j = n$ then either $i \leq \sqrt{n}$ or $j \leq \sqrt{n}$.

Proof:- The given statement if $i * j = n$ then either $i \leq \sqrt{n}$ or $j \leq \sqrt{n}$.

The contrapositive statement of given is,
$i > \sqrt{n}$ and $j > \sqrt{n}$ then $i * j \neq n$

Let us prove, contrapositive statement
$i > \sqrt{n}$ —(1), $j > \sqrt{n}$ —(2)

(1) $\Rightarrow$ $i > \sqrt{n}$ Multiply both sides by j
$i * j > \sqrt{n} * j$ —(3)

(2) $\Rightarrow$ $j > \sqrt{n}$ Multiply both sides by $\sqrt{n}$
$\sqrt{n} \times j > \sqrt{n} \times \sqrt{n}$, $\sqrt{n} \times j > n$ —(4)

From (3) & (4), $i * j > \sqrt{n} * j > n$

$i * j > n$, $i * j \neq n$.

The contrapositive of given statement is true. Hence the given statement also True.

INDUCTIVE PROOFS:

MATHEMATICAL INDUCTION :-

$P(n)$ is a statement involving an integer n, to show $P(n)$ is true for all $k \geq n_0$. This proof needs,

1. $P(n_0)$ is true
2. If $P(k)$ is true, then $P(k+1)$ is true for $k \geq n_0$.

Ex:
S.T $1+2+3+\cdots+n = \frac{n(n+1)}{2}$.

Step1: Basis:

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

Hence the proof.

Step2: Induction:

Assumption is,
$1+2+3+\cdots+k = \frac{k(k+1)}{2}$ is true.

To prove, $1+2+3+\cdots+(k+1) = \frac{(k+1)(k+2)}{2}$ is true.

L.H.S $\underbrace{1+2+3+\cdots+k}+(k+1)$

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

$= R.H.S$

Hence the proof.

CENTRAL CONCEPT OF AUTOMATA THEORY

FINITE AUTOMATA :-

A Finite Automata is formally denoted by five tuple,

$(Q, \Sigma, \delta, q_0, F)$ where

Q is the set of states

Σ is the Input Alphabet

δ is the transition function

q_0 is Initial State

F is final set of states.

Types of finite Automata :-

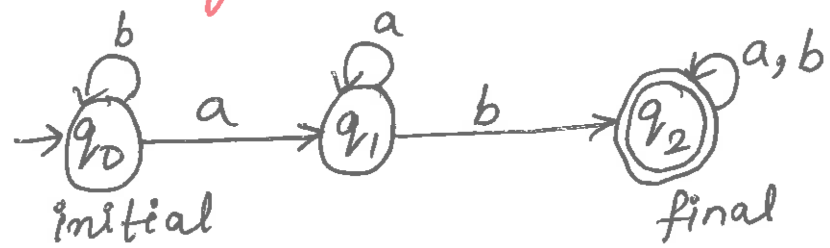
1. Deterministic Finite Automata (DFA)
2. Non-Deterministic Finite Automata (NFA)

DFA	NFA
1. From each state for each input symbol there is exactly one transition.	1. For each state for each input symbol we can have 0 or more transitions.
2. No ϵ-transitions	2. ϵ-transitions are allowed.

DFA Problems

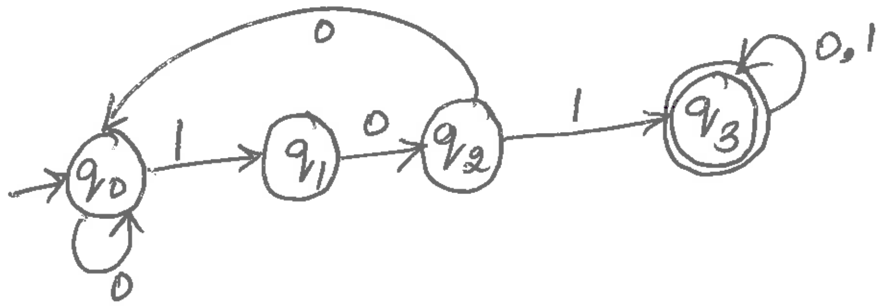
Ex:1

Design a DFA for the language having strings with ab as a substring over $\Sigma = \{a, b\}$.



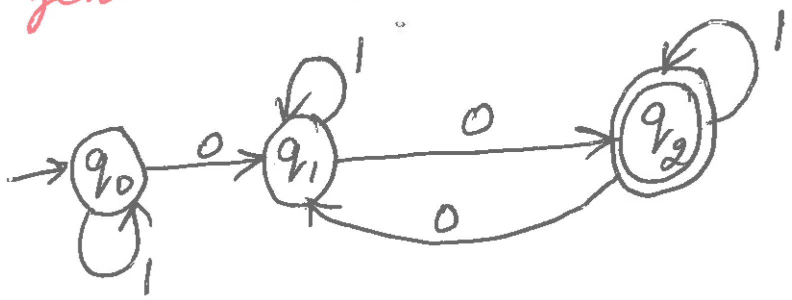
Ex:2

Design a DFA for the language having strings with 101 as a substring over $\Sigma = \{0, 1\}$.



Ex:3

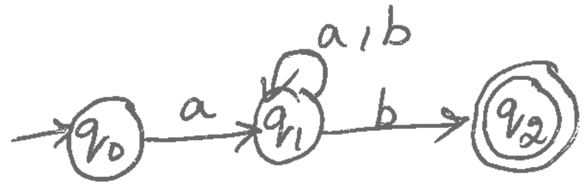
Design a DFA for the language having strings with even no of zero's over $\Sigma = \{0, 1\}$.



NFA problems

Ex:1

Design a NFA to accept strings that start with A and end with b over $\Sigma = \{a, b\}$ also write formula def. of NFA. check whether the string abaab is accepted or not!



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

$\{(q_0, q_1, q_2), \{a, b\}, \delta, q_0, q_2\}$

where δ is,

$\delta(q_0, a) = q_1$, $\delta(q_0, b) = \phi$

$\delta(q_1, a) = q_1$ $\quad$ $\delta(q_2, a) = \phi$

$\delta(q_1, b) = q_1$

$\delta(q_1, b) = q_2$

String : abaab

$\delta(q_0, abaab) = \delta(q_1, baab)$

$= \delta(q_1, aab)$

$= \delta(q_1, ab)$

$= \delta(q_1, b)$

$= q_2$ //

EQUIVALENCE OF NFA & DFA:

CONVERSION NFA to DFA :-

Ex: Construct a DFA equivalent to NFA given below,

$M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\})$

where $\delta(q_0,0) = \{q_0, q_1\}$

$\delta(q_0,1) = \{q_1\}$

$\delta(q_1,0) = \phi$

$\delta(q_1,1) = \{q_0,q_1\}$

Sol: Let M' be the DFA, let $[q_0]$ be the initial state of DFA, let δ' be the transition function of DFA,

$\delta'([q_0],0) = \delta(q_0,0) = \{q_0,q_1\}$

$\delta'([q_0],1) = \delta(q_0,1) = \{q_1\}$

$\delta'([q_0,q_1],0) = \delta(q_0,0) \cup \delta(q_1,0) = \{q_0,q_1\}$

$\delta'([q_0,q_1],1) = \delta(q_0,1) \cup \delta(q_1,1) = \{q_0,q_1\}$

$\delta'([q_1],0) = \delta(q_1,0) = \phi$

$\delta'([q_1],1) = \delta(q_1,1) = \{q_0,q_1\}$.

ANS:

States	0	1
$\{q_0\}$	$\{q_0,q_1\}$	$\{q_1\}$
$\{q_0,q_1\}$	$\{q_0 q_1\}$	$\{q_0,q_1\}$
$\{q_1\}$	ϕ	$\{q_0,q_1\}$

FINITE AUTOMATA with Epsilon Transistions:

EQUIVALENCE OF NFA with Epsilon Transistions:-

Ex: construct NFA from NFA with Epsilon,

$\epsilon\text{-closure}(q_0) = \{q_0, q_1\}$

$\epsilon\text{-closure}(q_1) = \{q_1\}$.

Processing of q_0:

$\delta(q_0,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0,\epsilon),0))$

$= \epsilon\text{-closure}(\delta(\{q_0,q_1\},0))$

$= \epsilon\text{-closure}(q_0) = \{q_0,q_1\}$.

$\delta(q_0,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0,\epsilon),1))$

$= \epsilon\text{-closure}(\delta(\{q_0,q_1\},1))$

$= \epsilon\text{-closure}(q_1) = \{q_1\}$

Processing of q_1:

$\delta(q_1,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1,\epsilon),0))$

$= \epsilon\text{-closure}(q_0) = \{q_0,q_1\}$.

$\delta(q_1,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1,\epsilon),1))$

$= \epsilon\text{-closure}(q_1) = \{q_1\}$.

ANS:

States	0	1
q_0	$\{q_0,q_1\}$	$\{q_1\}$
q_1	$\{q_0,q_1\}$	$\{q_1\}$

⇓

NFA without ϵ-Transistions.

CONVERSION OF NFA-ε to DFA:

Ex: Convert the NFA-ε move given below to an Equivalent DFA.

$\epsilon\text{-closure of } q_0 = \{q_0, q_1, q_2\}$

$\epsilon\text{-closure of } q_1 = \{q_1, q_2\}$

$\epsilon\text{-closure of } q_2 = \{q_2\}$.

$\epsilon\text{-closure of } \{q_0\} = \{q_0, q_1, q_2\} \rightarrow A$

1. $\hat{\delta}(A,0) = \epsilon\text{-closure of } (\delta(A,0))$

$= \epsilon\text{-closure}(\delta(\{q_0,q_1,q_2\},0))$

$= \epsilon\text{-closure}(q_0) = \{q_0,q_1,q_2\}$

$\hat{\delta}(A,1) = \epsilon\text{-closure}(\delta(A,1))$

$= \epsilon\text{-closure}(q_1) = \{q_1,q_2\} \rightarrow B$

$\hat{\delta}(A,2) = \epsilon\text{-closure}(\delta(A,2)) = \epsilon\text{-closure}(q_2)$

$= \{q_2\} \rightarrow C$

2. $\hat{\delta}(B,0) = \epsilon\text{-closure}(\delta(B,0))$

$= \epsilon\text{-closure}(\delta(q_1,q_2),0)) = \phi$.

$\hat{\delta}(B,1) = \epsilon\text{-closure}(\delta(B,1)) = \epsilon\text{-closure}(q_1)$

$= \{q_1,q_2\} \rightarrow B$

$\hat{\delta}(B,2) = \epsilon\text{-closure}(\delta(B,2)) = \epsilon\text{-closure}(q_2)$

$= \{q_2\} \rightarrow C$

3. $\hat{\delta}(C,0) = \epsilon\text{-closure}(\delta(C,0)) = \phi$

$\hat{\delta}(C,1) = \phi$.

$\hat{\delta}(C,2) = \epsilon\text{-closure}(q_2) = \{q_2\} \rightarrow C$

ANS:

States	0	1	2
A	A	B	C
B	ϕ	B	C
C	ϕ	ϕ	C

REGULAR LANGUAGE

Regular Expression

Following Languages by Regular Expression.

Set of Strings of a's and b's of length two

$(a+b)(a+b)$

Set Containing Zero or More 0's followed by Single 1

$(0+0^*1)$

Set of all strings ending With aba

$(a+b)^*aba$

Set over {1} having odd length of String

$1(11)^*$

The Set of all strings 0's is divisible by five is,

$1^*(00000)^*1^*$

Set of all strings abb as Substring

$(a+b)^*abb(a+b)^*$

String ends with 1 and does not Contains the Substring 00

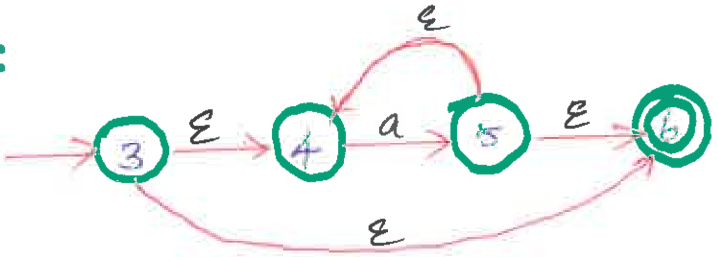
$(1+10)^*(101+1)^*1$

Equivalence of Finite Automata and Regular Expression

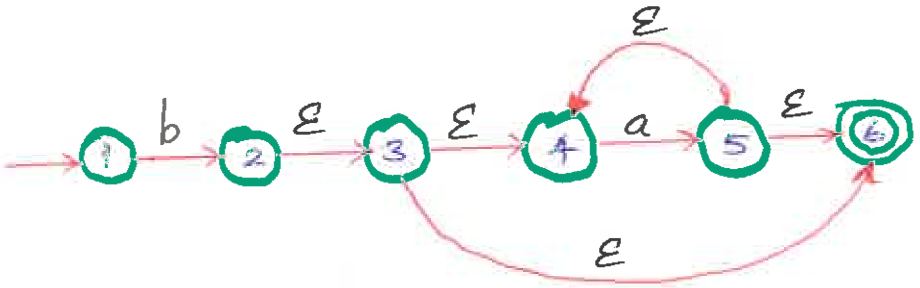
Construction of ε-NFA from the Regular Expression : $b+ba^*$

b:

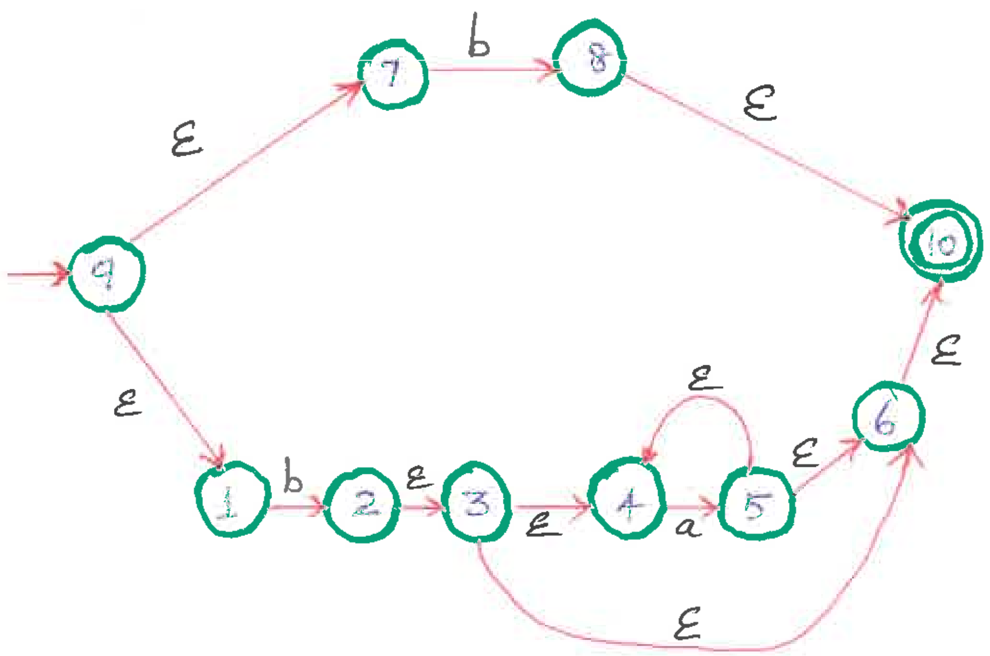
a^*:



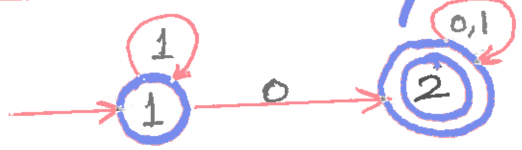
ba^*:



$b+ba^*$:



DFA TO REGULAR EXPRESSION



Let $k=0$,

$R_{11}^{(0)} = \varepsilon+1$

$R_{12}^{(0)} = 0$

$R_{21}^{(0)} = \phi$

$R_{22}^{(0)} = \varepsilon+0+1$

IF $k=1$,

$R_{ij}^{(k)} = R_{ij}^{k-1} + R_{ik}^{k-1}\left(R_{kk}^{k-1}\right)^* \cdot R_{kj}^{(k-1)}$

$R_{11}^{(1)} = (\varepsilon+1)^*$

$R_{12}^{(1)} = 1^*0$

$R_{21}^{(1)} = \phi$

$R_{22}^{(1)} = \varepsilon+0+1$

IF $k=2$,

$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}\left(R_{22}^{(1)}\right)^* R_{22}^{(1)}$

$= 1^*0 + 1^*0(\varepsilon+0+1)^*(\varepsilon+0+1)$

$= 1^*0 + 1^*0(\varepsilon+0+1)^*$

$R_{12}^{(2)} = 1^*0(\varepsilon+0+1)^*$

$\boxed{R_{12}^{(2)} = 1^*0(0+1)^*}$

MINIMIZATION OF DFA

1)

States	a	b
→ q_0	q_1	q_6
q_1	q_6	q_2
q_2	q_3	q_1
q_3	q_3	q_0
q_4	q_3	q_5
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	q_3

Input : a

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

Input : b

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

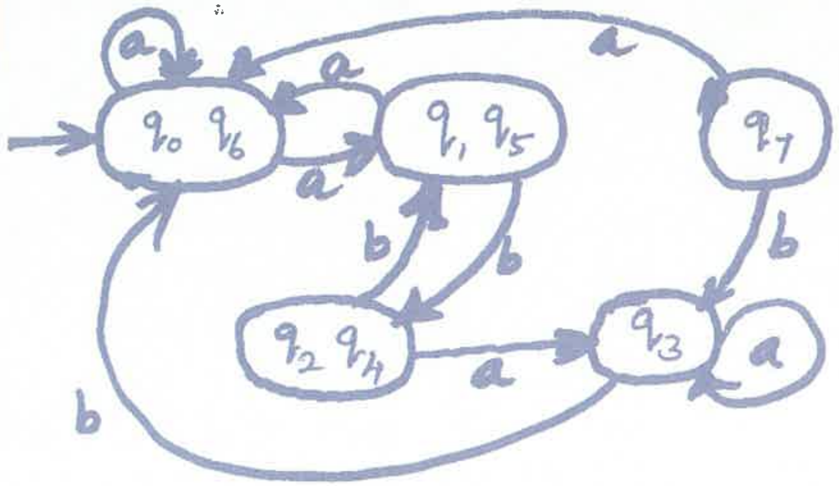
$\{q_0, q_1, q_5, q_6, q_7\}$ $\{q_2\ q_4\}$ $\{q_3\}$

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

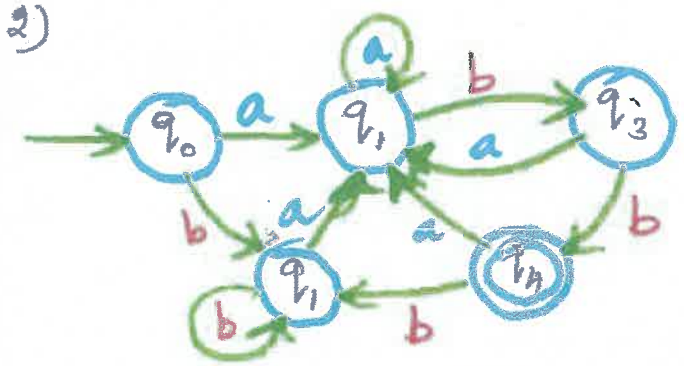
TRANSITION TABLE

	a	b
$q_0\ q_6$	$q_1\ q_5$	$q_0\ q_6$
$q_1\ q_5$	$q_0\ q_6$	$q_2\ q_4$
q_7	$q_0\ q_6$	q_3
$q_2\ q_4$	q_3	$q_1\ q_5$
q_3	q_3	$q_0\ q_6$

TRANISTION DIAGRAM :



2)



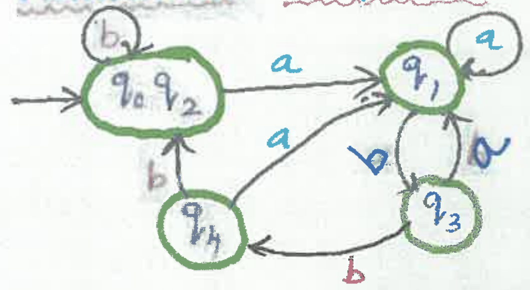
Input : b $\{q_0, q_1, q_2, q_3\}$ $\{q_4\}$

Input : b $\{q_0, q_1, q_2\}$ $\{q_3\}$ $\{q_4\}$

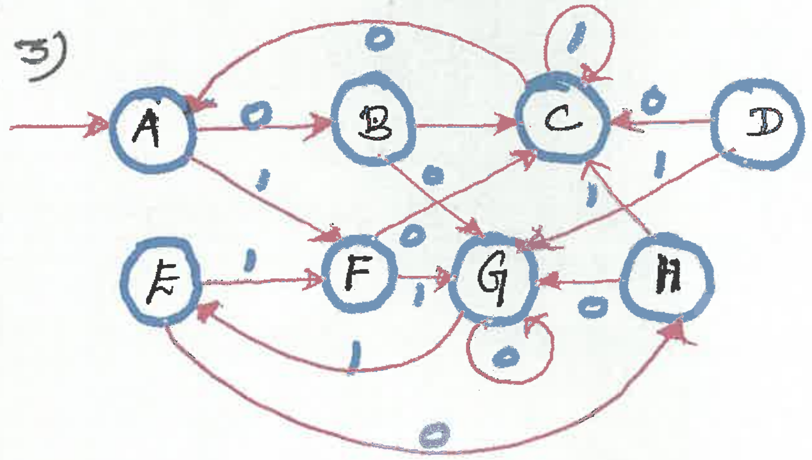
TRANSITION TABLE

STATE	a	b
$q_0\ q_2$	q_1	$q_0\ q_2$
q_1	q_1	q_3
q_3	q_1	q_4
q_4	q_1	$q_0\ q_2$

TRANSITION DIAGRAM :



3)



Input : 0

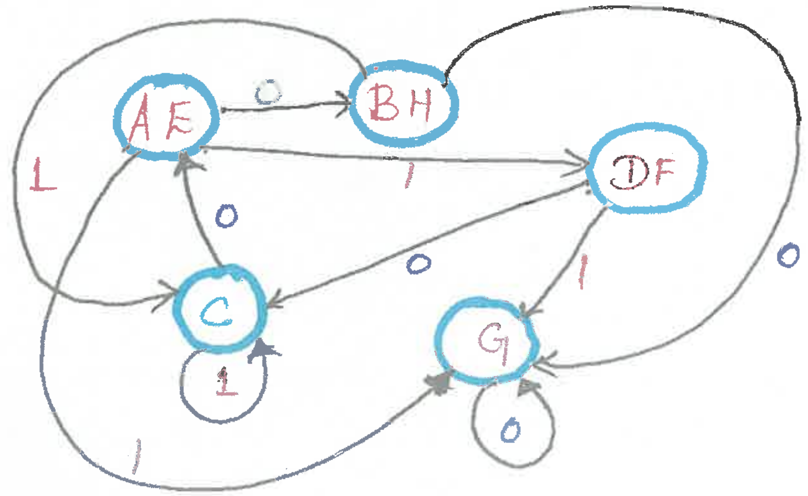
$\{A\ B\ D\ E\ F\ G\ H\}$ $\{C\}$

Input : 1

$\{A\ B\ E\ G\ H\}$ $\{B\ H\}$ $\{D\ F\}$ $\{C\}$

$\{A\ E\}$ $\{G\}$ $\{B\ H\}$ $\{D\ F\}$ $\{C\}$

TRANSITION DIAGRAM :



PUMPING LEMMA FOR REGULAR SET:

Pumping Lemma:

Let : $L \rightarrow$ Regular Language

$n \rightarrow$ Constant

Such that for Every string in 'W' in 'L'

Such that $|W| \geq n$

We can break W into three

* $y \neq \varepsilon$
* $|xy| \leq n$
* $xy^kz \in L \;\forall\; k \geq 0$

Note : Repeating y any no of times (or) Deleting y keeps the resulting string in the same language.

Problem Based on Pumping Lemma:

Prove that the set

$L = \{0^{i^2} \mid i \text{ is an integer}, i \geq 1\}$

Which consist of all strings of 0's whose length is perfect square is not regular

Assume the given Language L is a regular

Let us take the sample string $W = 0^{n^2}$

Where n is the constant of pumbing lemma

By pumbing lemma :

We can write, $0^{n^2} = xyz$

Where

1) $y \neq e$
2) $|xy| \leq n$
3) $xy^kz \in L \;\forall\; k \geq 0$

put $k=2$, We get the string xy^2z

By pumbing lemma,

$xy^2z \in L$

$|xy^2z|$ Must be a perfect square

Let us find $|xy^2z|$

$|x^2y^2z| = |xyz| + |y|$

$\leq n^2+n$ $\quad \therefore |xy| \leq n$ and $y \neq e$

$|xy^2z| > n^2 \longrightarrow (1)$

$|xy^2z| < (n+1)^2 \longrightarrow (2)$

From (1) and (2)

$|xy^2z|$ lies properly b/w two perfect square and hence $|xy^2z|$ is not a perfect square.

$\therefore xy^2z \notin L$ Which is a Contradiction that difference

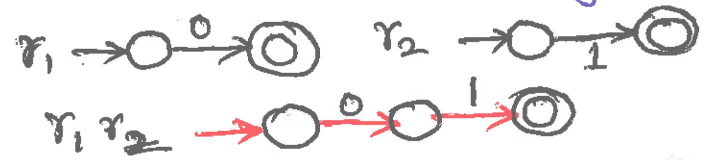
Given Language is not regular.

CLOSURE PROPERTIES OF REGULAR LANGUAGES

1) Union of two regular Language is Regular.

$L(m) = L(m_1) \cup L(m_2)$

2) Concatenation of two regular Languag is Regular.



3) Intersection of two regular Languge is Regular

$L \cap M$ is also regular

4) The difference of two regular Language is Regular

$L - M$ is Regular $\quad L - M = L \cap \overline{M}$ Regular.

5) The Compliment of Regular Language is Regular

$\overline{L} = \Sigma^* - L$

6) The reversal of a regular Language is Regular

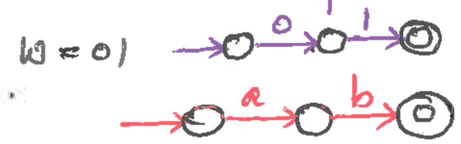
L^R Regular.

7) Closure of Regular is Regular.

$w = a \quad w = a^*$

8) The Homomorphism of Regular Language is Regular

$h(0) = a$, $h(1) = b$



Hence $h(L)$ also Regular.

9) The Inverse homomorphism of a regular Language is Regular.

UNIT-III GRAMMARS AND APPLICATIONS

GRAMMAR INTRODUCTION:

TYPES OF GRAMMAR:

Grammar denotes syntactical rules in languages.

Types:-

Grammar Type	Grammar Accepted	Language Accepted	Automation
Type0	Unrestricted Grammar	Recursively Enumerable Language.	Turing Machine.
Type1	Context Sensitive Grammar	Context Sensitive Language.	Linear Bounded Automata
Type2	Context Free Grammar	Context Free Language	Pushdown Automata
Type3	Regular Grammar (or) Regular Expression	Regular Language	Finite Automata

CONTEXT FREE GRAMMAR (CFG):-

Context Free Grammar (CFG),

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

where, V = Set of Non-Terminals

T = Set of Terminals

P = Set of productions

S = Start Symbol.

CONTEXT FREE GRAMMAR & LANGUAGES:-

DERIVATIONS & PARSE TREE:

Ex:1 Consider the grammar

$S \to aB \mid bA$

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

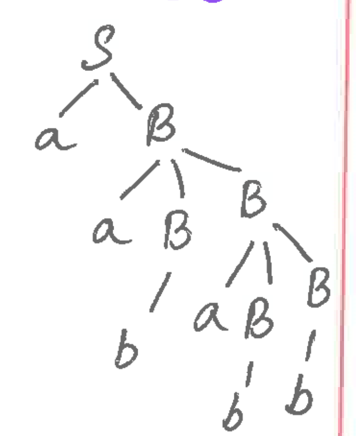
$B \to b \mid bS \mid aBB.$

write leftmost and rightmost derivations and draw parse tree for the string aababb.

LMD:-

$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aabB$
$\Rightarrow aabaBB$
$\Rightarrow aababB$
$\Rightarrow aababb$

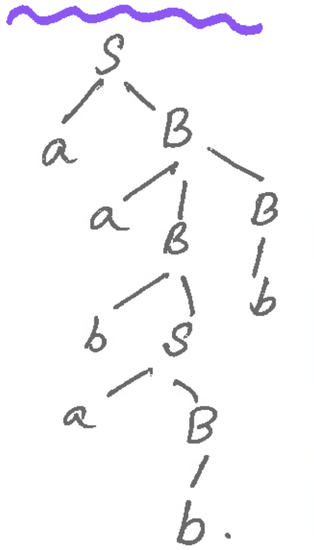
Parse Tree



RMD:-

$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aaBaBB$
$\Rightarrow aaBaBb$
$\Rightarrow aaBabb$
$\Rightarrow aababb.$

Parse-Tree



Ex:2

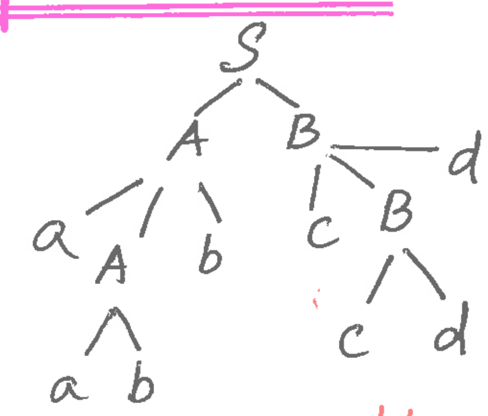
Consider the grammar G:

$S \to AB/c, \quad A \to aAb/ab$

$B \to cBd \;/cd, \quad C \to aCd \;/aDd$

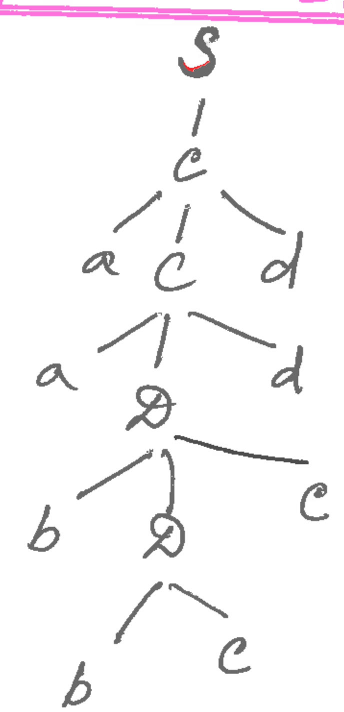
$D \to bDc \,/bc$. Show that the grammar G is Ambiguous for the input string "aabbccdd".

LMD PARSE TREE:



$\Rightarrow$ aabbccdd//

RMD PARSE TREE:



$\Rightarrow$ aabbccdd//

Thus G is an Ambiguous Grammar.

SIMPLIFICATION OF CFG :

* ELIMINATION OF ϵ-PRODUCTION :-

Ex:1 Eliminate ϵ-productions from the grammar.

$S \to AB$
$A \to aAA / \epsilon$
$B \to bBB / \epsilon$.

Sol:- S, A, B are nullable.

$$\boxed{\begin{array}{l} S \to AB / B / A \\ A \to aAA / aA / a \\ B \to bBB / bB / b. \end{array}}$$

Ex:2 Eliminate ϵ-productions from the grammar

$S \to XYZ$, $X \to 0X / \epsilon$, $Y \to 1Y / \epsilon$.

Sol:- X, Y are nullable

$$\boxed{\begin{array}{l} S \to XYZ / YZ / XZ / Z \\ X \to 0X / 0 \\ Y \to 1Y / 1. \end{array}}$$

* ELIMINATION OF UNIT PRODUCTION :-

Ex:1 Consider the grammar,

$E \to T / E + T$
$T \to F / T * F$
$F \to I / (E)$
$I \to a / b / Ia / Ib / I_0 / I_1$.

Sol:- Unit Productions are $E \to T$, $T \to F$, $F \to I$. After Eliminating Unit Productions

$$\boxed{\begin{array}{l} E \to E+T / T*F / (E) / a / b / Ia / Ib / I_0 / I_1 \\ T \to T*F / (E) / a / b / Ia / Ib / I_0 / I_1 \\ F \to (E) / a / b / Ia / Ib / I_0 / I_1. \\ I \to a / b / Ia / Ib / I_0 / I_1. \end{array}}$$

* ELIMINATION OF USELESS SYMBOLS :-

Ex: Consider the grammar

$S \to aB / bX$
$A \to Bad / bSX / a$
$B \to aSB / bBX$
$X \to SBD / aBX / ad$.

Sol:- Step1:- After removing non-generating symbols B,

$$\boxed{\begin{array}{l} S \to bX \\ A \to bSX \\ X \to ad \end{array}}$$

Step2:- After removing non-reachable symbol A,

$$\boxed{\begin{array}{l} S \to bX \\ X \to ad \end{array}}$$

CHOMSKY NORMAL FORM

$$\boxed{\begin{array}{l} NT \to Terminal \\ NT \to NT\ NT \end{array}}$$

Ex: Construct a grammar in chomsky Normal form equivalent to the grammar,

$S \to bA / aB$, $A \to bAA / aS / a$,
$B \to aBB / bS / b$.

Sol:- $A \to a$, $B \to b$, already in CNF.

Let us take, $S \to bA / aB$,
It can be converted to,

$S \to C_a B$
$C_a \to a$
$S \to C_b A$
$C_b \to b$.

Let us take $A \to bAA / aS / a$
It can be converted to,

$A \to C_b D_1$
$D_1 \to AA$
$C_b \to b$
$A \to C_a S$
$C_a \to a$
$A \to a$

Let us take $B \to aBB / bS / b$,
It can be converted to $\Rightarrow$

$$\boxed{\begin{array}{ll} B \to C_a D_2 & D_2 \to BB \\ B \to C_b S & C_b \to b \\ B \to b & \end{array}}$$

GREIBACH NORMAL FORM (GNF)

$\boxed{NT \rightarrow \text{One Terminal, any number of NTs.}}$

Lemma : 1

If $\left.\begin{array}{l} A \rightarrow \alpha_1 B \alpha_2 \\ B \rightarrow \beta_1 / \beta_2 / \dots / \beta_r \end{array}\right\}$ then $A \rightarrow \alpha_1 \beta_1 \alpha_2 / \alpha_1 \beta_2 \alpha_2 / \alpha_1 \beta_3 \alpha_2 / \alpha_1 \beta_4 \alpha_2 / \dots \dots \alpha_1 \beta_r / \alpha_2.$

Lemma : 2

If $\left.\begin{array}{l} A \rightarrow A\alpha_1 / A\alpha_2 \dots \\ A \rightarrow \beta_1 / \beta_2 / \dots \beta_r \end{array}\right\}$ then

$A \rightarrow \beta_i$

$A \rightarrow \beta_i B$

$B \rightarrow \alpha_i$

$B \rightarrow \alpha_i B.$

Ex: Convert to GNF for the grammar $G = (\{A_1, A_2, A_3\}, \{a, b\}, P, A_1)$ where P consist of the following

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 / b$

$A_3 \rightarrow A_1 A_2 / a.$

Sol: Let us consider, $A_3 \rightarrow A_3 A_1 A_3 A_2$

$A_3 \rightarrow b A_3 A_2 / a$

we can apply lemma 2,

Now, $A_3 \rightarrow A_3 \underbrace{A_1 A_3 A_2}_{} / \underbrace{b A_3 A_2}_{} / \underbrace{a}_{}$

$A \rightarrow A \alpha_1 \qquad \beta_1 \qquad \beta_2$

$A \rightarrow A\alpha_1$

$A \rightarrow \beta_1 / \beta_2$

$\Downarrow$

$A \rightarrow \beta_1, \quad A \rightarrow \beta_2$

$A \rightarrow \beta_1 B, \quad A \rightarrow \beta_2 B.$

$B \rightarrow \alpha_1$

$B \rightarrow \alpha_1 B.$

$A_3 \rightarrow A_3 A_1 A_3 A_2$

$A_3 \rightarrow b A_3 A_2 / a$

$\Downarrow$

$A_3 \rightarrow b A_3 A_2, \quad A_3 \rightarrow a$

$A_3 \rightarrow b A_3 A_2 B, \quad A_3 \rightarrow a B$

$B \rightarrow A_1 A_3 A_2$

$B \rightarrow A_1 A_3 A_2 B.$

Then we can apply lemma now, we get,

$A_1 \rightarrow b A_3 A_2 B A_1 A_3 / a B A_1 A_3 / b A_3 A_2 A_1 A_3 / a A_1 A_3 / b A_3$

$A_2 \rightarrow b A_3 A_2 B A_1 / a B A_1 / b A_3 A_2 A_1 / a A_1 / b$

$A_3 \rightarrow b A_3 A_2 / b A_3 A_2 B / a B / a$

$B \rightarrow b A_3 A_2 B A_1 A_3 A_3 A_2 / a B A_1 A_3 A_3 A_2 / b A_3 A_2 A_1 A_3 A_3 A_2 / a A_1 A_3 A_3 A_2 / b A_3 A_3 A_2 / b A_3 A_2 B A_1 A_3 A_3 A_2 B / a B A_1 A_3 A_3 A_2 B / b A_3 A_2 A_1 A_3 A_3 A_2 B / a A_1 A_3 A_3 A_2 B / b A_3 A_3 A_2 B.$

APPLICATIONS OF CONTEXT-FREE GRAMMARS

* Used in Parsing (Syntax Analysis) in compiler Design.
* Natural Language Processing.
* Human Activities Recognition.

CLOSURE PROPERTIES OF CONTEXT FREE LANGAUGES

* Union of two CFL's is context free
* Concatenation of two CFL's is context free.
* closure of a CFL is context free.
* Intersection of two CFL's is not context free.
* Intersection of a CFL and a regular language is context free.
* Complement of a CFL is not context free
* Substitution of a CFL is context free.
* Homomorphism of a CFL is context free.
* Inverse Homomorphism of a CFL is context free.

PUMPING LEMMA FOR CFL :-

Let 'L' be a context free language their exists a constant 'n', such that if 'z' is any string in L, such that $|z| \geq n$, then we can write, $z = uvwxy$, subject to following conditions,

i) $vx \neq \varepsilon$

ii) $|vwx| \leq n$

iii) $uv^iwx^iy \in L \quad \forall i \geq 0$.

PROBLEMS BASED ON PUMPING LEMMA :-

Ex:1 Show that the language $L = \{0^n 1^n 2^n \mid n \geq 1\}$ is not context free.

Sol:- Assume that the given language is context free,

Let us take the string $z = 0^n 1^n 2^n$ where n is constant of pumping lemma.

$z = uvwxy$, such that,

i) $vx \neq \varepsilon$, ii) $|vwx| \leq n$

iii) $uv^iwx^iy \in L$, $\forall i \geq 0$,

The string vwx cannot have all 3 symbols, 0's, 1's and 2's because $|vwx| \leq n$

case i) vwx has no 2's, $\therefore$ v & x consists of only 0's & 1's. put $i=0$, uv^iwx^iy, we get uwy,

$\therefore$ String uwy will have n no: of 2's but fewer than n 0's & 1's.

$\therefore$ $uwy \notin L$

which is a contradiction

$\therefore$ The given language is not context free.

Ex:2 Show that the language $L = \{0^i 1^j 2^i 3^j\}$ $i \geq 1$, $j \geq 1$, is not context free.

Sol:- Assume the given language is context free, let us take the string $z = 0^n 1^n 2^n 3^n$, where n is constant,

By Pumping Lemma,

$z = uvwxy$, such that

i) $vx \neq \varepsilon$,

ii) $|vwx| \leq n$

iii) $uv^iwx^iy \in L$, $\forall i \geq 0$,

The string vwx cannot involve all the symbols 0's 1's 2's & 3's.

It can have atmost two symbols,

case i)

vwx consists of only symbol

eg. Assume vwx consists of 0's,

put $i=0$ in uv^iwx^iy, the string uwy will have n no: of 1's, 2's & 3's. But fewer than n 0's.

The number of 0's and 2's donot match

$\therefore$ $uwy \notin L$ which is a contradiction

$\therefore$ The given language is not context free.

PUSH DOWN AUTOMATA

PDA Definition:

$$PDA : M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Design of PDA for the language $L = \{ 0^n 1^n / n \geq 1 \}$ by final State.

Solution:

Nature of the Problem:

Number of 0's are equal to Number of 1's

Execution procedure:

- q_0 – Accepting only one Zero
- q_1 – Accepting more Zero's till 1
- q_2 – Used to pop the Stack i/p Symbol 0
- q_3 – Accepting / Final State

MOVES AND INSTANTANEOUS DESCRIPTIONS

To design Moves:

$$\delta(q_0, 0, Z_0) = (q_1, 0Z_0)$$
$$\delta(q_1, 0, 0) = (q_1, 00)$$
$$\delta(q_1, 1, 0) = (q_2, \varepsilon)$$
$$\delta(q_2, 1, 0) = (q_2, \varepsilon)$$
$$\delta(q_2, \varepsilon, Z_0) = (q_3, Z_0)$$

Then PDA $M = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, q_3)$

Where $Q = \{q_0, q_1, q_2, q_3\}$

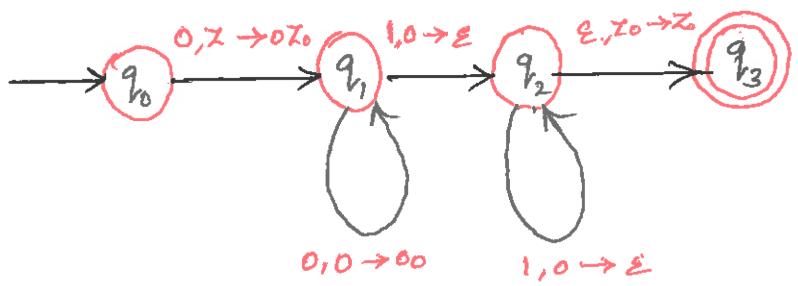
$\Sigma = \{0, 1\}$

$\Gamma = \{Z_0, 0\}$

q_0 is initial State

Z_0 is Stack Start Symbol

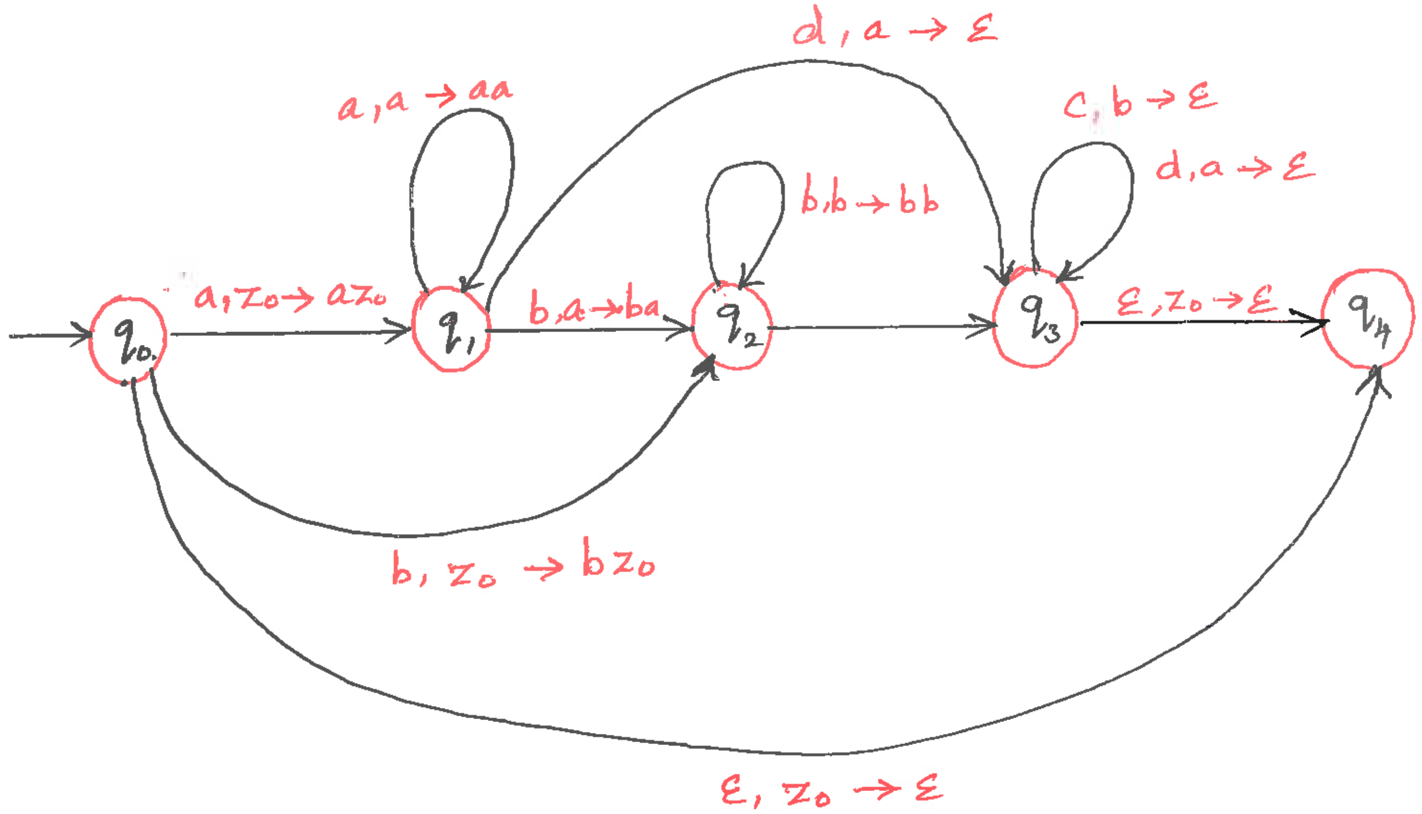
q_3 is Final State



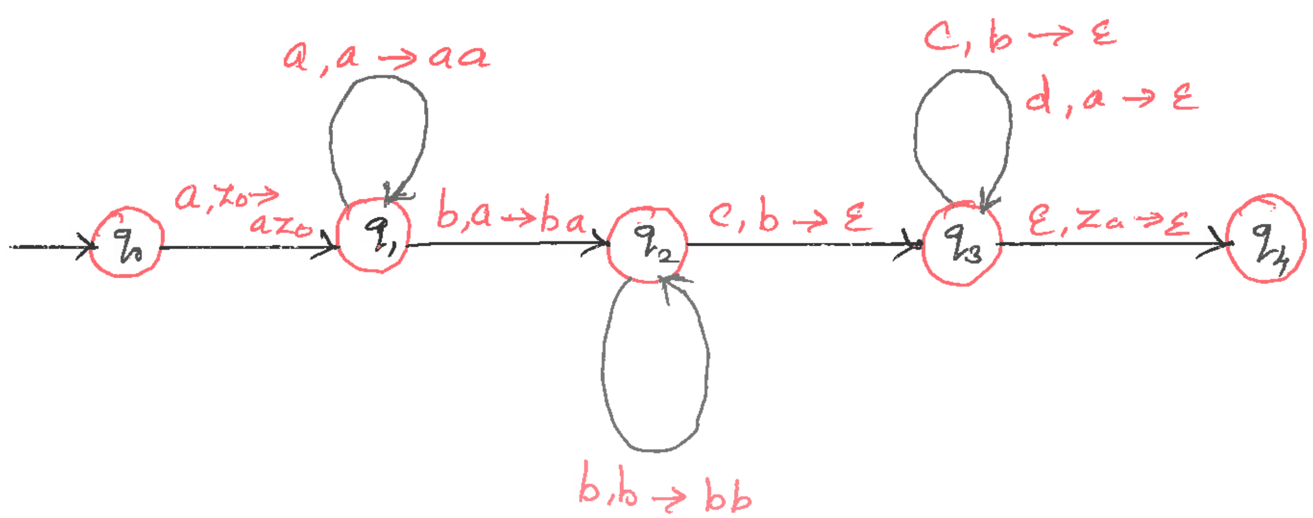
THE LANGUAGE OF A PDA

Design of PDA for the Language $L = \{a^n b^m c^m d^n /$ $m, n \geq 0$ by Empty Stack. And also design it for $m, n \geq 1$ by Empty Stack.

If $m, n \geq 0$.



IF $m, n \geq 1$.



Equivalance of PDA's AND CFG's

PDA → CFG

Let $M = (\{q_0, q_1\}, \{0,1\}, \{X, Z_0\}, \delta, q_0, Z_0, \phi)$

Where δ is given by,

$\delta(q_0, 0, Z_0) = (q_0, XZ_0)$
$\delta(q_0, 0, X) = (q_0, XX)$
$\delta(q_0, 1, X) = (q_1, \varepsilon)$
$\delta(q_1, 1, X) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, X) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$

Construct CFG G generating N(M)

S producations are,

P1 : $S \to [q_0, Z_0\, q_0]$
$\boxed{P2 : S \to [q_0, Z_0, q_1]}$
P3 : Let us take $\delta(q_0, 0\, Z_0) = (q_0, XZ_0)$

Producations are :

P3 : $[q_0, Z_0, q_0] \to 0\,[q_0, X, q_0]\,[q_0, Z_0, q_0]$
P4 : $[q_0, Z_0, q_0] \to 0\,[q_0\, X, q_1]\,[q_1, Z_0, q_0]$
P5 : $[q_0, Z_0, q_1] \to 0\,[q_0, X, q_0]\,[q_0, Z_0, q_1]$
$\boxed{P6 : [q_0, Z_0, q_1] \to 0\,[q_0, X, q_1]\,[q_1, Z_0, q_1]}$

Let us take,

$\delta(q_0, 0, X) = (q_0, XX)$

Producations are

P7 : $[q_0, X\, q_0] \to 0\,[q_0, X, q_0]\,[q_0, X, q_0]$
P8 : $[q_0, X, q_0] \to 0\,[q_0\, X, q_0]\,[q_1, X, q_0]$
P9 : $[q_0, X, q_1] \to 0\,[q_0, X, q_0]\,[q_0, X, q_1]$
$\boxed{P_{10} : [q_0, X, q_1] \to 0\,[q_0, X, q_1]\,[q_1, X, q_1]}$

P_{11} : $[q_0, X, q_1] \to \varepsilon$ $\quad \delta(q_0, 1, X) = (q_1, \varepsilon)$

$\boxed{P_{12} : [q_1, Z_0, q_1] \to \varepsilon}$ for $\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$

$\boxed{P_{13} : [q_1, X, q_1] \to \varepsilon}$ for $\delta(q_1, \varepsilon, X) = (q_1, \varepsilon)$

$\boxed{P_{14} : [q_1, X, q_1]} \to$ for

$\delta(q_1, 1, X) = (q_1, \varepsilon)$

P_2, P_6, P_{10}, P_{12}, P_{13}, P_{14} are the only producations that can able to produce the terminals. So we have to delete the other producations

CFG → PDA

Construct the given Expression to a PDA

$E \to I \mid E * E \mid E + E \mid (E)$
$I \to a \mid b \mid I_a \mid I_b \mid I_0 \mid I_1$

PDA is given by.

$M = (\{q\}, \{+, *, a, b, (,), 0, 1\},$
$\{I, E, +, *, a, b, (,), 0, 1\}$
$\delta, q, E)$

Where δ is defined by,

(i) $\delta(q, \varepsilon, I) = \{(q, a), (a, b), (q, I_a), (q, I_b), (q, I_0), (q, I_1)\}$

(ii) $\delta(q, \varepsilon, E) = \{(q, I), (q, E+E), (q, E*E), (q, (E))\}$

(iii)

$\delta(q, a, a) = (q, \varepsilon)$ $\quad \delta(q, (, () = (q, \varepsilon)$
$\delta(q, b, b) = (q, \varepsilon)$ $\quad \delta(q,),)) = (q, \varepsilon)$
$\delta(q, 0, 0) = (q, \varepsilon)$ $\quad \delta(q, +, +) = (q, \varepsilon)$
$\delta(q, 1, 1) = (q, \varepsilon)$ $\quad \delta(q, *, *) = (q, \varepsilon)$

Programming Techniques for Turing Machine

1. storage in Finite control
2. Multiple Track
3. checking off symbols
4. Subroutine.

storage in Finite Control

TM has finite Control

It store the following Information

1. Current state
2. Current Symbol

$$\delta(\underbrace{q_0}_{\text{current state}}, \underbrace{a}_{\text{Input}}) = (\underbrace{q_1}_{\text{next state}}, \underbrace{b}_{\text{Modified Symbol}}, \underbrace{R}_{\text{Movement Direction}})$$

So. Tm can recognise the language $0^n, n$

2. Multiple Tracks

Using multiple track, we can perform subtraction

#	1	1	1	1	1	#
B	B	B	B	1	1	B
B	1	1	1	B	B	B

$\boxed{11111 - 11 = 111}$

checking off symbols

* Tm can recognize any type of string using off symbols.
* Using off symbols, it decide the direction left or right.

→ Tm can recognize the $L = \{wcw\}$

$w = aba\ \underbrace{\varepsilon}_{\text{off symbols}}\ aba$

The Tm decide the direction depend upon the 'ε' off symbol.

Subroutine

* We can write subroutine as a TM
* We can construct subroutine for the following

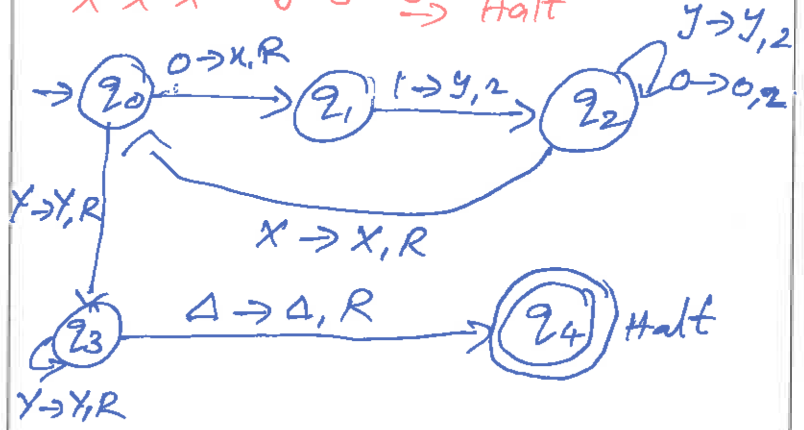
$f(a,b) = a+b.$

Turing Machine [TM]

Design Tm to recognize $L = 0^n 1^n$

n=3

0 0 0 1 1 1
X 0 0 1 1 1
X 0 0 Y
X X 0 Y Y
X X X Y Y Y Halt



Transition configuration

1. $\delta(q_0, 0) = (q_1, X, R)$
2. $\delta(q_1, 0) = (q_1, 0, R)$
3. $\delta(q_1, 1) = (q_2, Y, L)$
4. $\delta(q_2, 0) = (q_2, 0, L)$
5. $\delta(q_2, X) = (q_0, X, R)$
6. $\delta(q_0, Y) = (q_3, Y, R)$
7. $\delta(q_3, \Delta) = (q_4, \Delta, R)$

Comparision of FA, PDA and TM

Finite Automata

1. It recognizes Regular language
2. The i/p Tape is of finite length
3. one direction movement

PDA

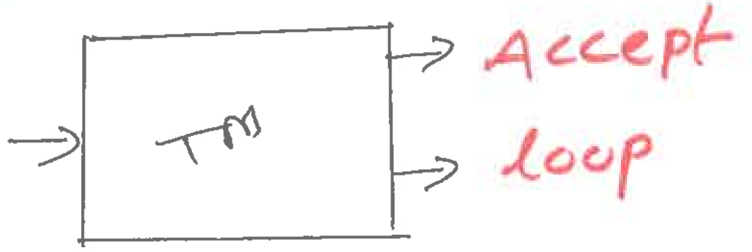
1. It will recognize CFL, RL
2. stack memory used
3. Push & Pop operation.

TM

1. It recognize ALL language
2. Infinite length taped is used.
3. Head can move in both directions.

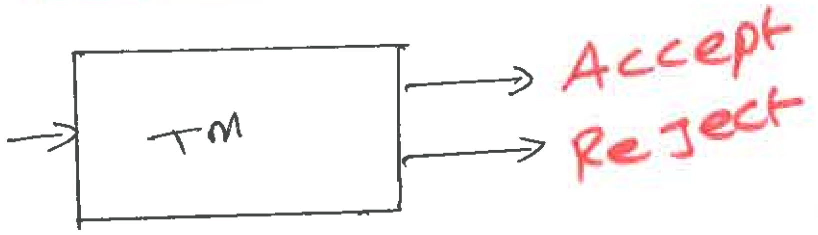
Undecidability

Recursive Enumerable language



* R.E language can be accepted or recognize by TM.
* TM will not enter into rejecting state, it means TM can loop forever.

Recursive language



The recursive language can be decided by TM, which means it will enter into accept state or reject state for the string of language.

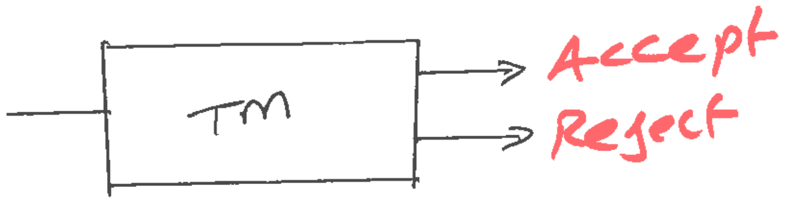
Undecidable problem

$Halt_{TM}$ is undecidable

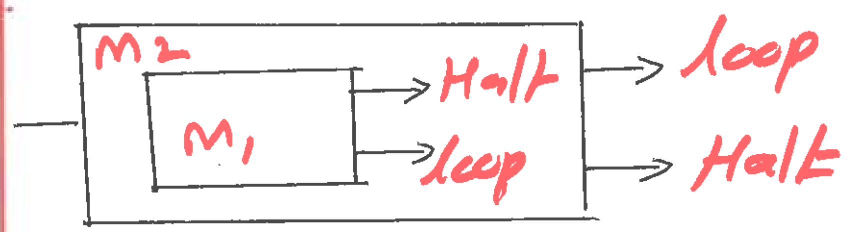
we can prove that $Halt_{TM}$ is undecidable.

Basic idea

if ($Halt_{TM}$ is decidable)
{
Decidable language is also Turing acceptable
}



But Halting problem is undecidable.



[If M_1 Halt Then
M_2 will loop
If M_1 in loop Then
M_2 will Halt]

* It is contradiction
* So $Halt_{TM}$ is undecidable.

* Prove that the diagonalization language Ld is not Recursively Enumerable Ld.

creation of Diagonalization lang.

0	1	1	0
0	1	0	0
1	1	1	0
1	0	1	1

Diagonal value = 0111

complement = 1000

[If w is in Ld Then
TM will accept 'w'
But By the definition of Ld
$w \notin Ld$]

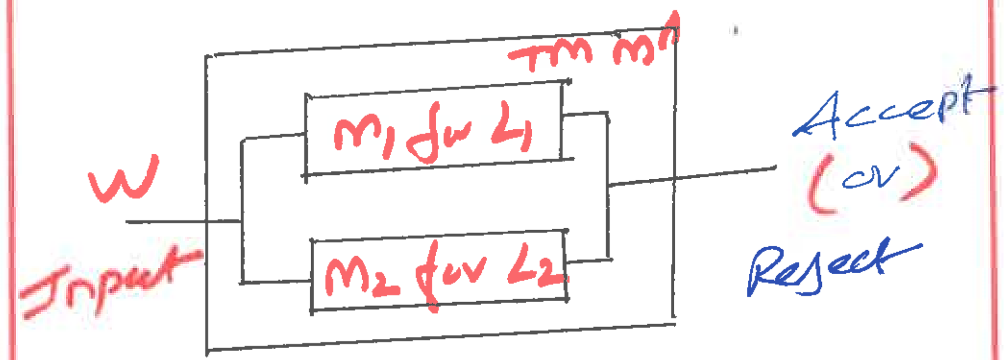
* Thus Ld is not recursively Enumerable language.

* Mathematical Preliminaries

Recursive & Recursively Enumerable languages

* **Theorem**

If L_1 & L_2 are recursive language then $L_1 \cup L_2$ also recursive language.



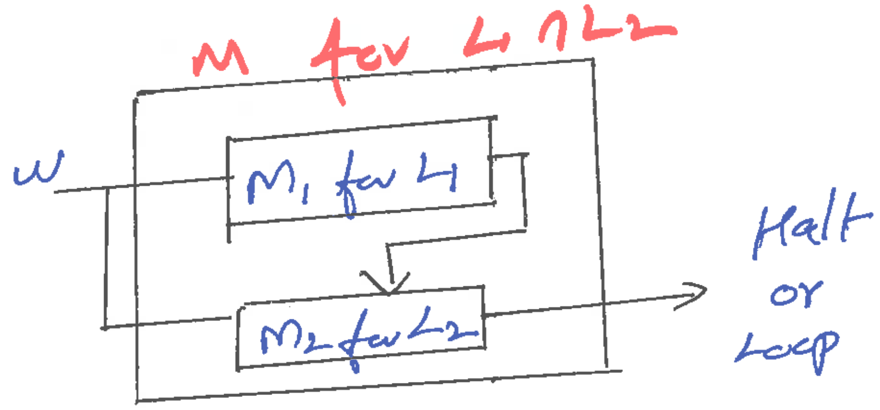
[If $w \in (L_1 \cup L_2)$ Then M_1 Halt or M_2 Halt.
If $w \notin (L_1 \cup L_2)$ Then M_2 & M_2 will not Halt]

* Hence L_1 & L_2 are recursive then $L_1 \cup L_2$ also recursive language.

* we conclude that m' behaves for the language of $[L_1 \cup L_2]$.

Theorem

If two language L_1 and L_2 are recursively Enumerable then their intersection $L_1 \cap L_2$ also recursive Enumerable.



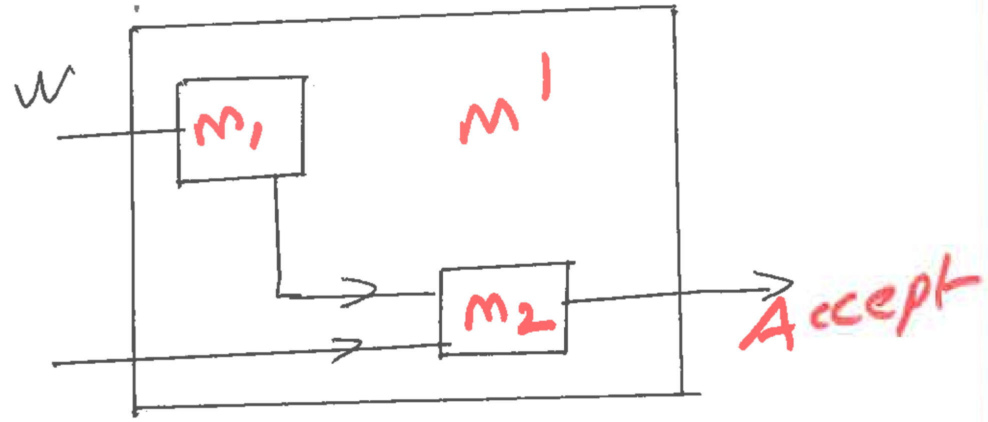
[If $w \in L_1$ & $w \in L_2$ Then $w \in L_1 \cap L_2$
If $w \in L_1$ [or] $w \in L_2$ then $w \notin L_1 \cap L_2$]

* thus $L_1 \cap L_2$ is Recursively Enumerable language

* we conclude that M behaves for the language of $L_1 \cap L_2$.

Rice Theorem

Every non trival property of Recursively Enumerable language is undecidable.



1. [If $w \in m_1$ [or] $w \notin m_1$]
2. [If M_1 accept w Then m' accept the language of m_2]

* By the condition of 1 & 2 we can not decide code for m'.

* This proves that the property of Recursively Enumerable is undecidable.

UNDECIDABILITY-3

Enumerating Binary Code

Obtain the code for $\langle M, 1011 \rangle$ where

$M = (\{q_1, q_2, q_3\}, \{0,1\}, \{0,1,B\}, \delta,$
$q_1, B, \{q_2\})$ has the moves

$\delta(q_1, 1) = (q_3, 0, R)$

$\delta(q_3, 0) = (q_1, 1, R)$

$\delta(q_3, 1) = (q_2, 0, R)$

$\delta(q_3, B) = (q_3, 1, L)$

Consider the following replacements

q_1 by one zero
q_2 by two zeros
q_3 by 3 zeros

stats

left by One zero
right by 2 zeros

Directions

0 by one zero
1 by 2 zeros
B by 3 zeros

Tape Symbols

code for $\langle M, 1011 \rangle$ is

$111 \underbrace{0100100010100}_{\text{Code-1}} 11 \underbrace{000101010010 0}_{\text{Code-2}}$

$11 \underbrace{00010010010100}_{\text{Code-3}} 11 \underbrace{0001000100010 0010}_{\text{Code-4}}$

$111 \underbrace{1011}_{w}$

Post's Correspondence Problem

Let $\Sigma = \{0,1\}$

Let A and B be strings. Find the instance of Post's correspondence problem

i	List A w_i	List B x_i
1	1	111
2	10111	10
3	10	0

Let $M = A$,

$i_1 = 2, \; i_2 = 1, \; i_3 = 1, \; i_4 = 3$

Take this combination 2113

By concatinating strings in this Series

$w_2 w_1 w_1 w_3 = x_2 x_1 x_1 x_3$

$\underline{10111}\;\underline{1}\;\underline{1}\;\underline{10} = \underline{10}\;\underline{111}\;\underline{111}\;\underline{0}$

$\therefore$ Instance of PCP is 2113

For another instance 2113 2113 of a PCP has a solution.

SIMATS
SCHOOL OF ENGINEERING

Engineer to Excel

Approved by AICTE | IET-UK Accreditation

Saveetha Nagar, Thandalam, Chennai - 602 105, TamilNadu, India