Finite automata to Regular Expression Conversion using ARDEN'S THEOREM

Let P, Q and R be three regular expressions over Σ. If P does not contain $\wedge$, then eqn

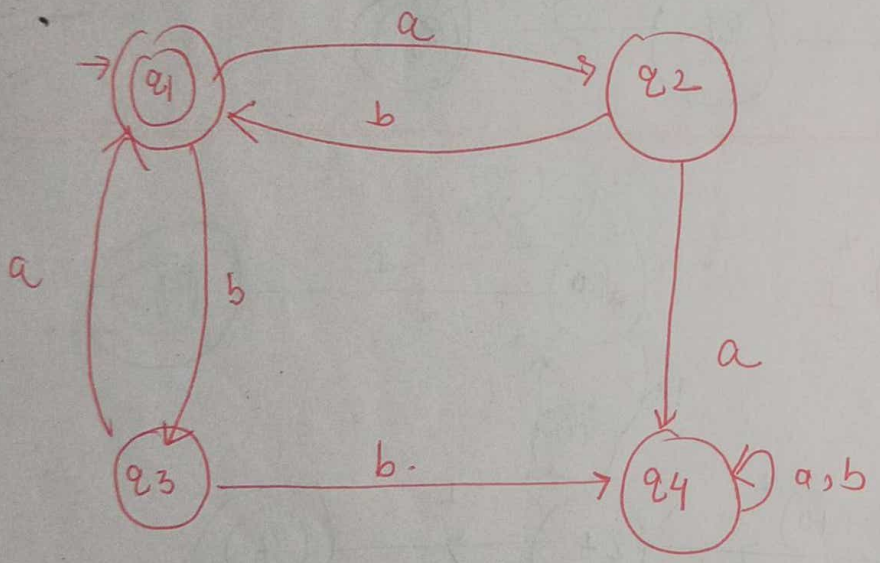
$$R = Q + RP$$

has a unique soln given by $R = QP^*$.

$$Q + RP = Q + (QP^*)P = Q(\wedge + P^*P)$$

$$= QP^*.$$

Q: Convert this into regular expression.



We can apply the above method directly since graph does not contain any null move & there is only one initial state

$$q_1 = q_2 b + q_3 a + \wedge$$

$$q_2 = q_1 a$$

$$q_3 = q_1 b$$

$$q_4 = q_4 a + q_4 b + q_2 a + q_3 b$$

$q_1 = q_1 ab + q_1 ba + \wedge$

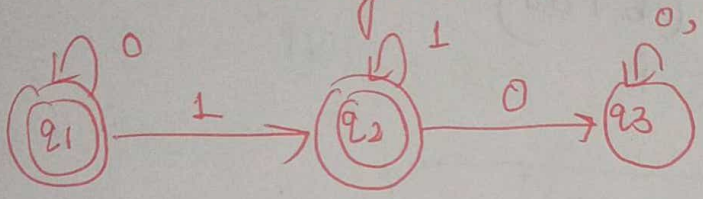
$q_1 = \underbrace{q_1(ab+ba)}_{R\ \ \ \ \ \ Q} + \underbrace{\wedge}_{P}$ $\quad R = Q + RP$

$= QP^*$

$= \wedge(ab+ba)^*$

$= (ab+ba)^*$

Q2 Describe in english the set accepted by FA whose transition diagram is



Soln

$q_1 = q_1 0 + \wedge \Rightarrow \wedge 0^* = 0^*$

$q_2 = q_1 1 + q_2 1$

$q_3 = q_2 0 + q_3 0 + q_3 1$

$q_2 = 0^* 1 + q_2 1$

~~$q_2 = q_2(1 + 0^* 1)$~~ $\quad R = Q + RP$

~~$R = P + QR$~~

$q_2 = q_2 1 + 0^* 1 \quad = QP^*$

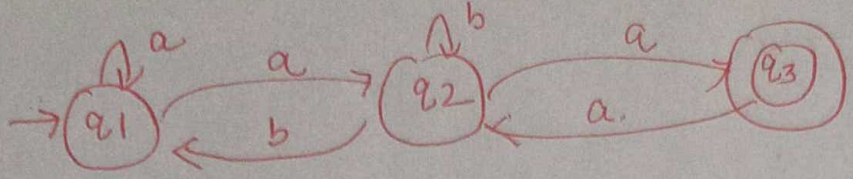
$q_2 \quad (0^* 1) 1^*$

$\Rightarrow q_1 \& q_2$

$q_1 + q_2 = 0^* + 0^* 1 1^*$

$= 0^*(\wedge + 11^*) = 0^*(1 1^*)$

$= 0^* 1^*$



$q_1 = q_1a + q_2b + \wedge$ — (1)

$q_2 = q_2b + q_3a + q_1a$ — (2)

$q_3 = q_2a$ — (3)

$q_2 = q_2b + q_2aa + q_1a$

$R = Q + RP$.

QP^*

$\underset{R}{q_2} = \underset{Q}{q_1a} + \underset{R}{q_2}\underset{P}{(b+aa)}$

$\boxed{q_2 = q_1a(b+aa)^*}$

Substituting q_2 in q_1

$R = Q + RP$

$\boxed{QP^*}$

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

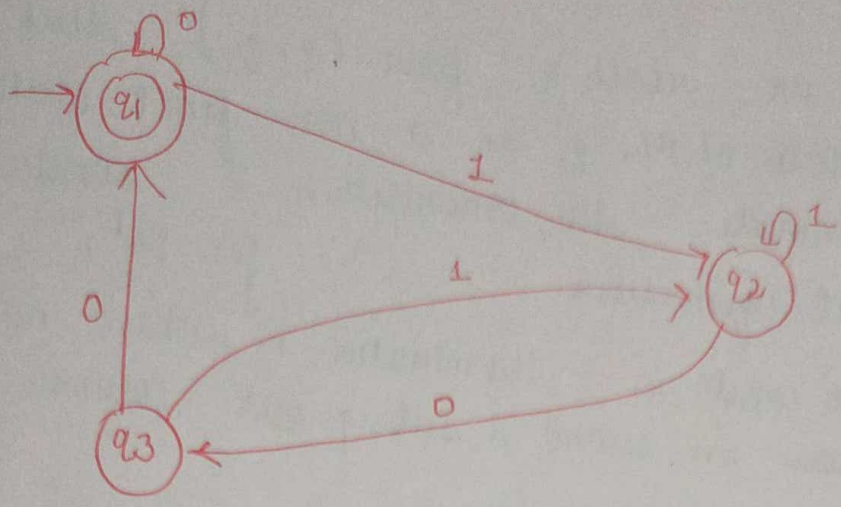
$q_1 = q_1(a + a(b+aa)^*b) + \wedge$

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

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

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

Construct a Regular expression corresponding to state diagram



$q_1 = q_1 0 + q_3 0 + \wedge$ — (1)

$q_2 = q_2 1 + q_3 1 + q_1 1$ — (2)

$q_3 = q_2 0$ — (3)

$q_2 = q_2 1 + q_1 1 + (q_2 0) 1$

$q_2 = q_1 1 + q_2 (1 + 01)$

$q_2 = q_2 (1+01) + q_1 1$

$$q_2 = q_1 1 (1+01)^*$$

$$q_3 = q_1 1 (1+01)^* 0$$

$q_1 = q_1 0 + q_1 1 (1+01)^* 0\ 0 + \wedge$

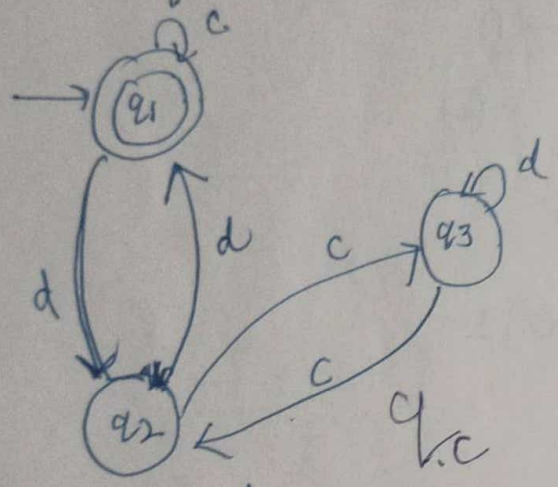
$q_1 = q_1 (0 + 1 (1+01)^* 00) + \wedge$

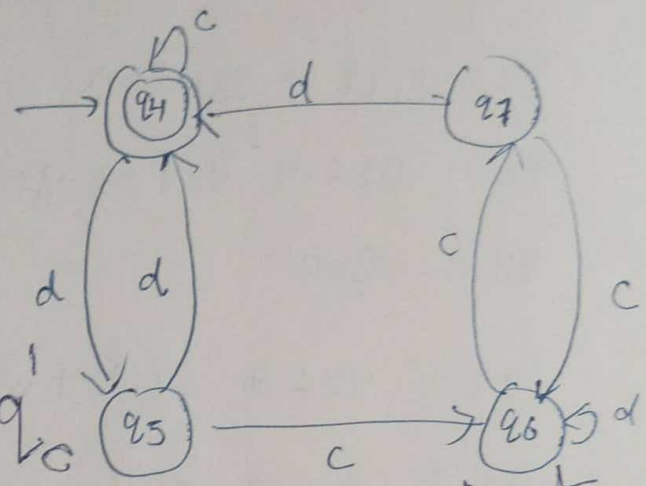
$= (0 + 1(1+01)^* 00)^*$

Equivalence of two finite automata

Case1 If we reach a pair (q, q') such that q is final state of M, q' is a non-final state of M'. we terminate the construction & conclude that M, M' are not equivalent

Case2 Here constn is terminated when no new element appears in second & subsequent columns which are not in first column.





All equivalent

q, q'	$qc, q'c$	$qd, q'd$
(q_1, q_4)	(q_1, q_4) ✓	(q_2, q_5) ✗
(q_2, q_5)	(q_3, q_6)	(q_1, q_4) ✓
(q_3, q_6)	(q_2, q_7)	(q_3, q_6)
(q_2, q_7)	(q_3, q_6)	(q_1, q_4)

q

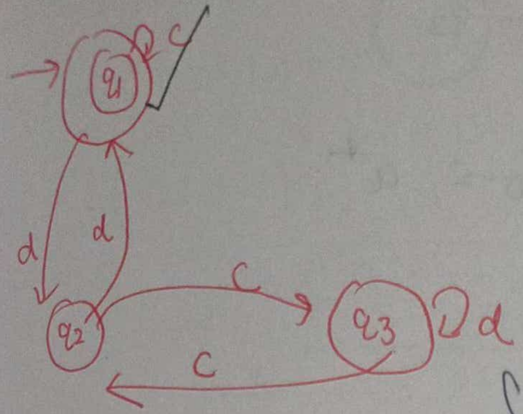
1 NFX

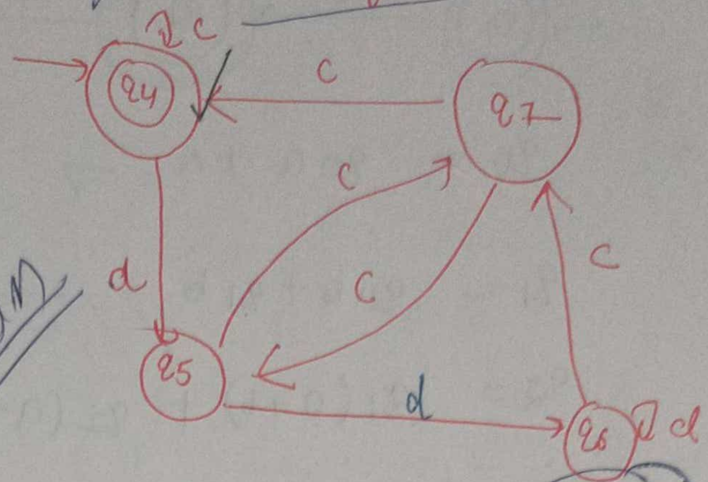
F F ✓

N NF ✓

We donot get a pair (q, q') where q is a final state, & q' is a non-final state.

show that automata M_1 & M_2 are equivalent.





cond^n

(q1, q4)

(q, q')	q_c, q_c'	q_d, q_d'
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_7)	(q_1, q_6)

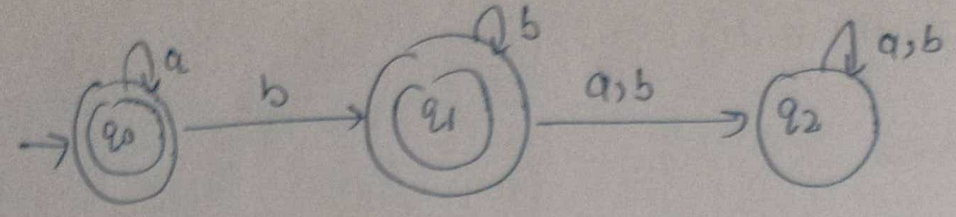
(q_1, q_6) → q_1 is final but q_6 is non final.
F NF

M & M' are not equivalent

2 or 3 questions

FA: Advantages Application
& Limitations M/ivs — out class
Third Unit — Two first Units

Q)



$q_0 = q_0 a + \wedge \Rightarrow q_0 = a^*$

$q_1 = q_0 b + q_1 b \longrightarrow$

$q_2 = q_1(a+b) + q_2(a+b)$

$q_1 = a^* b + q_1 b$

$q_1 = q_1 b + a^* b$

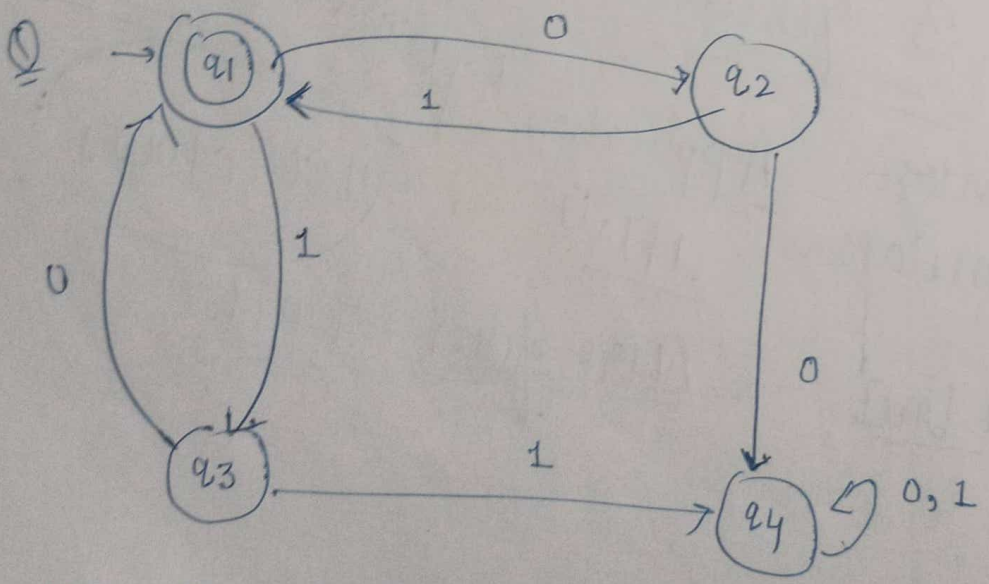
$= (a^* b) b^*$

$q_0 + q_1 = a^* + a^* b b^*$

$= a^* (\wedge + b b^*)$

$= a^* b^* \quad (\varepsilon + RR^* = R^*)$

Q)



Soln $q_1 = q_2 1 + q_3 0 + \wedge \quad -(1)$

$q_2 = q_1 0 \quad -(2)$

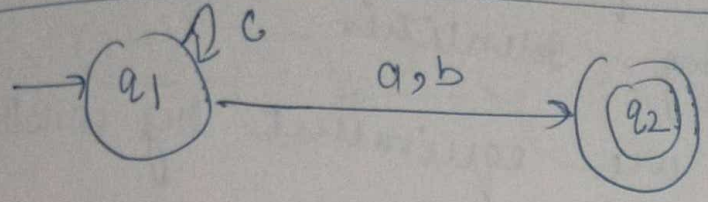
$q_3 = q_1 1 \quad -(3)$

$q_4 = q_3 1 + q_2 0 + q_4(0+1) \quad -(4)$

$q_1 = q_1 01 + q_1 10 + \wedge$

$q_1 = q_1 (01 + 10) + \wedge$

$q_1 = (01 + 10)^*$



$q_1 = q_1 c$ — (1) $\Rightarrow$ $q_1 = c^* \wedge \Rightarrow c^*$

$q_2 = q_1 (a+b)$

$q_2 = c^* (a+b)$

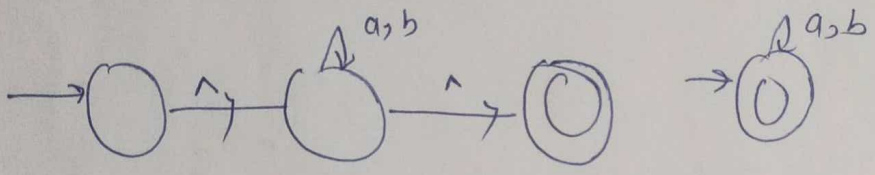
Equivalence of two regular expressions

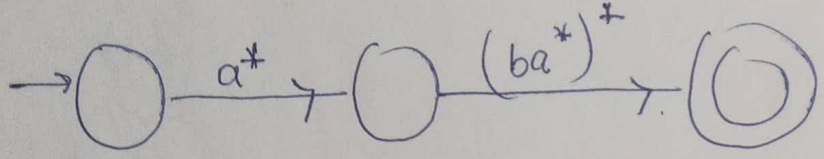
A & B → are equivalent if & only if they represent same set or if there corresponding finite automata are equivalent or we make them equal by using identities.

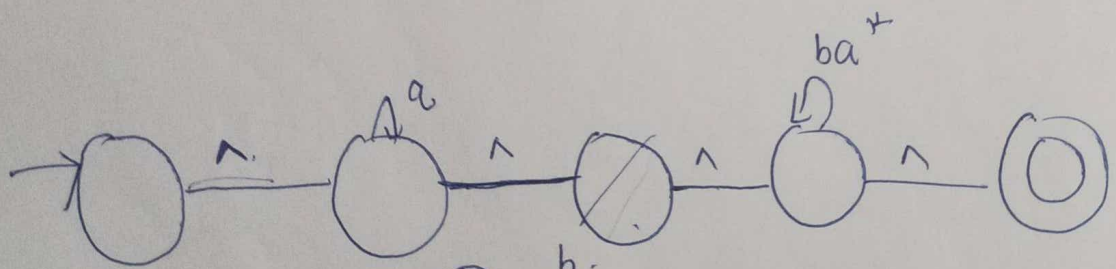
Method 1. A & B are equivalent by constructing finite automata's of A & B.

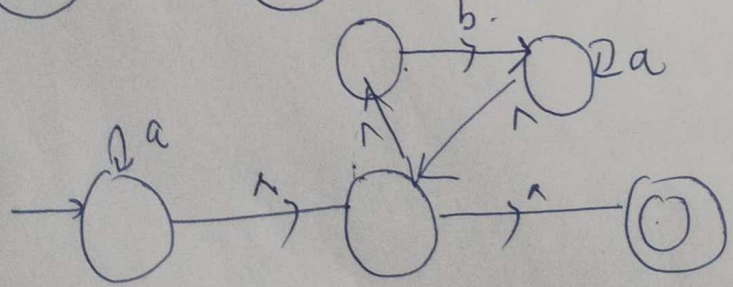
Method 2 equivalence of A & B can be proved with the help of identities.

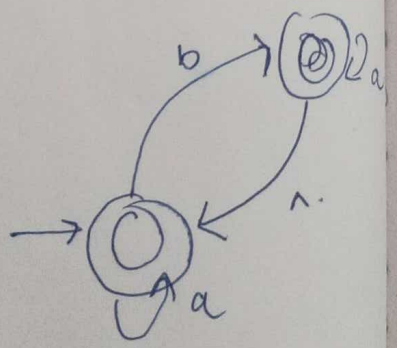
Example :- $(a+b)^* = a^*(ba^*)^*$

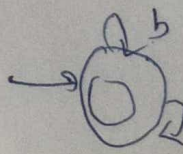






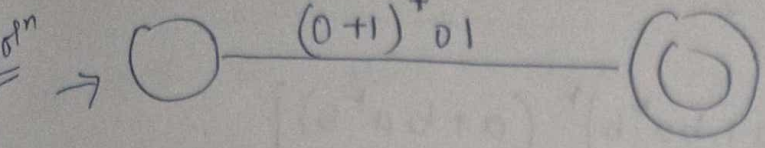


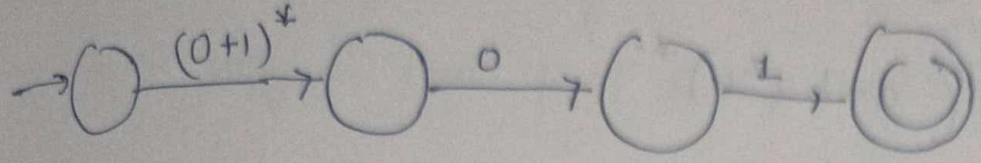


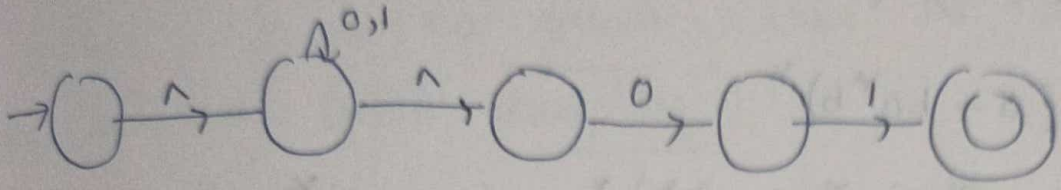


Q $(0+1)^*01 = 0^*(10^*)^*01$

Soln







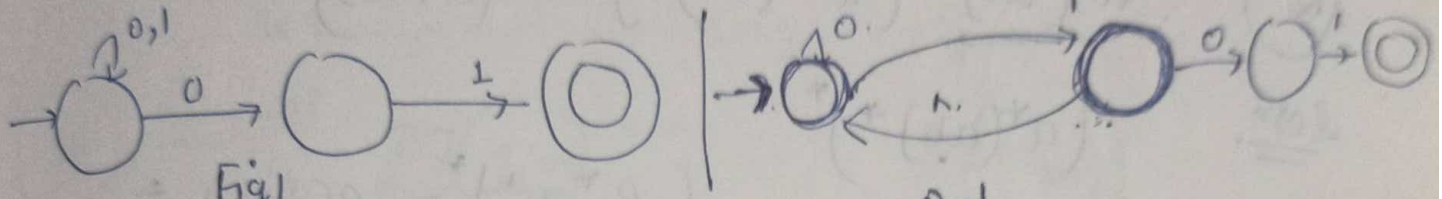
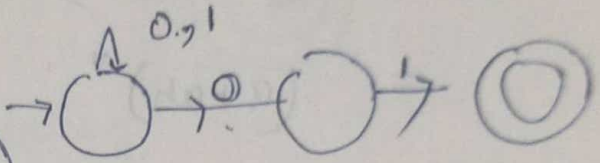
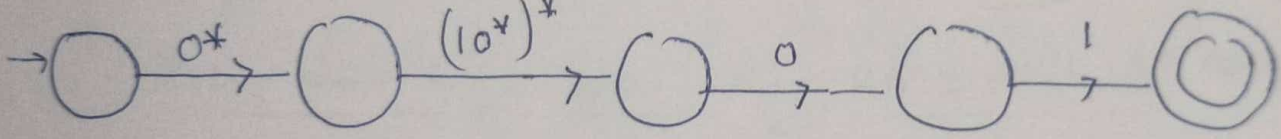
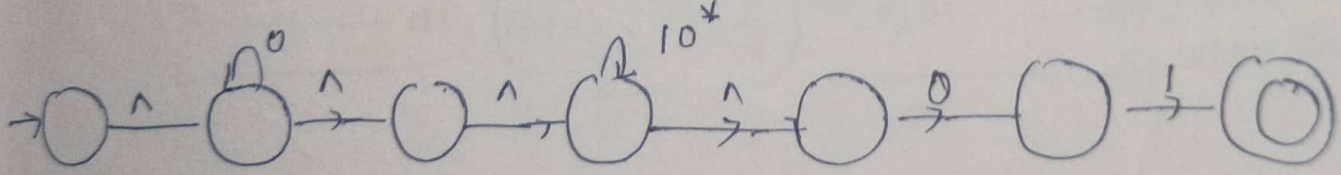


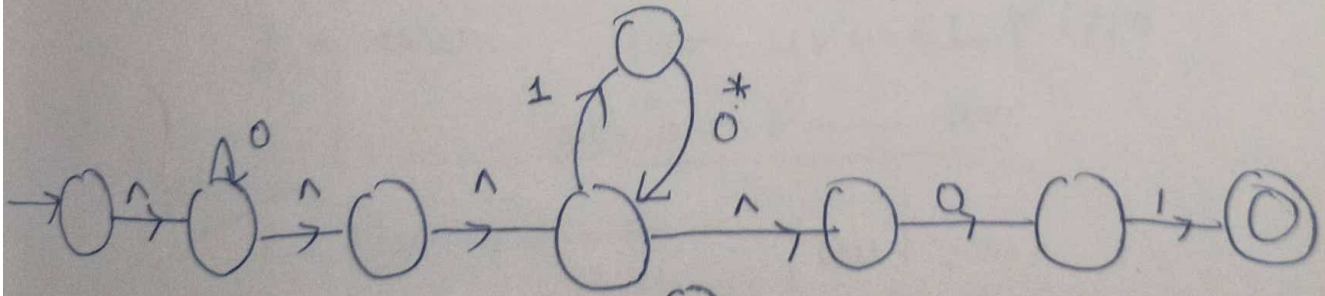
Fig1

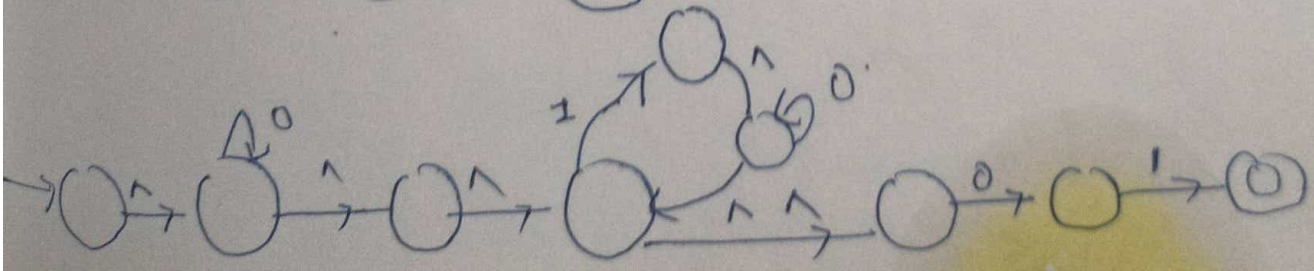
(2) $0^*(10^*)^*01$











Ex $(b+aa^*b)+(b+aa^*b)(a+ba^*b)^*(a+ba^*b)$

$= a^*b(a+ba^*b)^*$

Soln $(b+aa^*b)\left[\wedge + (a+ba^*b)^*(a+ba^*b)\right]$

$= (b+aa^*b)(a+ba^*b)^*$

$= b(\wedge + aa^*)(a+ba^*b)^*$

$= a^*b(a+ba^*b)^*$

(2) $a^*(ab)^*(a^*(ab)^*)^* = (a+ab)^*$

Soln $(a^*(ab)^*)^*$

$= (a+ab)^*$, $(R_1^*R_2^*)^* = R(+R_2)^*$

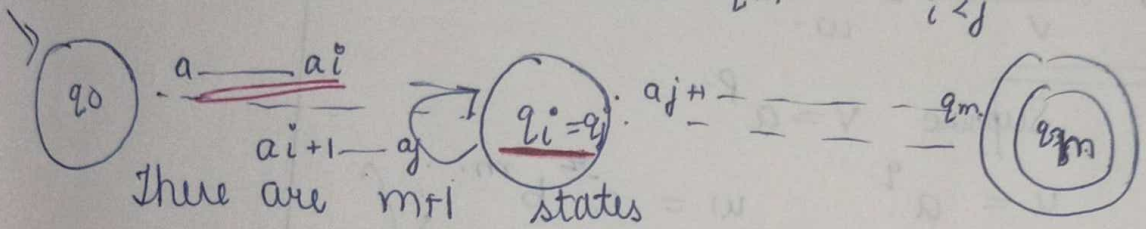
Pumping Lemma

Let L be a regular set. Then there is a constant n such that if z is any word in L, and $|z| \geq n$ we may write $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$ for all $i \geq 0$ $uv^i w$ is in L

n is no greater than the no. of states of the smallest DFSA accepting L.

$z = a_1 - - - - a_m$ $\quad m \geq n$.

$q_0\ q_1 - - - - - q_m$. $\quad i < j$



There are $m+1$ states

Suppose DFSA has n states.

Pigeonhole principle — two of them will be equal.

$a_1 - - - a_i$, $a_{j+1} - - - a_n$ will be accepted

$a_1 - - - a_i$, $(a_{i+1} - - - a_j)^i$ $(a_{j+1} - - - a_m)$ will be accepted

u $\quad v$ $\quad w$

$z = uvw$. $\quad uv^i w \in L \ \forall\ i \geq 0$

$a_1\ a_2 - - - a_n\ a_{n+1} - - - a_m$

$q_0\ q_2 - - q_i q_j - - q_n$ $\quad |uv| \leq n$.

$uv^i w \in L$

This is useful in showing that certain sets are not regular.

$\{a^n b^n \mid n \geq 1\}$ // Don't follow this method

Soln Suppose L is regular.

then L will be accepted by FSA

By Pumping Lemma there is n such that if $z \in L$ and $|z| > n$

$z = uvw$ such that $uv^i w \in L$

$a^m b^m \quad m > n.$

aaa ... a bbb ... b.

u v w.

Suppose $v = a^p$.

$u = a^q \qquad w = a^r b^m.$

$a^r \qquad p + q + r = m$

$a^q (a^p) a^r b^m$

$= a^q (a^p)^i a^r b^m \in L \quad \forall i \geq 0.$

$\therefore$ L is not regular

Q2 $L = \{a^{n^2} \mid n \geq 1\}$

a^{n^2}

uv a

Let L be accepted by DFSA with n states

$|uv| \leq n \qquad |v| \geq 1$

$1 \leq |v| \leq n.$

$|uv^2w| = |uvw| + |v|$

$\leq n^2 + n$

$\{a, a^4, a^9 \ldots a^{n^2}, a^{(n+1)^2} \ldots$

L is regular ⇒ pumping lemma holds

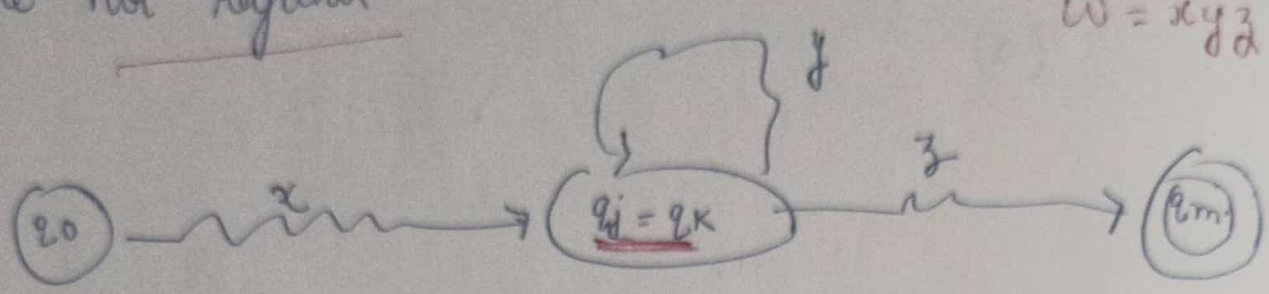
$L = \{ a^p b^{m^2} \mid p \geq 1, m \geq 1 \} \cup \{ b^q \mid q \geq 1 \}$

n.

a - - - b

(a)

Pumping Lemma It gives a method of pumping many input strings from a given string. It is used to show that certain sets are not regular.



Let $w = a_1 a_2 \_\_ a_m$ $m \geq n$.

↓ no. of states

$\delta(q_0, a_1 a_2 \_\_ a_i) = q_i$ for $i = 1, 2, \_\_ m$.

Let q_j & q_k (two coincide states).

Then, j & k satisfy the condn $0 \leq j < k \leq n$.

Decompose w into three substrings

$x = a_1 a_2 \_\_\_ a_j$

$y = a_{j+1} \_\_ \_\_ a_k$

$z = a_{k+1} \_\_ \_\_ a_m$.

$q_i \_ a_j \; a_{j+1} \; a_k \; a_{k+1} \; a_m$

$x \quad y \quad z$

$|xy| \leq n$

as $k \leq n$, $|xy| \leq n$

and $w = xyz$.

The automaton M starts from initial state q_0.

On applying string x, it reaches $q_j (= q_k)$.

On applying string y, it comes back to $q_j = q_k$

So, after application of y^i for each $i \geq 0$, the automaton is in same state q_j.

On applying z, it reaches q_m, a final state

Hence $xy^iz \in L$.

To prove that wilain sets are not regular

Step1 Assume that L is regular. Let n be no of states in corresponding finite automaton

Step2 Choose a string w such that $|w| \geq n$.

Use pumping lemma to write

$w = xyz$, with $|xy| \leq n$ & $|y| > 0$.

Step3 Find a suitable integer i such that $xy^iz \notin L$

This contradicts our assumption. Hence L is not regular

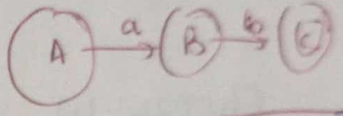
8 Show that $L = \{ a^n b^n \mid n \geq 1 \}$ is not regular

soln Step1 Suppose L is regular. Let n be the no of states in finite automaton accepting L

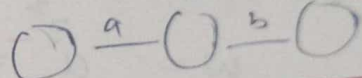
Step2 Let $w = a^n b^n$.

Then put $n = 1$

$w = ab$.



No of states $n = 3$.



$|w| \leq n$

Choose w such that $|w| \geq n$

$|w| \geq n$

$n = 3$.

$w = aabb$

$|w| = 4$

xy^iz

$w = xyz$

$= aabb$

$|xy| \leq n$

$w = xyz$

$= aabb$

$x = a$ $z = bb$

$y = a$

$xy^iz = aa^ibb$

$= abb$

$x = a$ $z = bb$.

$y = a$

$|xy| \leq n$

$|2| \leq 3$.

Step3 $w = xy^iz$ Put $i = 0$

$xz = abb$ ⇒ which is not of form $a^n b^n$

Put $i = 2$

$xy^i z \Rightarrow xyyz$

aaabb

$xy^2z \notin L$.

$\therefore$ L is not regular

Q2 Show that $L = \{ a^{i^2} \mid i \geq 1 \}$ is not regular

Soln **Step1** Assume that L is regular. Let n be no. of states in the corresponding finite automaton

Step2 Put $i = 1$ $\quad a^{i^2} = a$

Put $i = 2$ $\quad a^{2^2} = a^4 =$ aaaa

$n = 5$

Choose w such that $|w| \geq n$

Put $i = 3$

$w = a^{3^2} = a^9 =$ aa a a a a a a a

Choose $x = aa$

$y = aa$

$z = a a a a a$

$|xy| = 4 \qquad |xy| \leq 5.$

$4 \leq 5$

Step3 $w = xy^i z \qquad i = 0$

$xz = aa\ a a a a a$

$= a^7$

which is not of form a^{i^2} so, given L is not regular

Q Show that $L = \{a^p \mid p \text{ is prime}\}$ is not regular

Step1 We suppose L is regular

Step2 Let p be a prime no. greater than n.

$p = 3$. $\quad p = 2, 3, 5, 7, 11$.

$a^3 = aaa \Rightarrow \boxed{n = 4}$

Choose w such that $|w| > n$

~~$w = 5$~~. Put $p = 5$

$a^5 \Rightarrow aaaaa$

$\boxed{|w| = 5}$

Step3 Choose $xy^iz \Rightarrow i = 0$

~~$xz =$~~ ~~$xz =$~~

$xy^iz = xz$

$= aaa$

$w = xyz$

$= aaaaa$

$x = a$

$y = aa, \quad z = aa$.

$|xy| \leq n$.

$2 \leq 4$.

Put $y = 1 \quad xyz \Rightarrow aaaaa$.

Put $y = 2 \quad xyyz = aaaaaaa$

$xyyyz = aaaaaaaaa \Rightarrow a^9$.

9 is not prime no.

Q Show that $L = \{ww \mid w \in \{a,b\}^*\}$ is not reg

Soln Let $w = aba$ $n = 4$.

$ww = abaaba$

~~$n = 7$ No of states~~

Choose ~~tut~~ ww

$w = xyz$

$x = ab$. $y = a$ $z = aba$.

$|xy| \leq n$,

$w = xy^i z$

$= ab.aba$ $\Rightarrow$ which is not of form ww

Pumping Lemma

the term pumping lemma is made up of two words one is "pumping" and second is "lemma". The word pumping means to generate many input strings by pushing a symbol in an input string again & again. The word "lemma" refers to intermediate theorem in a proof. Pumping lemma is used to prove that given language is not regular.

A language is regular if

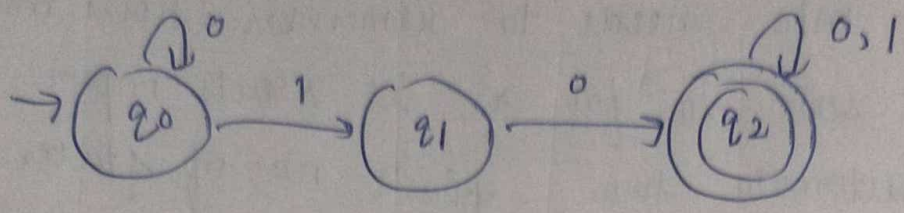
(1) Language is accepted by finite automata

(2) A regular grammar can be constructed to exactly generate strings in language

(3) A regular expression can be

Consider the language $L = \{ a^n b^n \mid n \geq 0 \}$
finite automata has very limited memory. To accept this language, the m/c needs to remember how many a's have been seen so far as it reads input. Because finite automata has finite no. of states but no of a's are unlimited. so, m/c needs to keep track of unlimited no. of possibilities but finite automata can't do so because of its memory.

Finite automata with ε- Moves



we can't move from state q_0 to q_1 without using input symbol.

Finite automata with ∧ moves allows transition from one state to another without consuming a symbol.

∧ move is an extension of finite automata that allows moves on empty input

Formal Defn of ε-NFA

is five tuple $A = (Q, \Sigma, q_0, F, \delta)$ be a non-deterministic finite automata with ε- NFA

Q → finite set of states

S → finite set of input symbols

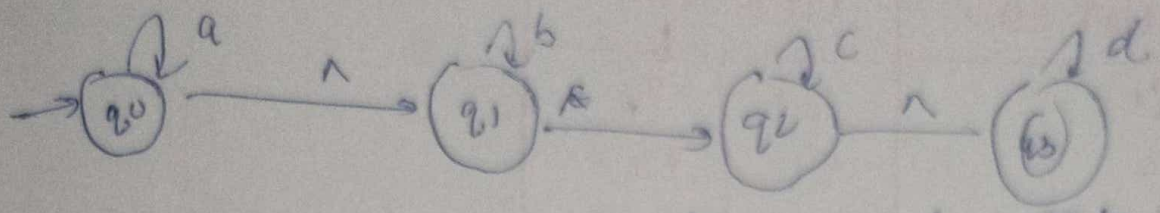
q_0 → initial state

$q_0 \in Q$

F → set of final states

δ is the transition fx which takes as input a state and input symbol & returns set of states o/p.

$Q \times \Sigma^* \rightarrow 2^Q$ (2^Q is power set of Q)



To find ^- closure of state q_0

(1) add q to E

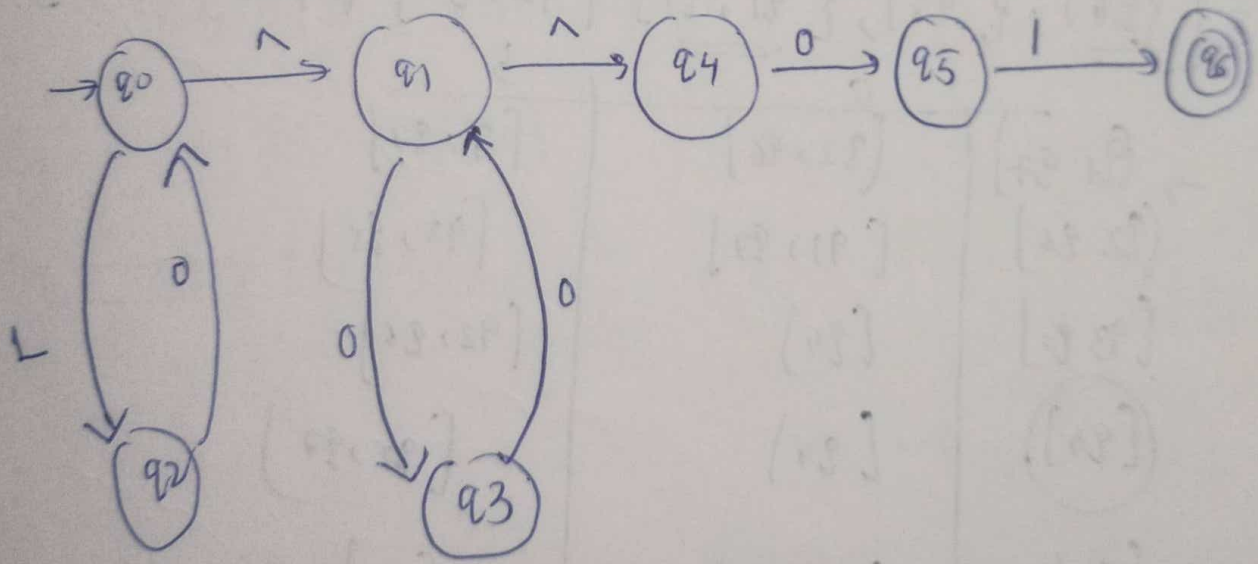
$E = \{ q_0 \}$

(2) find all the states that are reachable from q with label ^

$\delta(q_0, ^) = q_1$

$\delta(q_1, ^) = q_2$

$E = \{ q_0, q_1, q_2, q_3 \}$



Minimization

	Present state	Next state a	Next state b
→	q_1	q_2	q_4
	q_2	q_1	q_3
	q_3	q_4	q_2 X
	q_4	q_4	q_1 X
	q_5	q_4	q_6 X
	q_6	q_1	q_5
	q_7	q_6	q_7
	q_8	q_7	q_4 X

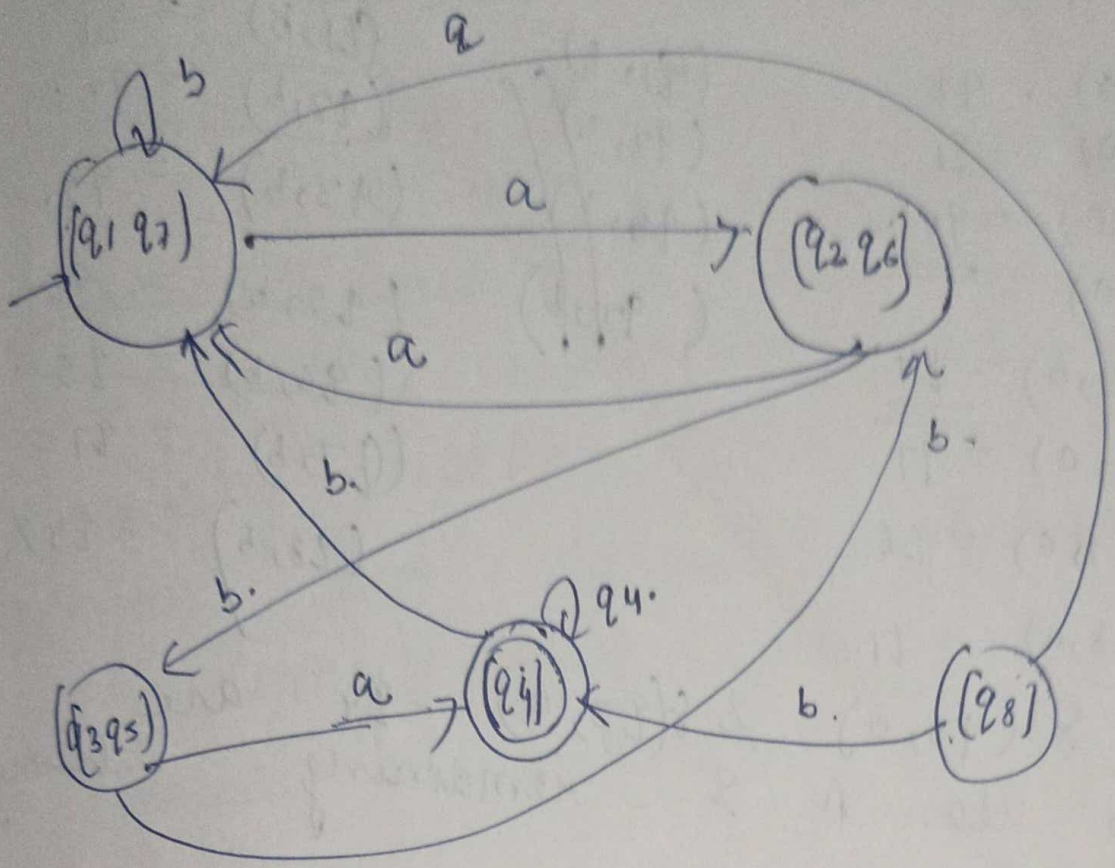
Sol^n $\Pi_0 = \{q_4\}\ \{q_1, q_2, q_3, q_5, q_6, q_7, q_8\}$

$\Pi_1 = \{q_1, q_2, q_6, q_7\}\ \{q_3, q_5\}\ \{q_8\}\ \{q_4\}$

$\Pi_2 = \{\{q_4\}\{q_8\}\ \{q_1, q_7\}\ \{q_2, q_6\}\ \{q_3, q_5\}\ \{q_5\}$

$\Pi_3 = \{\{q_4\}, \{q_8\}, \{q_1, q_7\}\ \{q_2, q_6\}\ \{q_3, q_5\}\ \{q_5\}$

	a	b
→ $[q_1\ q_7]$	$[q_2, q_6]$	$[q_3, q_7]$
$[q_2\ q_6]$	$[q_1, q_7]$	$[q_3, q_5]$
$[q_3\ q_5]$	$[q_4]$	$[q_2, q_6]$
$[q_4]$	$[q_4]$	$[q_1, q_7]$
$[q_8]$	$[q_1, q_7]$	$[q_4]$



$Q_0 = \{q_1, q_7\}$

$(q_1, a) = (q_2, q_6) = (q_7, b) \Rightarrow$

$(q_7, a) = (q_1, q_7) = (q_7, b)$

$Q_1 = \{q_1, q_2, q_7 q_6\}$.

~~$Q_1 = \{q_1, q_3, q_5, q_2,$~~

Q {q4} {q1, q2, q3, q5, q6, q7, q8}
A B

δ(q1,0) = q2 (q1,b) (q1,b) = q1 ✓
δ(q2,0) = q1 (q2,b) (q2,b) = q3 ✓
δ(q3,0) = q4 ✗ (q3,b) (q3,b) = q2 ✓
(q4,0) (q4,b) (q5,b) = q6 ✓
δ(q5,0) = q4 ✗ (q6,b) (q6,b) = q5 ✓
δ(q6,0) = q7 (q7,b) (q7,b) = q7 ✓
δ(q7,0) = q6 (q8,b) (q8,b) = q4 ✗
δ(q8,0) = q7.

δ(q3,a) & δ(q5,a) = q4 are belong to class A & remaining belong to class B.

{q4} {q3, q5} {q1, q2, q6, q7, q8}
A B1

{q4} {q3, q5} {q1, q2, q6, q7} {q8}
A B1 B2 B21

δ(q3,a) = q4 δ(q3,b) = q2
δ(q4,a) = q4 δ(q4,b) = q6

{q4} {q3, q5} {q2, q6} {q1, q7} {q8}

δ(q1,a) = q2 δ(q1,b) = q1
δ(q2,a) = q1 δ(q2,b) = q3
δ(q6,a) = q7 δ(q6,b) = q5
δ(q7,a) = q1 δ(q7,b) = q7

(1) What is Computation (TOC)

step-by-step solution to a given problem. eg multiply two numbers
dictionary & search for a word.
find a word in dictionary.

→ graph reachability problem:-
checking whether there is a path between two vertices in a graph.

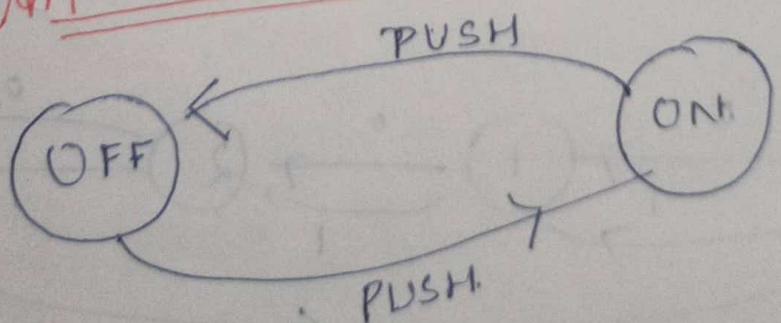
Computational devices used to compute our solution

eg. ✓ calculator (computational device)
cell phone (smart phones)
computer, pen & paper

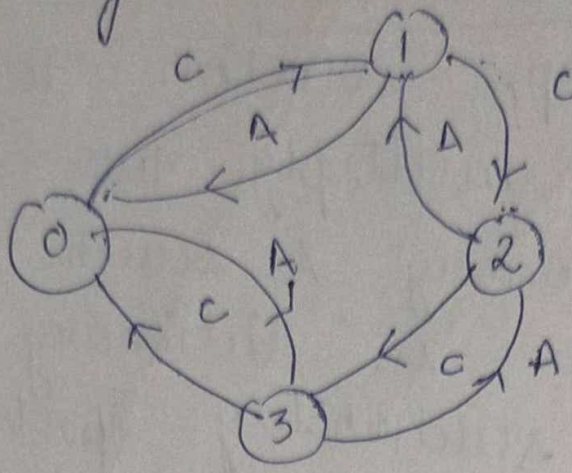
Study computational devices based on resources that use.

Finite Automata →
has finite amount of memory.
Automata is the plural of word - automaton.

eg ✓ An electric switch (on or off)



2) Fan regulator



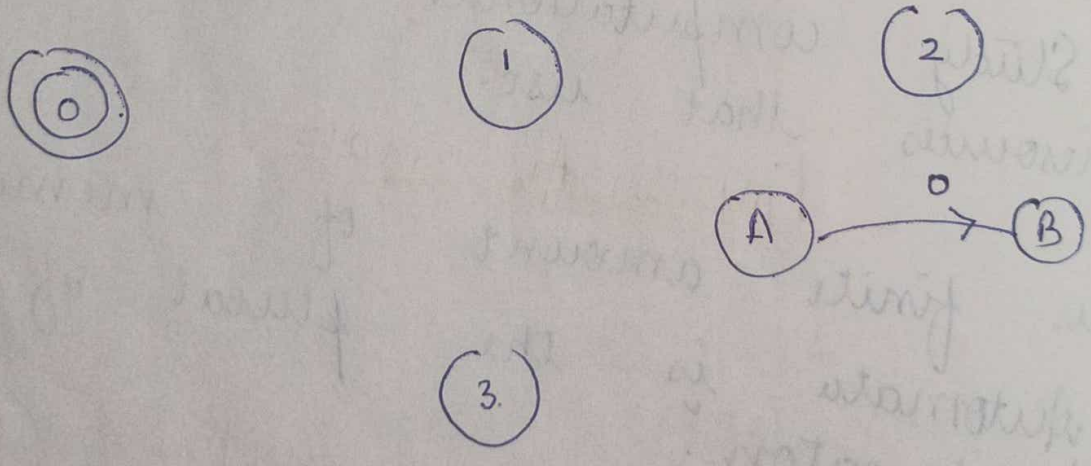
CCACCAC

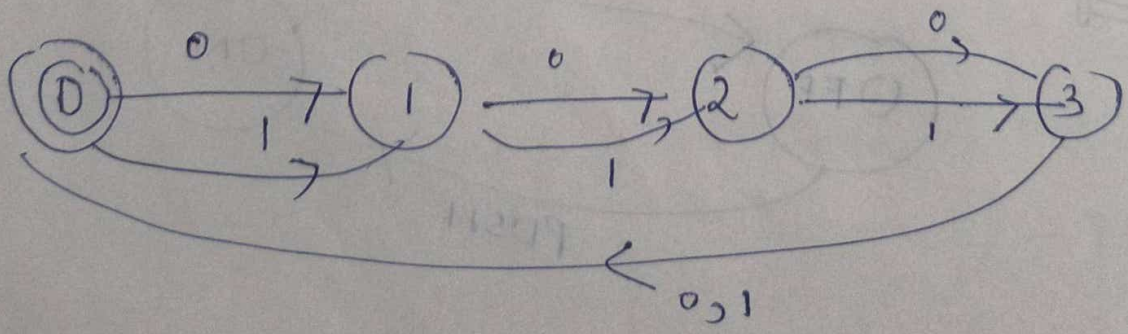
2 operations

4 states

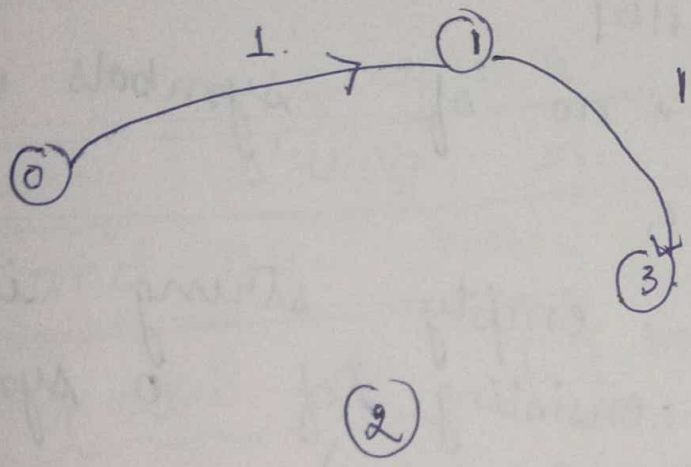
(3) $L = \{x : | x$ is a binary string divisible by 4$\}$

B	Decimal	Belongs to L
100	4	✓
110 (4 2 1)	6	✗
1100 (8 4 2 1)	12	✓





Observe that set of binary no's divisible by 4 are exactly those that have 00 as a suffix



00 01 11 10

001

4 2 1

1 1 0

4 2 1

No divisible by 4 or not

Definitions and Notations

An alphabet is a finite set of symbols. denoted by Σ

$\Sigma = \{0, 1\}$ - alphabet set of binary numbers.

String - A string over an alphabet is a sequence of symbols from alphabet

$w = 0110$

Length of string → no. of symbols in string

$|w| = 5$

The epsilon → empty string is the string consisting of 0 symbols. denoted as ϵ (epsilon)

Kleen star

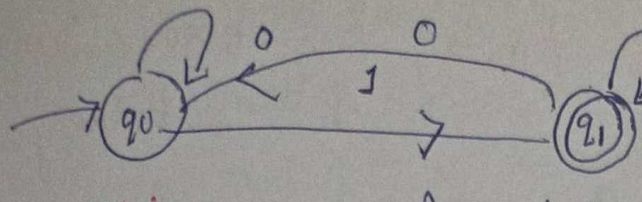
$\Sigma^* = \{ w \mid w$ is a string over Σ and length of $|w| = i \}$

Kleen Σ^+ $\bigcup_{i>0} \Sigma^i$

Language → A language L over an alphabet Σ, is a subset of Σ^*.

$L = \{ w \mid w$ ~~was~~ ends with a $1 \}$

$L = \{ 1, 01, 011, 101 \text{---} \}$



→ final state

Definition :- A deterministic finite automaton is a 5 tuple

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

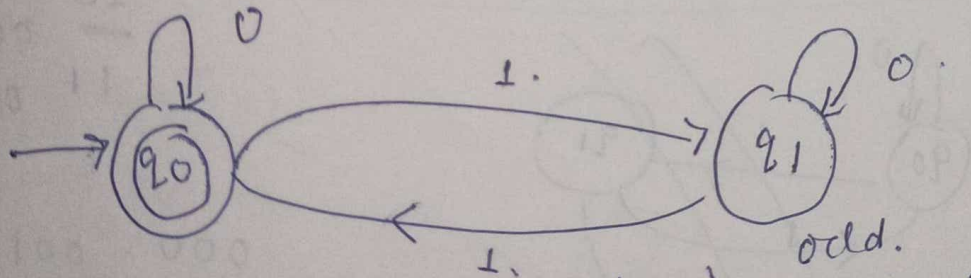
accepts a string w if starting at the state q_0, the DFA ends at an accept state on reading the string w.

* DFA M accepts a language $L \subseteq \Sigma^*$ if every string in L is accepted by M and no more.

denoted as $L(M)$

Ex:

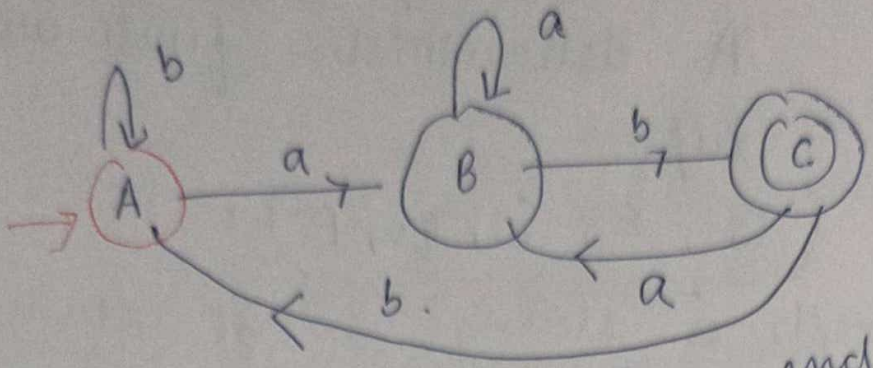
$L_1 = \{ w \in \{0,1\}^* \mid w$ has an even no. of 1's



q_0 → all strings having an even no of 1's

q_1 → all strings ——— odd no of 1's

Q2:- $L_2 = \{ w \in \{a,b\} \mid w$ ends with the substring ab $\}$



A → All strings that end with ab

B → _____ a.

A → remaining substrings

Q3 $L_3 = \{ w \in \{0,1\}^* \mid w$ is divisible by 3

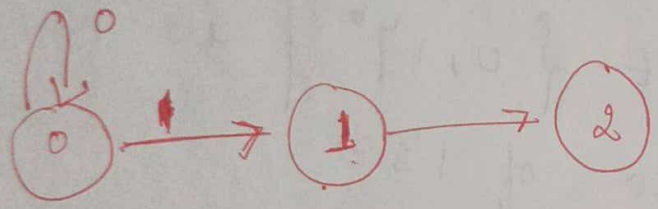
$((a+b)(a+b)(a+b))^*$

0, 1, 2

000

001

010



0

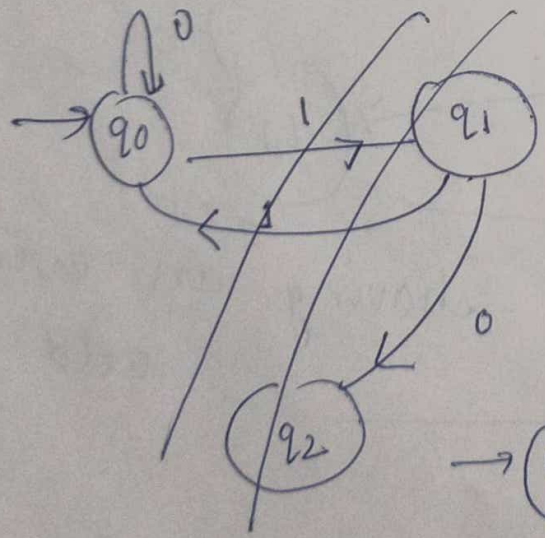
1

10

001

11 00001

1.



000 001

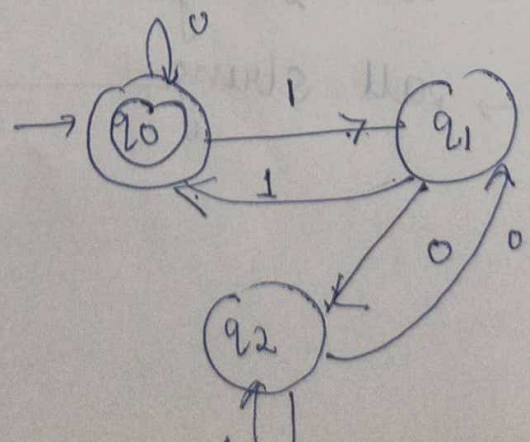
101

110

001

010

011 100



w mod 3 = 0

w mod 3 = 1

w mod 3 = 2

Q4 $L_4 = \{w \in \{0,1\}^* \mid w$ begins & ends with same symbol$\}$

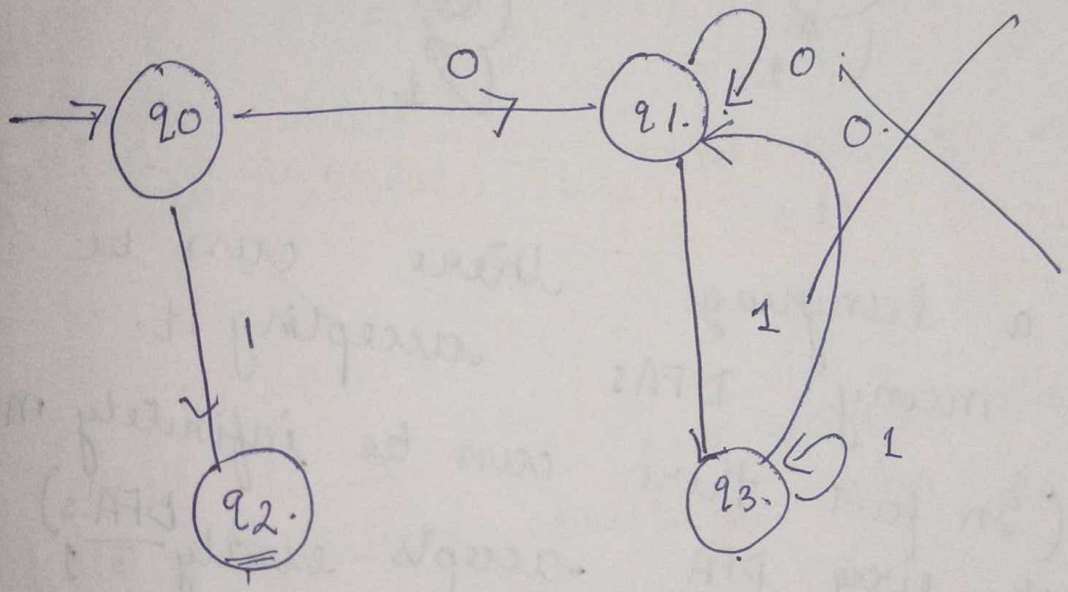
$L = \{$ q1 begins with 0 and end with 0

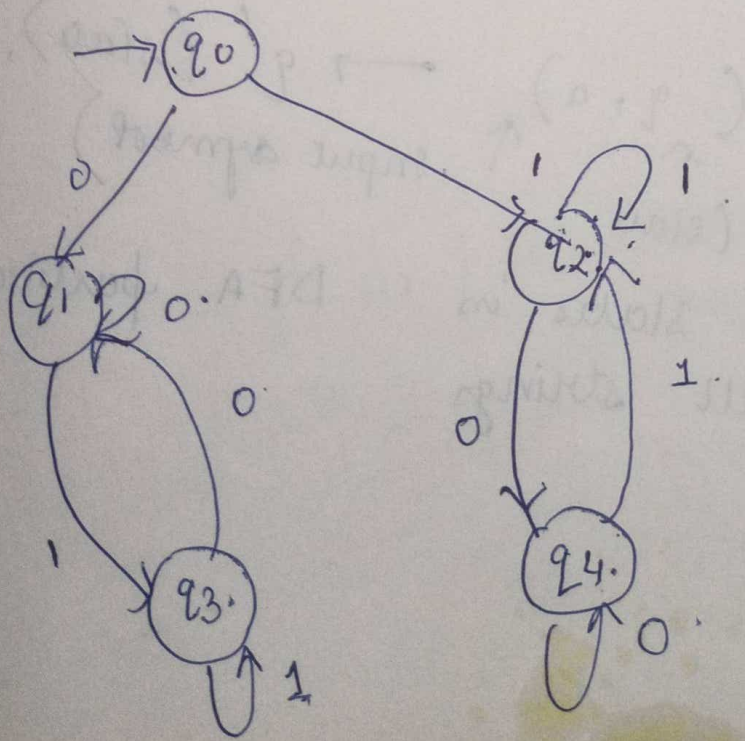
q3 beg ——— 0 and end with 1.

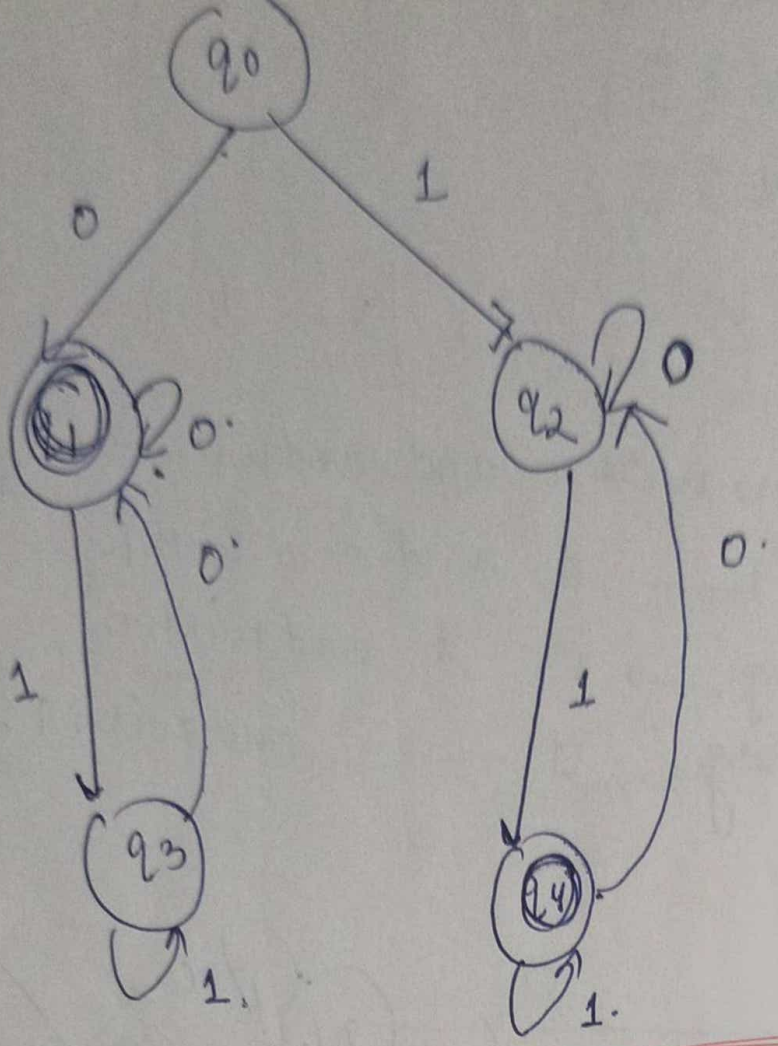
q4 beg ——→ 1 & end with 0

q2 ~~beg~~ 1 ——— end with 1.

$\}$







0 → 0
0 → 1
1 → 0
1 → 1

Deterministic finite automata

For a language there can be many DFAs accepting it.
(In fact there can be infinitely many DFA's)
But every DFA accepts exactly 1 language.

→ given a $(q, a) \longrightarrow q'$ (state).
(state) (input symbol)

The set of states in DFA partitions the set of all strings

4) Computation by DFA and Regular operation

Let $M = (Q, \Sigma, \delta, q_0, F)$ and let w be a string $w = a_1 a_2 \ldots a_n$ where $a_i \in \Sigma$.

we say that M accepts w if ∃ (there exists) a sequence of states $r_0, r_1, r_2 \ldots r_n$ such that

initial cond. (1) $r_0 = q_0$.

transition cond. (2) $r_i = \delta(r_{i-1}, a_i)$ $\forall i = 1, 2, \ldots n$.

accept cond. (3) $r_n \in F$.

Note: The states r_i need not be distinct.

Let $L \subseteq \Sigma^*$ we say that M accepts L if $L = \{w \in \Sigma^* \mid M \text{ accepts } w\}$

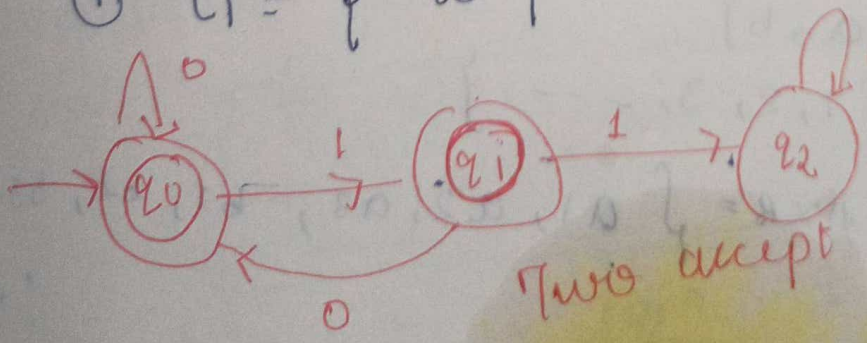
Let $L \subseteq \Sigma^*$ we say that L is regular if there exists a DFA M such that

$$L = L(M).$$

$\therefore L = \{0^n 1^n \mid n \geq 0\}$ is not regular.

Examples

① $L_1 = \{w \mid w$ does not contain 11 as a substring$\}$



Dump state → from where the automaton can't reach an accept state

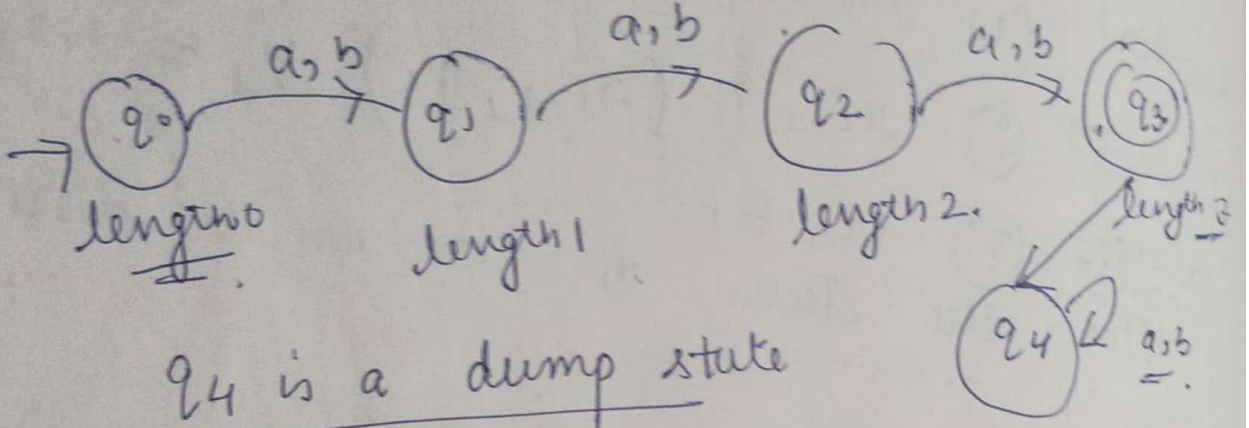
→ 0110

~~tot~~ → 101 ✓

100 → q_0 ✓

$L_2 = \{ x \mid |x| = 3 \}$.

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



q_4 is a dump state

Regular Operations Let $A, B \subseteq \Sigma^*$

(1) Union $A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$.

(2) Concatenation

$A \cdot B = \{ xy \mid x \in A \text{ and } y \in B \}$

also denoted as AB.

(3)

$A = \{ a, b \}$

$B = \{ 1, 2, 3, \ldots \}$

$AB = A \times B = \{ a1, a2, a3, b1, b2, b3 \ldots \}$

(3) Star Operation (unary operation)

$A^* = \{ x_1 x_2 \ldots x_K \mid K \geq 0 \text{ and } x_i \in A \text{ for all } i \}$

eg $A = \{10, 001\}$

$A^* = \{ \wedge, 10, 001, 1010, 10001, 00110, 001001 \}$

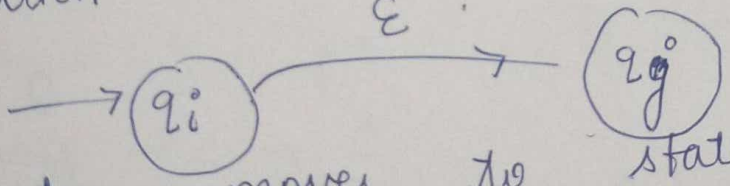
---

(5) Non-Determinism → from a state q_i on an input symbol a, the automaton can go to multiple states

K ($K \geq 0$)

computation happens simultaneously along each of these paths

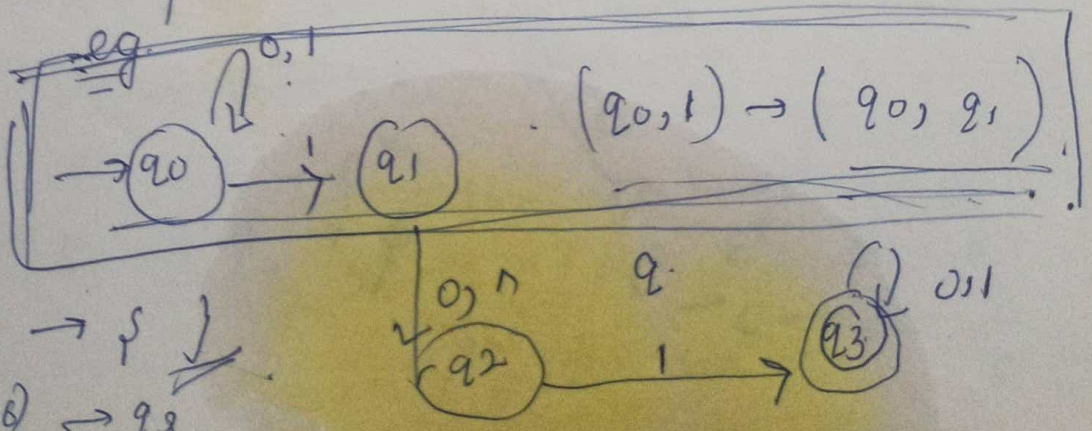
→ has ε transitions → if there is a ε-transition



then, automaton moves to state q_i and q_j without reading the next input bit.

→ An input is accepted if there is some computation path that leads to an accept state

eg



$(q_0, 1) \rightarrow (q_0, q_1)$

$(q_1, 1) \rightarrow \{\}$

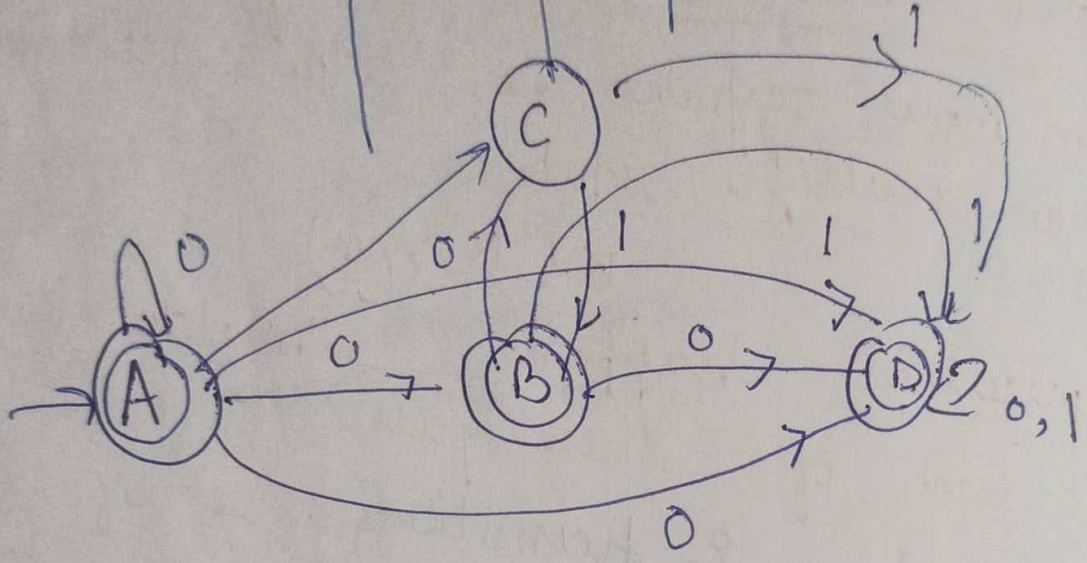
$(q_1, 0) \rightarrow q_2$

$(q_1, , \varepsilon) \rightarrow \{q_2, q_2\}$ The transitions are labelled with symbols from the set

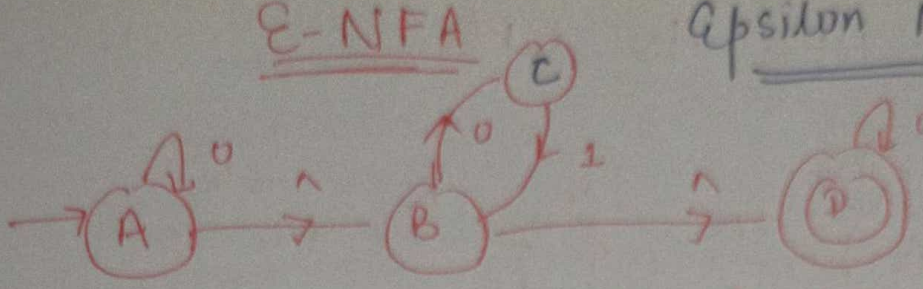
$$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$$

Consider an input $w = 0101110$

States	~~ε~~	0	1
$\{q_0\}$		$\{q_0\}$	$\{q_0, $



ε-NFA — Epsilon NFA



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

$\delta: Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$.

Epsilon closure (A) = what are the states

$A \xrightarrow{\varepsilon} \{A, B, D\}$ $A = \{A,$

ε-NFA to NFA

	0	1
→ A	{A, B, C, D}	{D}
B	{C, D}	{D}.
C	∅	{B, D}.
D	{D}	{D}.

D: $\varepsilon^* 0 \varepsilon^*$ → D → D → D

$\varepsilon^* 1 \varepsilon^*$ → D → D → D

A: $\varepsilon^* 0 \varepsilon^*$ → A, B, D · A

$\delta(\varepsilon\text{-closure}(A))$

B: $\varepsilon^* 0 \varepsilon^*$ → B → B → C → C

D → D → C

$\varepsilon^* 1 \varepsilon^*$ → B → B → ∅

D → D → D

C: $\varepsilon^* 0 \varepsilon^*$ → C → C → ∅ → ∅

A: $\varepsilon^* 0 \varepsilon^*$ → A → A → A, B, D

B → C → C

D → D → D.

$\varepsilon^* 1 \varepsilon^*$ → A → A → ∅

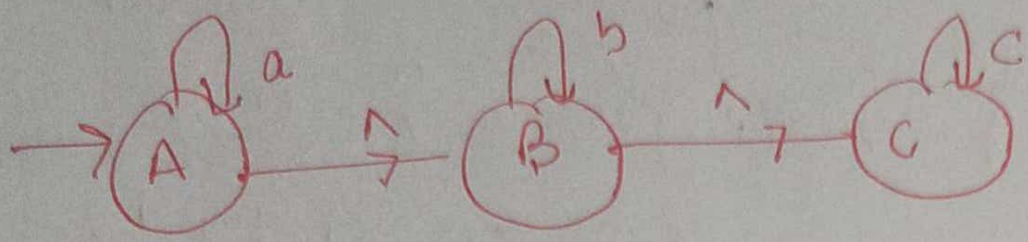
B → ∅

D → D → D

C: $\varepsilon^* 1 \varepsilon^*$ → C → B → B

D.

ε-NFA to NFA



	a	b	c
A	{A, B, C}	{B, C}	{C}
B	~~{B, C}~~	{B, C}	{C}
C	ϕ	ϕ	{C}

ε* a ε*

A → A → A, B, C

B → B → B, C

C → ϕ

ε* b ε*

A → A → ϕ

B → B → B, C

C → ϕ

ε* c ε*

A → A → ϕ

B → ϕ

C → C → C

ε* a ε*

C → C → ϕ → ϕ

ε* b ε*

C → C → ϕ → ϕ

ε* b ε*

B → B → B, C

C → ϕ

a ε*

B → ϕ → ϕ

C → ϕ

ε* c ε*

B → B → ϕ

C → C → C

Lec 6. Week 2

Non-Deterministic Finite Automata

Difference between DFA & NFA

DFA	NDFA
(1) (q, a) → single state	(1) (q, a) → multiple states
(2) single computation	(2) multiple computation
(3) no ϵ-transition	(3) have λ-transition
(4) Accept if the computation ends at an accept state	(4) accept if one of the computation paths end at an accept state

NFA is the five tuple

$N = (Q, \Sigma, \delta, q_0, F)$ $\delta: Q \times \Sigma_\varepsilon \to 2^Q$

↓ power set of Q

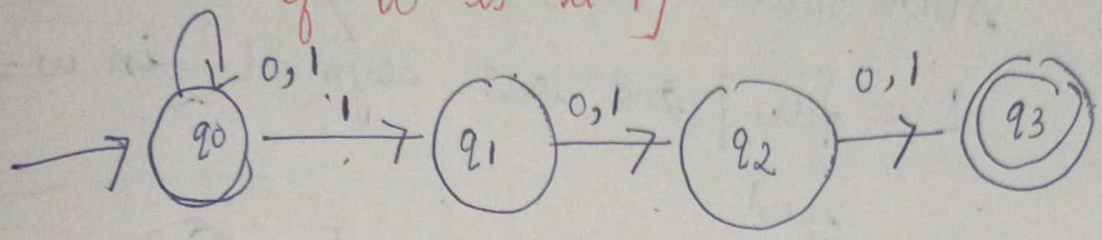
Defn We say that N accepts an input $w = a_1, a_2 \_\_\_ a_n$ if we can write w as $w = b_1 b_2 b_3 \_\_\_ b_m$ where $b_i \in \Sigma_\varepsilon$ and there exists a seq. of states $r_0, r_1, r_2 \_\_ r_m$ (not necessarily distinct) such that (1) $r_0 = q_0$ (initial condn)

(2) $r_i \in \delta(r_{i-1}, b_i)$ for all $\forall i = 1 \_\_ m$.

$q_m \in F$ (acceptance condn)

$$L(N) = \{ w \in \Sigma^* \mid N \text{ accepts } w \}$$

Examples → $L_1 = \{ w \in \{0,1\}^* \mid$ 3rd last symbol of w is a 1$\}$



$\delta(q_0, 1110) \vdash$ (1110 / 1010)

$(q_0, 1110)$

$(q_0, 110)$

$(q_1, 10)$

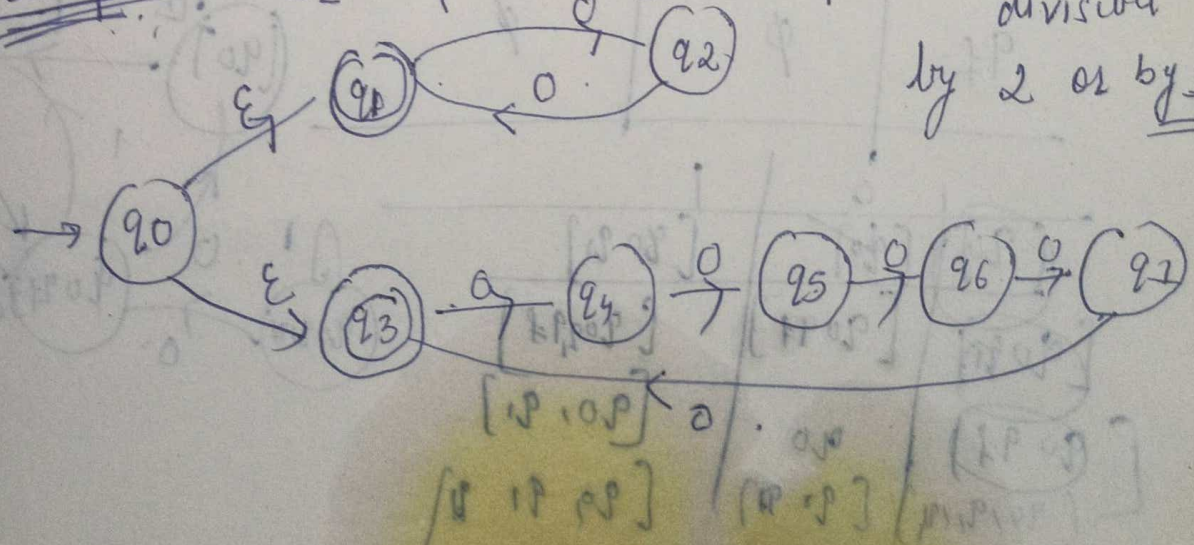
$(q_2, 0)$

$\vdash q_3$

show that there exists a DFA for L.

$L_K = \{ w \in \{0,1\}^* \mid$ the K^{th} last bit is 1$\}$.

Example 2 $L_2 = \{ w \in \{0\}^* \mid |w|$ is divisible by 2 or by 5$\}$

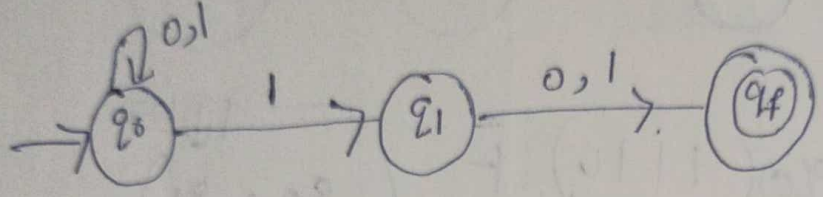


(7) lec Equivalence of NFA and DFA, Closure properties

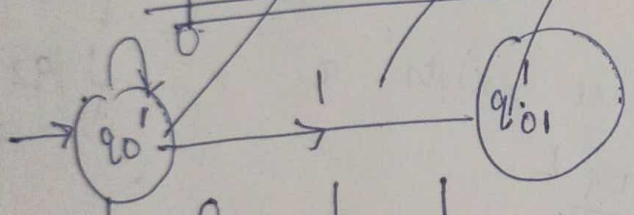
Every DFA is also an NFA.

Theorem $N = (Q, \Sigma, \delta, q_0, F)$ is NFA. there exists a DFA $D = (Q', \Sigma, \delta', q_0', F')$ such that $L(N) = L(D)$

$L = \{w \mid$ 2nd last symbol in $w = 1\}$

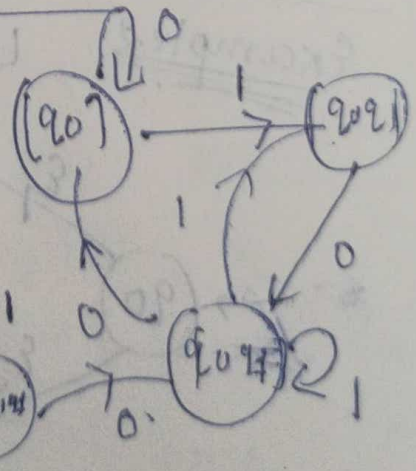


Equivalent DFA



	0	1
$[q_0]$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_f\}$	$\{q_f\}$
q_f	ϕ	ϕ

	0	1
$[q_0]$	$[q_0]$	$[q_0 q_1]$
$[q_0 q_1]$	$[q_0 q_f]$	$[q_0 q_1 q_f]$
$[q_0 q_f]$	q_0	$[q_0, q_1]$
$[q_0 q_1 q_f]$	$[q_0 q_f]$	$[q_0 q_1 q_f]$



Regular Operations

Theorem: If L_1 and L_2 are regular then $L_1 \cup L_2$, $L_1 \cdot L_2$ and L_1^* are regular.

For a regular language, we can assume that there is NFA accepting it with a unique accept state.

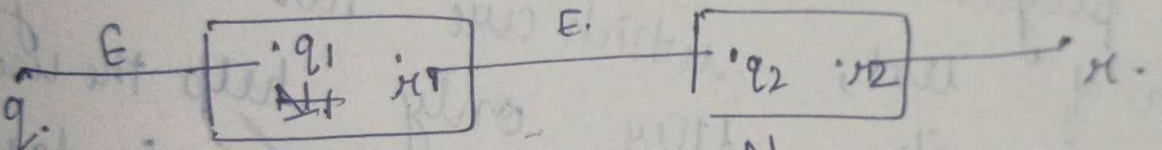
Language	NFA	start state	accept state
L_1	N_1	q_1	r_1
L_2	N_2	q_2	r_2

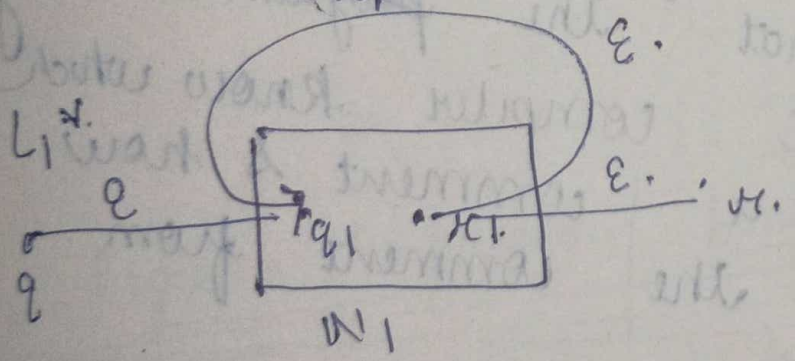
$L_1 = L(N_1)$

$L_2 = L(N_2)$

$L_1 \cup L_2$

Concatenation





L_1

L_2

Lec 8

Regular Expressions

are an algebraic way to represent languages

representation of languages

(1) → many applications in text editors the way the word application or the s/w goes about searching for a word in your document is by regular expression

(2) compiler design → when we write a C program, we put comments in C program. Comments are basically pieces of text, which are not necessary to compile, they only help the user understand what the program is doing how does the C compiler know which part of text is a comment & how does it remove the comment from main program

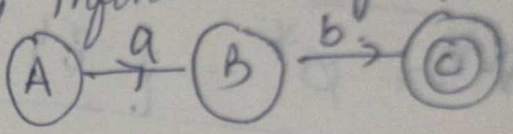
→ comment → by putting comment is by giving slash star string /*

→ Searches your entire program.

(3) Examples of Regular Expression

	Regular expression	Language
(1)	0	{0}
(2)	ε	{ε}
(3)	0 ∪ 01	{0, 01}
(4)	1^*	{∧, 1, 11, 111 ---}
(5)	$(0 \cup 01)\,1^*$ or $(0+01)\,1^*$	{0, 01, 011, 0111, 01111 ----}

It is a finite expression and capable of generating infinite language.



{ab}

ab. L = {ab}

Defn R is said to be a regular expression (RE) if R has one of the following forms :-

$(a+b)^*$ = {∧, a, b, ---} commutative

$a \in \Sigma$.

(1) Regular Expression	Language
(1) ϕ	{ }
(2) ε	{ε}
(3) a	{a}
(4) $R_1 \cup R_2$	$L(R_1) \cup L(R_2)$
(5) $R_1 \cdot R_2$	$L(R_1) \cdot L(R_2)$
(6) R_1^*	$(L(R_1))^*$
(7) (R_1)	$L(R_1)$

Note for every regular expression R there is a unique language L(R) corresponding to it but the converse is not true

Precedence of the symbols

(1) ( )
(2) *
(3) .
(4) +

** $\emptyset^* = \{\varepsilon\}$

RE	Language
(1) 01	{01}
(2) 01 + 1	{01, 1}
(3) (01 + ∧)1	{011, 1}
(4) (0 + 10)* (ε + 1)	{∧, 1, 0, 01, 10, 101, 00, 001, 010, 0101, 100, 1001, 1010, 10101 ---}

Examples (Lang) → RE

Language	Regular Expression
① $\{w \mid w$ has a single 1$\}$	$0^* 1 0^*$
② $\{w \mid w$ has atmost a single 1$\}$	$(0^* + 0^* 1 0^*)$
③ $\{w \mid \|w\|$ is divisible by 3$\}$	$((0+1)(0+1)(0+1))^*$
④ $\{w \mid w$ has a 1 at every odd position and length of $\|w\|$ is odd$\}$	$1((0+1)1)^*$

Algebraic properties of RE

Lec 9 :-

① (a) $R_1 + (R_2 + R_3) = (R_1 + R_2) + R_3$.

(b) $R_1 (R_2 R_3) = (R_1 R_2) R_3$

② (a) $(R_1 + R_2) \cdot R_3 = R_1 R_3 + R_2 R_3$

(b) $R_1 (R_2 + R_3) = R_1 R_2 + R_1 R_3$.

③ Commutativity $R_1 + R_2 = R_2 + R_1$ (only + operator is commutative)

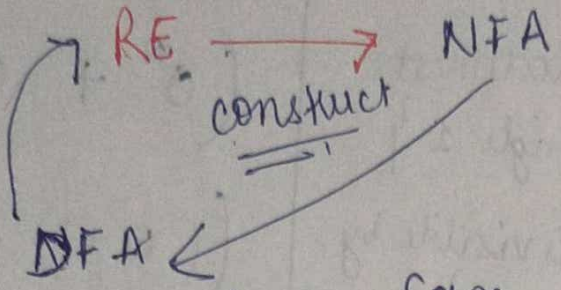
④ (a) $R_1 + \phi = R_1$

(b) $R_1 \cdot \varepsilon = R_1$

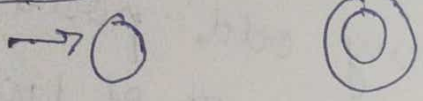
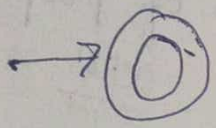
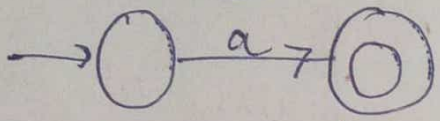
⑤ $(R_1^*)^* = R_1^*$

Theorem A language L is regular if and only if there is a regular expression

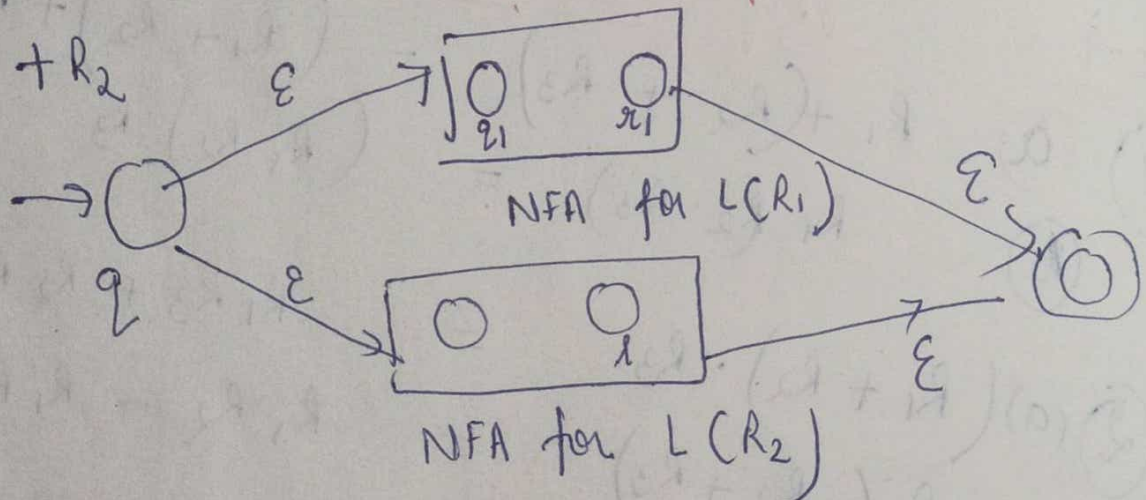
$$L = L(R)$$



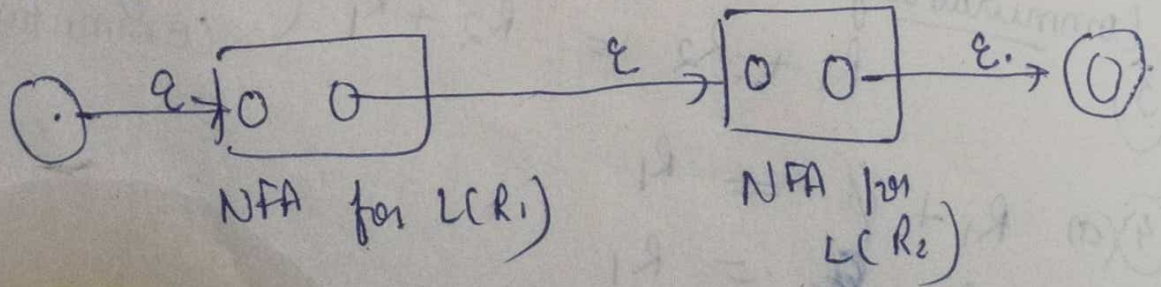
Proof is based on Case Analysis

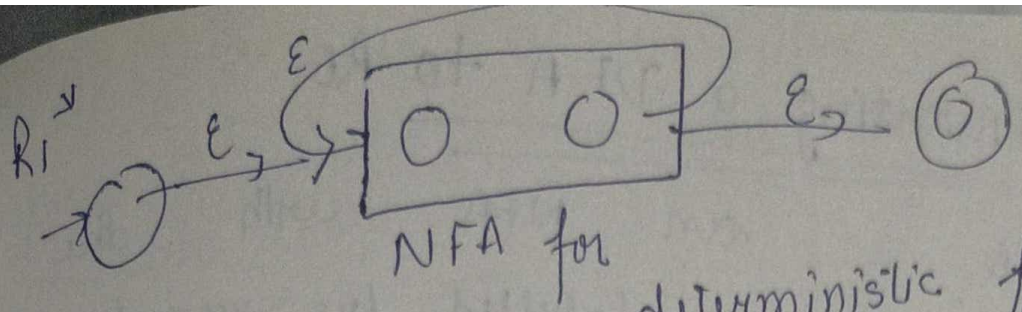
RE	NFA
(1) $\emptyset$	
(2) ε	
(3) a	

(4) $R_1 + R_2$



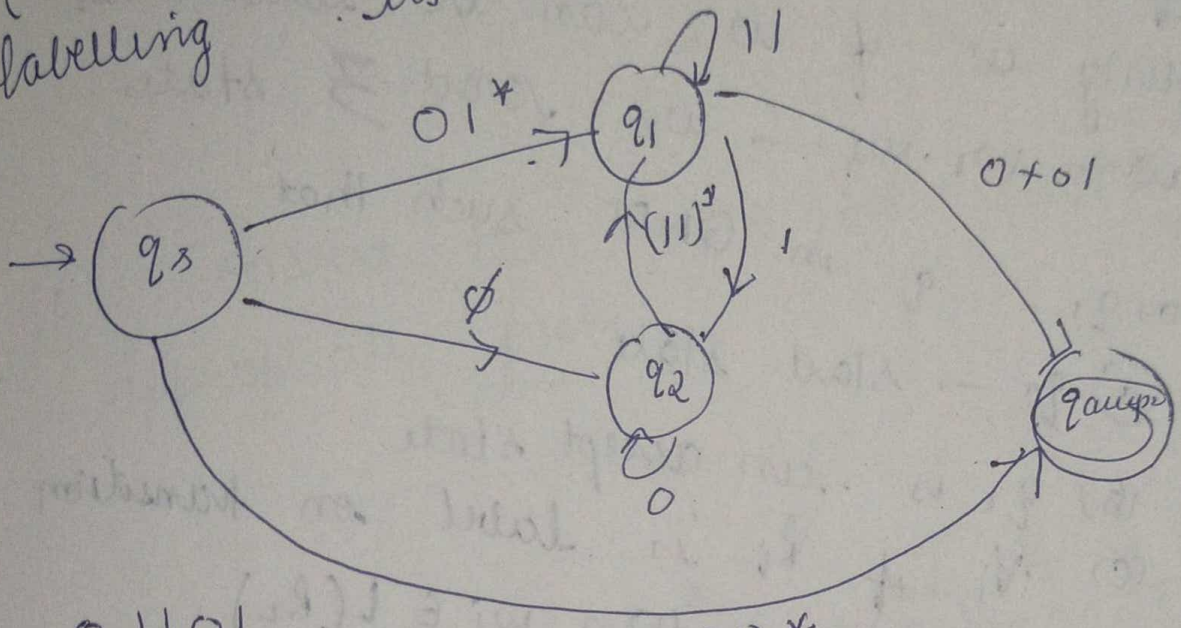
(5) $R_1 . R_2$





A generalized non-deterministic finite automaton (GNFA) is a NFA that has regular expressions as its transitions.

labelling



$q_s \xrightarrow{011} q_1 \xrightarrow{01} q_{accept}$

$q_s \xrightarrow{0} q_1 \xrightarrow{11} q_1 \xrightarrow{01} q_{accept}$

② → 10 → no way of going from q_s to q_{accept} forming string 10.

Lec 10 Converting a DFA to RE

A GNFA is an NFA with the transitions being labelled by regular expression

Defn A GNFA is said to accept a string w if w can be written as $w = w_1 \cdot w_2 \cdots w_k$ and there are states $q_0, q_1 \cdots q$ in GNFA such that

(a) q_0 → start state

(b) q_k is an accept state

(c) $\forall i$ if R_i is label on transition q_{i-1} to q_i, then $w_i \in L(R_i)$

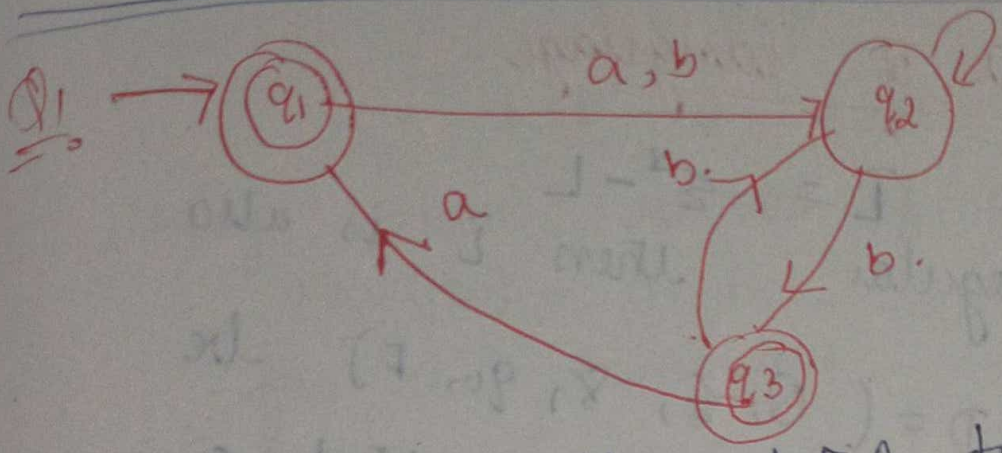
Without loss of generality, we may assume the following:-

(1) GNFA has a unique accept state

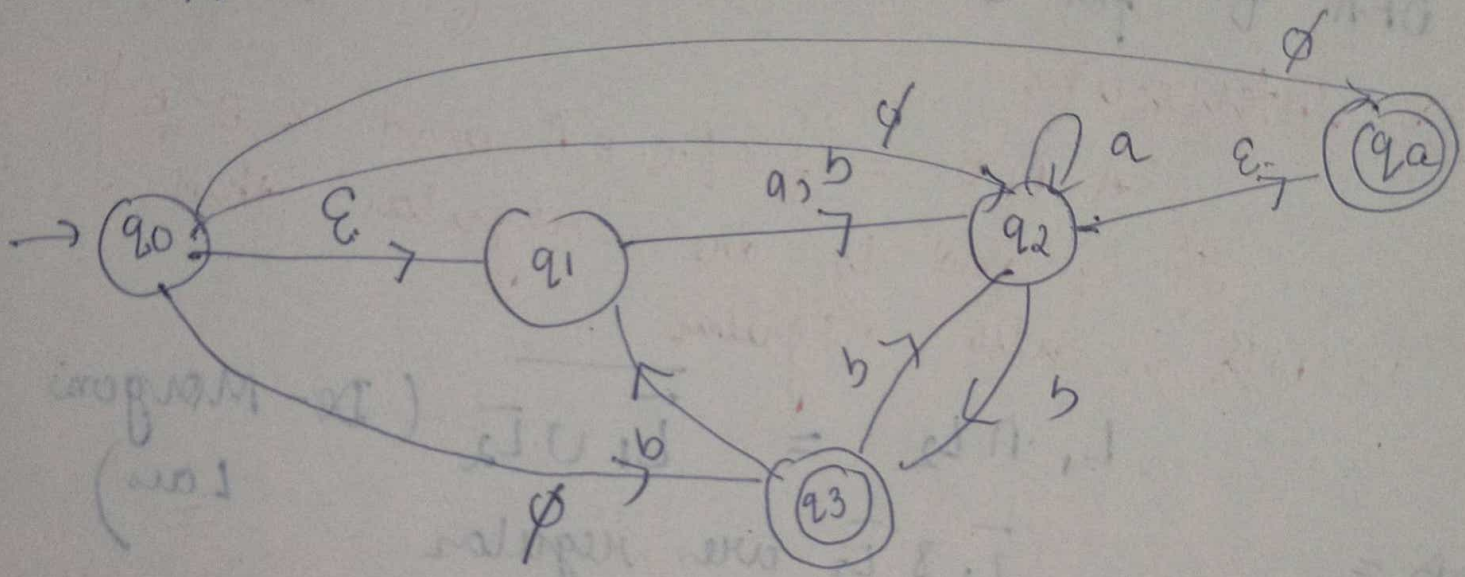
(2) There are no incoming transitions to the start state and no outgoing transitions from the accept state

(3) There are transitions from start state to every other state and from every state to the accept state.

4) There is a transition between every pair of states that are not the start/ accept state



(1) Convert the DFA to GNFA to appropriate form



Week3 Lec 11

Closure properties of Regular Languages

(1) complement of language

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

If L is regular then $\bar{L}$ is also regular. Let $D = (Q, \Sigma, \delta, q_0, F)$ be some DFA for L. We construct a DFA D' for $\bar{L}$ where, $D' = (Q, \Sigma, \delta, q_0, Q-F)$

(2) Intersection

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$

If L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular

$$L_1 \cap L_2 = \overline{\bar{L_1} \cup \bar{L_2}} \quad \text{(De-Morgan's Law)}$$

$A \cap B = \overline{\bar{A} \cup \bar{B}}$

$\bar{L_1}$ & $\bar{L_2}$ are regular

$\rightarrow \bar{L_1} \cup \bar{L_2}$ is also regular

$\Rightarrow \overline{\bar{L_1} \cup \bar{L_2}}$ is regular

(3) Set difference :-

$$A - B = \{x \in A \mid x \in A \text{ and } x \notin B\}$$

$$A - B = A \cap \bar{B}$$

If L_1 & L_2 are regular then $L_1 - L_2$ is also regular.

$L_1 - L_2 = \text{?}\ L_1 \cap \bar{L_2}$

$L_2 - L_1 = L_2 \cap \bar{L_1}$ is also regular

Reversal of Language

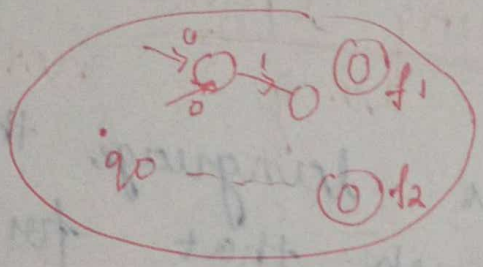
Let $w = a_1 a_2 \ldots a_n$ be a string then reverse of w is a string

$rev(w) = a_n a_{n-1} \ldots a_1$

for a language $L \subseteq \Sigma^*$

$rev(L) = \{ w \in \Sigma^* \mid rev(w) \in L \}$

If L is regular then $rev(L)$ is also regular Let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA for L.



Homomorphisms

Let Σ, Γ be two alphabets homomorphism $h : \Sigma \to \Gamma^*$ ($w = a_1 a_2 \ldots$ in Σ^* can be a string (every symbol is $\in$ it gives a string in Γ)

Let $w \in \Sigma^*$ then $h(w) = h(w) = h(a_1) \cdot h(a_2) \cdots h(a_n)$

Let $L \subseteq \Sigma^*$,

$h(L) = \{ h(w) \in \Gamma^* \mid w \in L \}$

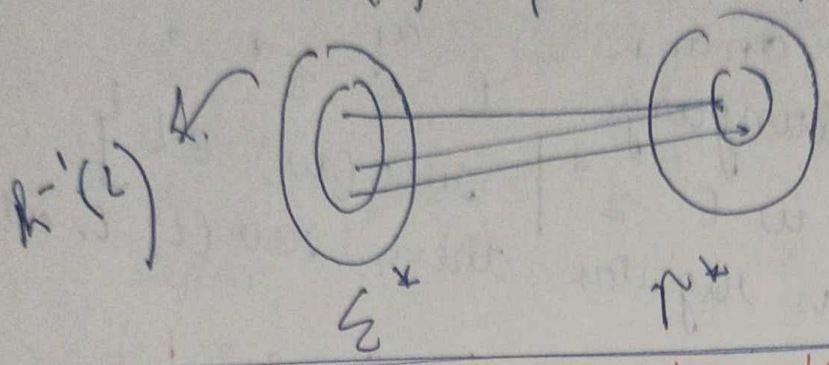
If L is regular then $h(L)$ is also regular.

Inverse Homomorphism

$h : \Sigma \to \Gamma^*$ is a homomorphism

$\rightarrow L \subseteq \Gamma^*$, define

$$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$$



Lec 12

Non-regular Languages

which requires some sort of counting.

Pumping Lemma

To prove that lang is not regular

If L is a regular language there exists $p > 0$, such that for all strings x_0 where $|x_0| > p$ there exists a partition

$|w| > p$.

$|w| = xyz$

$|xy| \leq p$

and $|y| > 0$.

$\forall$ for $i \geq 0$,

$xy^iz \in L$.

$A \Rightarrow B$ — no is even

then $\sim B \Rightarrow \sim A$ (Contrapositive form).

Example

$L = \{0^n 1^n \mid n \geq 0\}$

Soln Let p = $w = 0011$ $n = 2$.

$n = 3$

$w = 000111$

$w = xyz$.

$w = \underline{00}\,\underline{0}111$

$x = 00$

$y = 01$

$z = 11$

$w = \underline{00}\ \underline{01}\ \underline{11}$

$x \quad y \quad z$

$xy^i z =$ $i=0 \Rightarrow xz = 0011$

$i=1 \Rightarrow xyz = 000111$

$i=2 \quad xy^2z = \underline{00}\,0\underline{101}\,11$

$xy^2z \notin L$.

next

$p = 5$.

$|w| = 6$. $p = 5$.

$|xy| \leq p$.

$|xy| \leq 5$.

Lec 13 More examples of non-regular languages

(1) $L_1 = \{ 0^n 1^n \mid n \neq 0 \}$ is not regular

(2) $L_2 = \{ a^l b^m c^n \mid l+m \leq n \}$

$l = 1$
$m = 1$
$c = 2$

~~$\{ a$~~

(3) $L_3 = \{ w \in \{0,1\}^* \text{ no of 0's} = \text{no of 1's in } w \}$

$L \cdot (0^* 1^*) \cap L_3 =$

(4) ~~$L \cap L' = L''$~~ If L is regular and L' is not regular it does not imply that L'' is not regular

(5) If L and L' are non-regular even then L'' may be regular

*

$L_4 = \{ 0^p \mid p \text{ is prime} \}$ is not regular

$\{2, 3, 5, \ldots\}$

010. $(01)^*$

$L_1 \cup L_2$

$(L_1 \cup L_2)^2$

(6.) q 6.

010101

$\Sigma^2 = \{a, b\}$

a+b {a, b}

$\Sigma^2 = \{aa,$

0+11 {0, 11}

(4)

$2^2 \cdot 4$

$4^4 = 256$

$\{0^* 1 0^*\} \cup \{0^* 1^* 0\}$

000100101

$L_1 L_2^* \cup L_1^*$

{100101, 000}

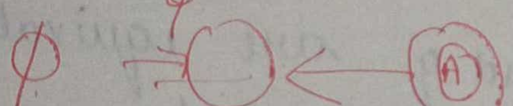
L_1 L_2

$\phi\, 0^* \Rightarrow \phi$

1110 101

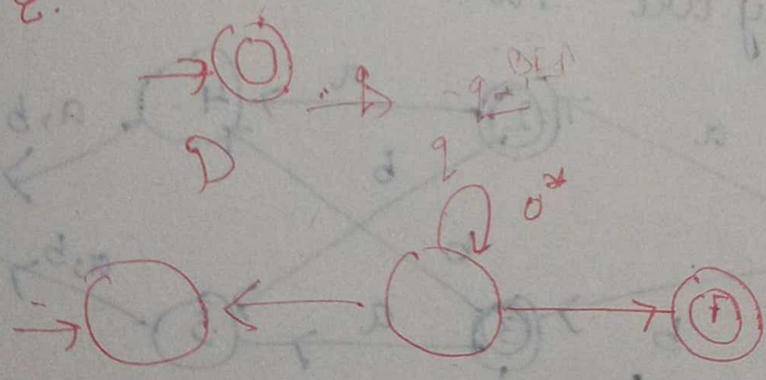
$\phi \cup \phi^*$

101.



$\{\phi, \varepsilon\}$

ε.



$L_1 L_2^*$

$\phi\, 0^* \Rightarrow \phi$

$\phi \cup \phi^*$

$\phi^* \rightarrow \varepsilon$

$\phi \cup \phi^2 \cup \varepsilon$

$\phi, \{\varepsilon\}$

NDF → (q) p q ASS

$L_1 = \{4, 5, 6, 7\}$

ϕ

(q)

$L_2 = \{0, 1, 2\}$

Lec 14

DFA minimization

DFA minimization algorithm that produces a minimal DFA from given DFA

Algorithm

(1) Input : $D = (Q, \Sigma, \delta, q_0, F)$

(1) (a) Construct a table of all $\{p, q\}$ pairs where p, q are $\in Q$

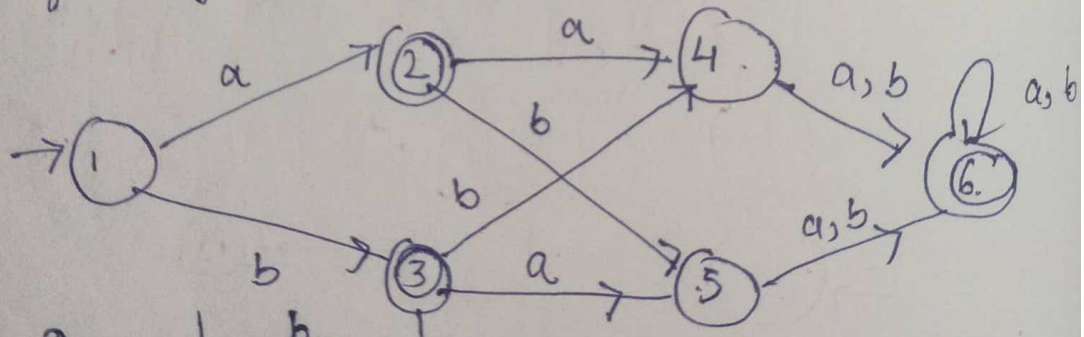
(2) Mark a pair $\{p, q\}$ if $p \in F$ and $q \notin F$ or vice versa

(3) Repeat the following until no more pairs can be marked.

Mark $\{p, q\}$ if $\{\delta(p, a)$ & $\delta(q, a)\}$ is marked for some $a \in \Sigma$.

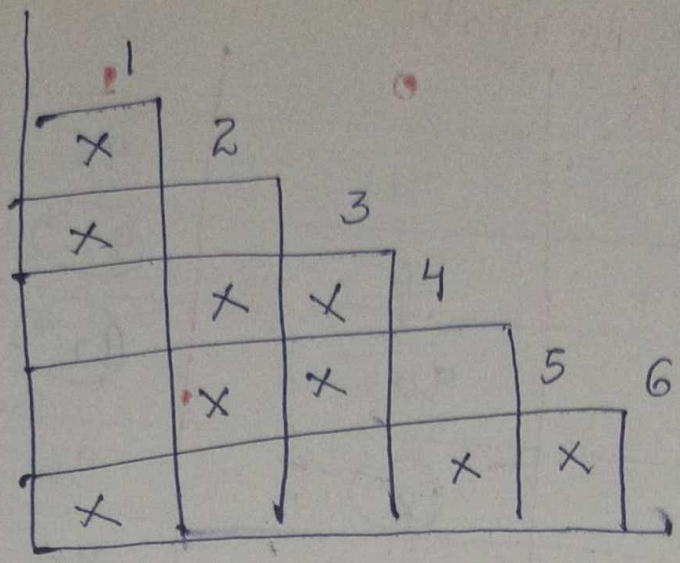
4) Two or more states p and q are equivalent if they are not marked.

eg.



	a	b
1	2 ✓	3 ✓
2	4	5
3	5	4
4	6 ✓	6 ✓
5	6 ✓	6 ✓
6	6 ✓	6 ✓

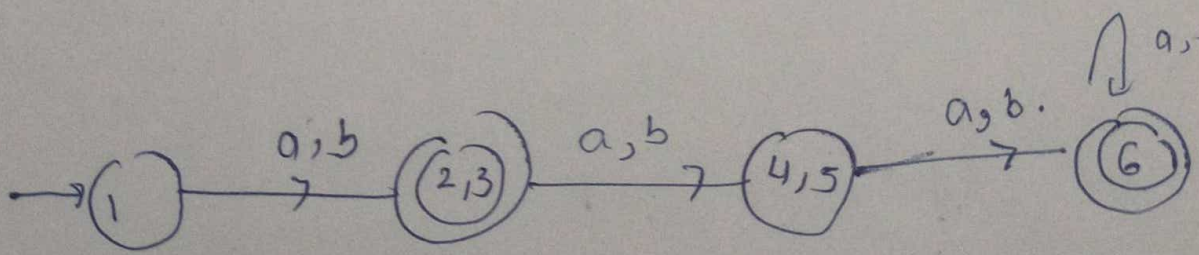
2, 3, 6.



$\pi_0 = \{1, 4, 5\}, \{2, 3, 6\}$

$\pi_1 = \{1, 4, 5\}, \{2, 3\} \{6\}$

$\pi_2 = \{1\} \{4, 5\}, \{2, 3\}, \{6\}$



Minimum Automata

State	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2^*
q_2	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

$\Pi_0 = \{ \{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\} \}$

$\Pi_1 = \{q_2\}, \{q_0, q_4, q_6\} \{q_3, q_5\} \{q_1, q_7\}$

$\Pi_2 = \{q_2\}, \{q_0, q_4\} \{q_6\}, \{q_3, q_5\} \{q_1, q_7\}$

$\Pi_3 = \{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_3, q_5\}, \{q_1, q_7\}$

Page 95

Sol

State	a	b
q0	q1	q0
q1	q0	q2
q2	q3	q1
q3	q3	q0
q4	q3	q5
q5	q6	q4
q6	q5	q6
q7	q6	q3

Π_0 $\{q_3\}$ $\{q_0, q_6\}$ $\{q_1, q_5\}$ $\{q_2, q_4\}$ $\{q_7\}$

---

q0

q1

q2

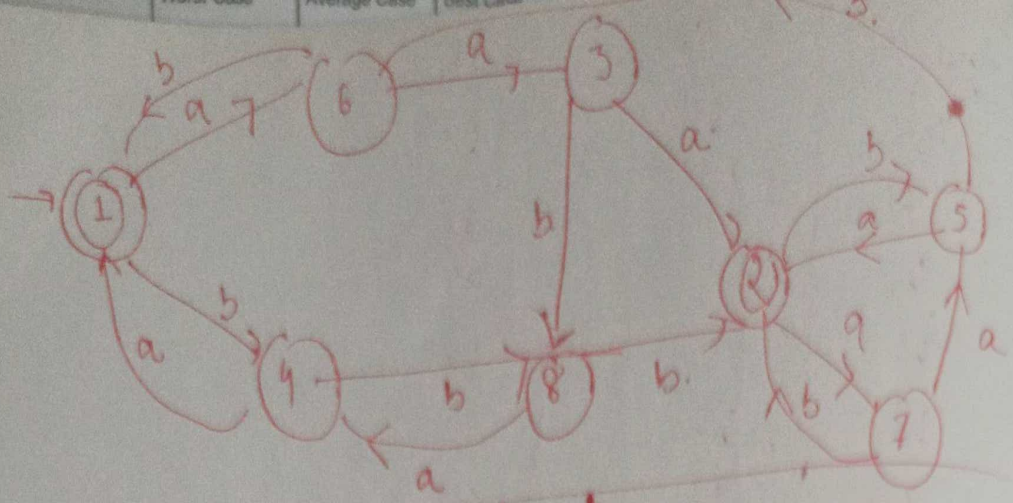
q3

q4

q5

q6

q7

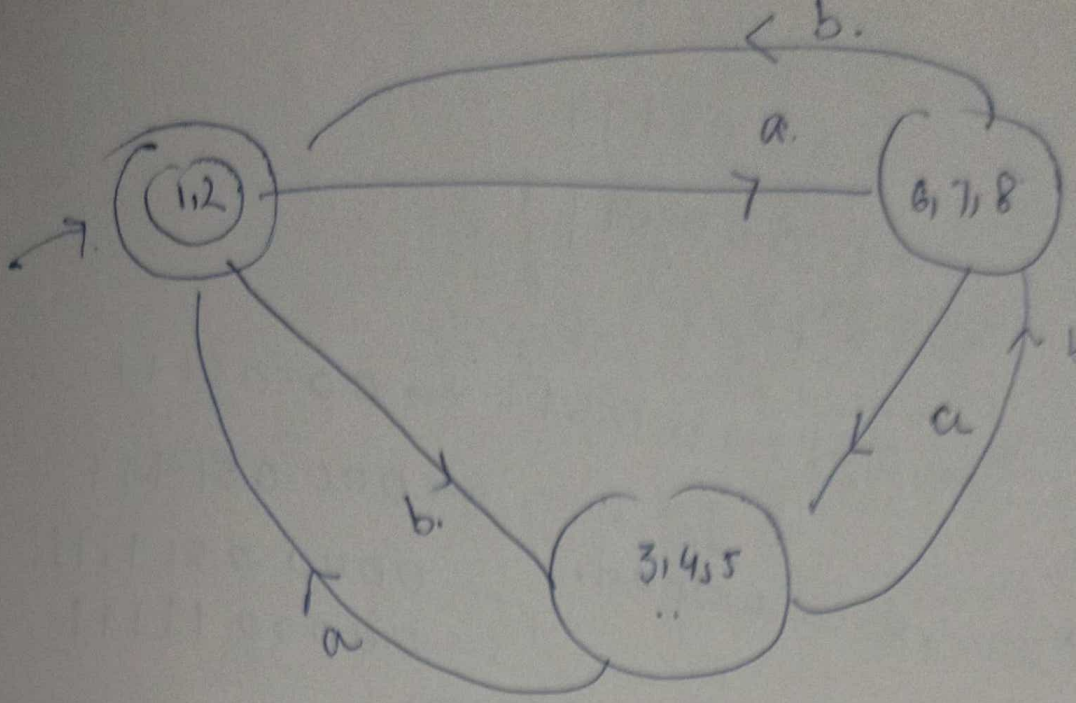


State \ Σ	a	b
1	6	4
2	7	5
3	2	8
4	1	8
5	2	6
6	3	1
7	5	2
8	4	2

$\Pi_0 = \{1, 2\}\ \{3, 4, 5, 6, 7, 8\}$

$\Pi_1 = \{1, 2\}\ \{3, 4, 5\},\ \{6, 7, 8\}$

$\Pi_2 = \{1, 2\}\ \{3, 4, 5\}\ \{6, 7, 8\}$



a^b / (a^b is power)

Lec15 Introduction to CFGs

Recognize a larger class of languages than regular languages.
→ Application in programming languages & compiler design

→ Terminal symbols. similar to the alphabet symbols.
→ Variable symbols → can be replaced with a string of variables & terminals.
Production Rules →.
Start variable → starting point of the computation.

e.g.
$S \to 0S1$
$\to 00S11$
$\to 0011$

Terminals {0, 1}
Variables {S}

S[at Variable :- S.

$S \to 0S1$
$S \to \wedge$.

$\rightarrow 00\underline{S}11$

$\rightarrow 00\,0S11$

$\rightarrow 000111$

$w = 0^5 1^5$.

$S \rightarrow 0S1 \rightarrow 00S11 \rightarrow 00\,0\underline{S}11$

$\rightarrow 000\,0S\,111$

$\rightarrow 0000\,0S\,1111$

$\rightarrow 00000\,11111$.

Hence, w is accepted.

Every string of the form $0^n 1^n \mid n \geq 0$ is accepted by grammar

Example 2.

Terminals:- $\{0, 1\}$

Variables:- $\{S, A, B\}$

$S \rightarrow ASB \mid \wedge$.

$A \rightarrow 0$

$B \rightarrow 11$

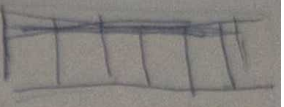
$S \rightarrow ASB \rightarrow 0SB \rightarrow 0B \rightarrow 011$

$S \rightarrow ASB \rightarrow 0SB \rightarrow 0ASBB \xrightarrow{\wedge} 0ABB \xrightarrow{B \rightarrow 11} 0A11B \xrightarrow{B \rightarrow 11} 0A1111 \rightarrow 00\,1111$

$L = \{0^n 1^{2n} \mid n \geq 0\}$

Lec 16

Examples of CFGs, Reg



Defn – A CFG is a 4 tuple

(V, Σ, P, S) ⟶ Start variable

V ↓ Variable, Σ ↓ Input alphabet

$\rightarrow S$

$$P \subseteq V \times \{V \cup \Sigma\}^*$$

Let $A \to w$ be a production rule.

Let $u, v \in \{V \cup \Sigma\}^*$.

Then we say that the string uAv yields uwv in one step

For

$u \overset{*}{\Rightarrow} v$ (in zero or more steps)

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

A language L is said to be a CFL if there exists a CFG, $L = L(G)$

eg

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

$S \to ASB \mid \wedge$

$A \to a$

$B \to bb \mid b$.

(2) $L_2 = \{w \in \{0,1\}^* \mid$ no. of 0's in w = no. of 1's in w.

$w = 0 \cdots 1$

$S \to 0S1S \mid 1S0S \mid \wedge$

③ Language of balanced parenthesis

→ string over '(' and ')' such that they are balanced and well matched

((()) ()

() () ()
11 22 33

$L_3 = \{ w \in \{(,)\}^* \mid w \text{ is balanced} \}$

$S \to (S) \mid SS \mid \wedge$.

$S \to SS$
$\to (S)S$
$\Rightarrow (S)S$
$\Rightarrow ((S))S$
$\Rightarrow (())S$
$\Rightarrow (())(S)$
$\Rightarrow (())()$

Regular languages are context free

Let L be a regular language.
∃ a DFA $D = (Q, \Sigma, \delta, q_0, F)$
such that $L = L(D)$

Construct a CFG.

$V = \{ R_i \mid q_i \in Q \}$

Then add the rule

$R_i \rightarrow aR_j$

$q_i \in F$, then add the rule

$R_i \rightarrow \wedge$.

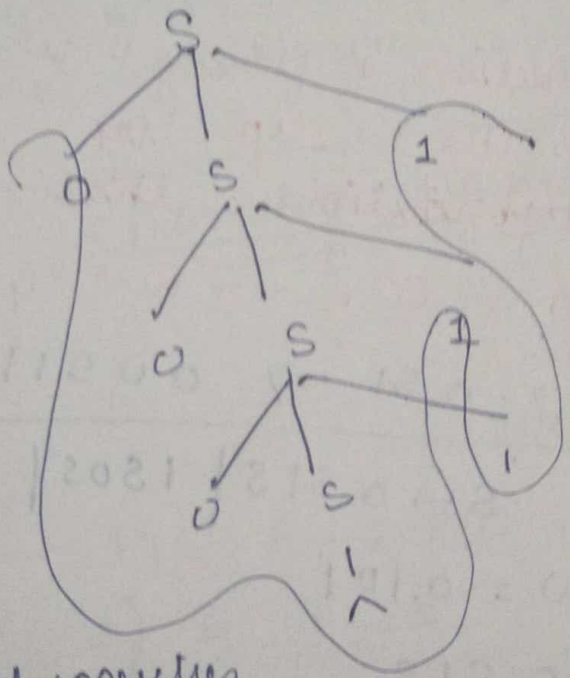
Lec 11 Parse trees and Ambiguity

Let G be a CFG and w be a string $\in L(G)$. A parse tree of w with respect to G is a rooted, ordered tree that represents the derivation of w. Syntactic structure of w

eg $G_1 \rightarrow S \rightarrow 0S1 \mid \wedge$.

$w = 000111$

000111.



Some properties

→ Every internal node is a variable

→ Every leaf node is either a terminal or ε.

If it is $\wedge$, then it is the only child of its parent.

Concatenation of the leaves of parse tree from left to right gives the string.

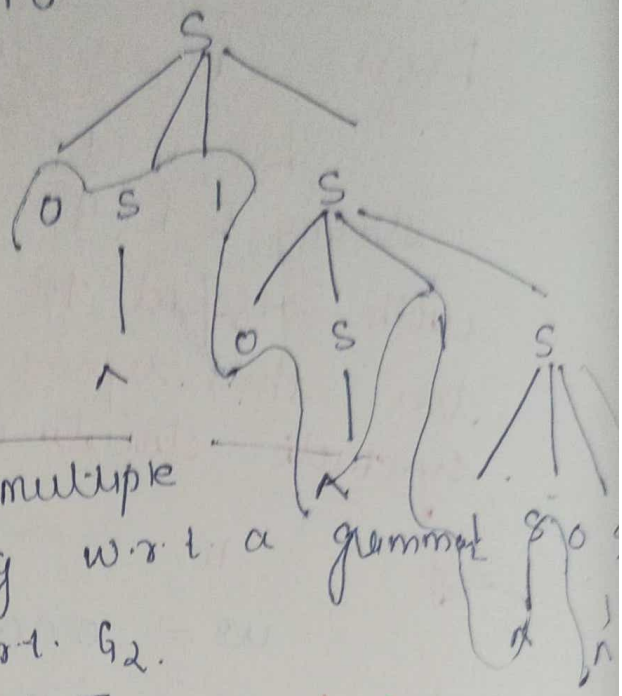
$S \to 0S1S \mid 1S0S \mid \varepsilon$.

$w = 010110$

$S \to 0S1S$

$\Rightarrow 01S0S$

$\Rightarrow 01$



There may exist multiple parse trees for a string w.r.t a grammar.

eg. 010110 w.r.t. G_2.

The derivation of string w w.r.t a grammar G is the step by step (seq. of substitutions that yields w).

eg $w = 0^2 1^2$ w.r.t G

$S \to 0S1 \Rightarrow 00S11 \Rightarrow 0011$.

$G_2 \Rightarrow S \to 0S1S \mid 1S0S \mid \wedge$

$w = 0101$

$S \to 0S1S$

$\Rightarrow 01S$

$\Rightarrow 010S1S$

$\Rightarrow 0101$

$S \rightarrow 0S1S$

$\Rightarrow 0S1\underline{0}S1S$

$\Rightarrow 0\underline{1}0\underline{1}.$

$S \rightarrow 0S1S$

$\Rightarrow 01S0S1S$

$\rightarrow 0101$

A leftmost derivation of w.r.t G is a derivation where in each step the left most variable of a string gets replaced.

eg. derivation 1 & 3

A string $w \in L(G)$ if it has two leftmost derivations for same string we say a grammar G is ambiguous if ∃ some ambiguous string $\in L(G)$.

Lec 18 Chomsky Normal Form

A CFG $G = (V, \Sigma, P, S)$ is said to be in Chomsky Normal form (in CNF) if every production rule has

1) $A \rightarrow BC$ where B, C → variable
2) & $A \rightarrow a$ where $a \in \Sigma$.
3) S does not appear on r.h.s of any rule.
4) $S \rightarrow \wedge$ may be present depending on whether $\wedge \in L(G)$ or not.

Average Case

Q Converting a CFG $G = (V, \Sigma, P, S)$ to a in CNF.

Removing λ rules

$A \to \lambda$

$B \to uAv$

$C \to u_1 A u_2 A u_3$

9

$B \to uAv \mid uv.$

$C \to u_1 A u_2 A u_3 \mid u_1 A u_2 u_3 \mid u_1 u_2 A u_3$

$\mid u_1 u_2 u_3.$

Removing unit productions

(1) $A \to B$

$B \to u$

$B \to u$

$A \to u$

(4) Shortening the RHS

$A \to u_1 . u_2 \cdots u_k \quad (k > 3)$

(5) Add variables

$A \to uv$

$A \to U_1 V_1$

$U_1 \to u$

$V_1 \to v$

eg Q1. $S \to ASB$

$A \to aASA \mid a \mid \wedge$

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

Soln

~~$S_0 \to S$~~

$S \to ASB$

$A \to aASA \mid a \mid \wedge$

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

Remove Null Transitions

$A \to \wedge$

$S_0 \to S$

$S \to ASB \mid SB$

$A \to aASA \mid aSA \mid aAS \mid as \mid a$

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

$B \to \wedge$

$S_0 \to S$

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

$A \to aASA \mid aSA \mid aAS \mid as \mid a$

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

Remove Unit Productions

$S_0 \to ASB \mid AS \mid SB$

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

$B \to SbS \mid bb \mid aASA \mid aSA \mid aAS \mid as$

Non-CFLs, pumping Lemma

$w \in L$.

$w = uv^i x y^i z$

If max. degree of T is d and height of T is h, then $|w| \leq d^h$.

Examples of non CFLs

$w = uvxyz$

$|vxy| \leq p$ & $|vy| \neq 0$.

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

(2) $L = \{ ww \mid w \in \Sigma^* \}$

$S \rightarrow AB \mid BA \mid A \mid B$

$A \rightarrow aAa \mid aAb \mid bAa \mid bAb \mid a$

$B \rightarrow aBa \mid aBb \mid bBa \mid bBb \mid b.$

$S \rightarrow AB$

$\downarrow$

$S \rightarrow aAaB$

$S \rightarrow aaaaBa$ ✓

$\Rightarrow aaaaba$

$aaaba$

$abaaa$

$S \rightarrow BA$

$\rightarrow aBaA$

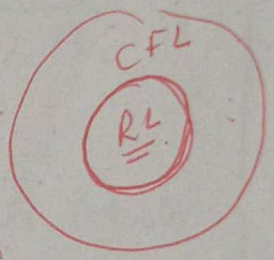
$\Rightarrow abaa$

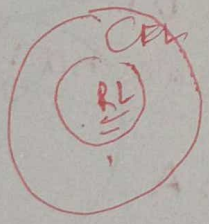
$aaba$

① Union of R and CFL $\rightarrow$ CFL

② Inter $R \cap CFL \rightarrow CFL$

③ $RL \cdot CFL \rightarrow CFL.$





$|vy| \leq n$

$u\ v\ x\ y\ z$

$ab\ ab\ ba$

$0^k 1^k$

$L_1 = a^k b^k$

$L_2 = a^n b^n$

$a^n b^n$

$uv^i xy^i z$

abb

$aaabbb.$

(a) $a\ ba$

$a\ a\ ba$

X

eg $ab\ ab\ ba$

$w = uvxyz$

$w = uv^i xy^i z$

$n = 4$

$|w| = 4.$

$|w| \geq n.$

$a\ b\ a\ x\ y\ z$

$u\ v\ ab\ ab\ ba$

$w =$

abab $ ba
u v x y z

~~aba $ ba~~

uvxyz uxz ⇒ a abba

a bb a bab B $

S → abA
A → baB
B → 0A | bb

S → ab baB
S → abbabb

S → abA
→ abbaB
→ abbaaA
⇒ abbaabaA
⇒ abbaababb

S → abA
A → baB
B → aA | bb

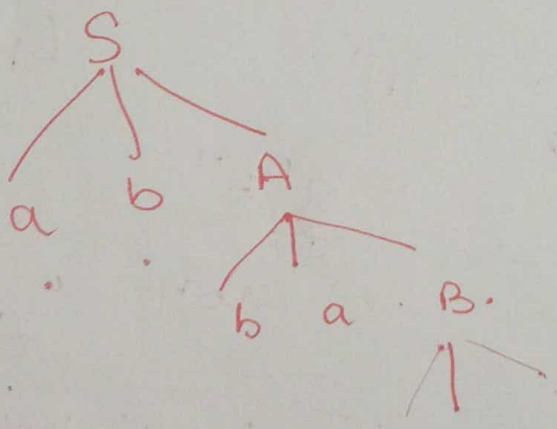
S → abA
⇒ abbaB
→ abbaaA
→ abbaababb

① ~~90~~ ✓
② ~~77~~ ✓
48

c
$0^n \$ 0^n$

$0^n 1^n 0^n 1^n \mid n \geq 0$

$a^n b^n a^n b^n$



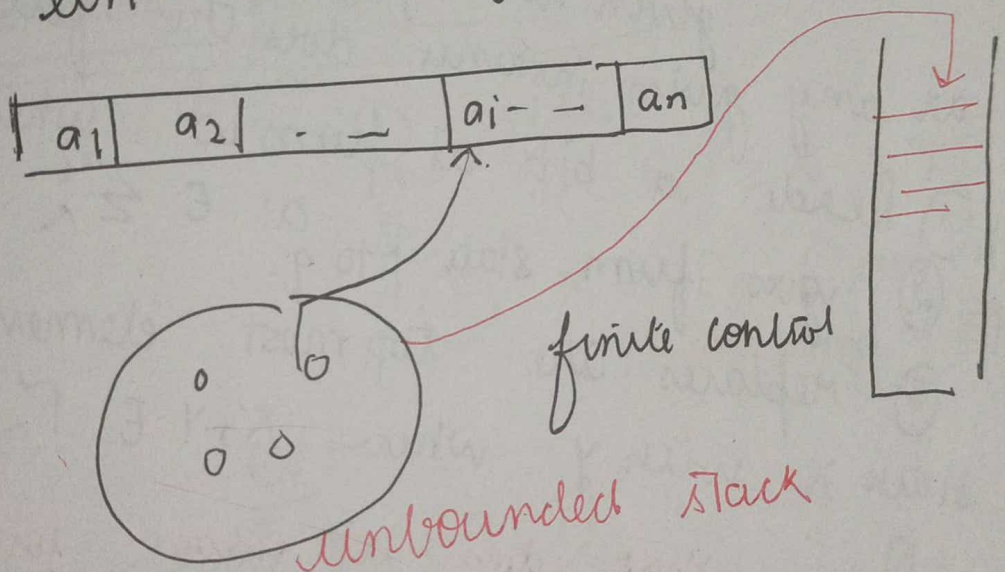
Lec 21. PDA

ε-NFA + stack

Push operation → adds an element to top of stack

Pop → removes the top most element

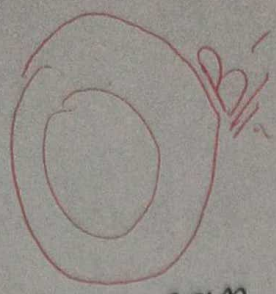
A pushdown automaton has a input tape, a finite set of states and an unbounded stack → the stack can store an arbitrary amount of info



Unbounded stack

PDA

At any given point of time, the pushdown automaton (PDA) → is in some state p, reads some input symbol, ai can access the top most element of stack. (say X)



→ A PDA at this point, can move from state p to q.

→ can change the top most element of the stack from X to Y.

$$(p, a_i, X) \rightarrow (q, Y)$$

We allow the stack to have a different alphabet (usually denoted as $\tilde{\Gamma}$)

---

given an input w the PDA at any given instance does the following

① Reads a bit a_i from w where $a_i \in \Sigma^{a_i}_{\wedge}$

② goes from state p to q.

③ replaces the top most element of stack X with Y where $X, Y \in \tilde{\Gamma}_{\wedge}$

① ≡ { What does it mean for X to be ∧ Pushing Y onto stack }

② ≡ What does it mean for Y to be ∧ Popping X from stack

22nd Lec

Formal Defn of PDA

7- 6-Sixple $(Q, \Sigma, \Gamma, \delta, q_0, F)$

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ input alphabet

$\Gamma \rightarrow$ stack alphabet

$\delta: Q \times \Sigma_\wedge \times \Gamma_\wedge \rightarrow 2^{Q \times \Gamma_\wedge}$

$q_0 \rightarrow$ start state

$F \rightarrow$ accept states.

Transition function of PDA

Input

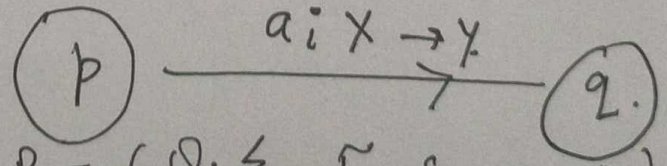
1. state
2. $a_i \in \Sigma_\wedge$ (input symbol)
3. $X \in \Gamma_\wedge$ (symbol at top of stack)

O/P

pairs of (q, Y)

$q \in Q$, $Y \in \Gamma_\wedge$

Due to non-determinism of PDA, there can be multiple transitions on the same tuple (p, a, X)



$P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is said to accept a string $w \in \Sigma^*$, if there exists

(1) a sequence of symbols.
$a_1, a_2 \text{---} a_m \; (\in \Sigma_\wedge)$

(2) States $r_0, r_1 \text{---} r_m \in Q$

(3) strings $s_0, s_1, \text{---} s_m \in \Gamma^*$

such that

(i) $w = a_1 a_2 \text{---} a_m$

(ii) $r_0 = q_0$ and $s_0 = \wedge$ (initial cond^n)

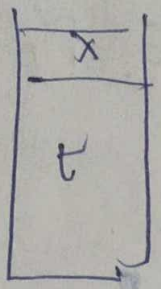
$\forall i$, if $(r_i, Y) \in \delta(r_{i-1}, a_i, X)$

then $s_{i-1} = Xt$ and

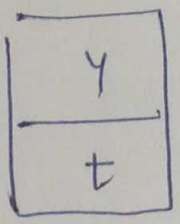
$s_i = Yt$ for some $t \in \Gamma^*$ and

$X, Y \in \Gamma_\wedge$

(iv)



Before the i^{th} step

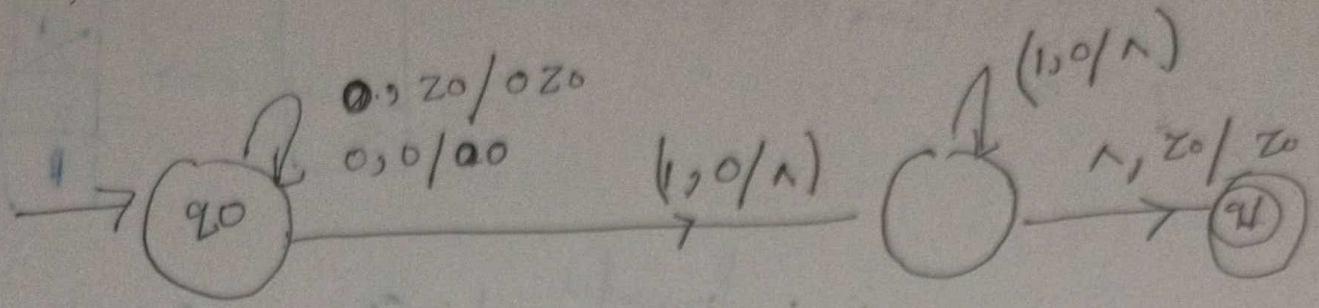


After i^{th} step

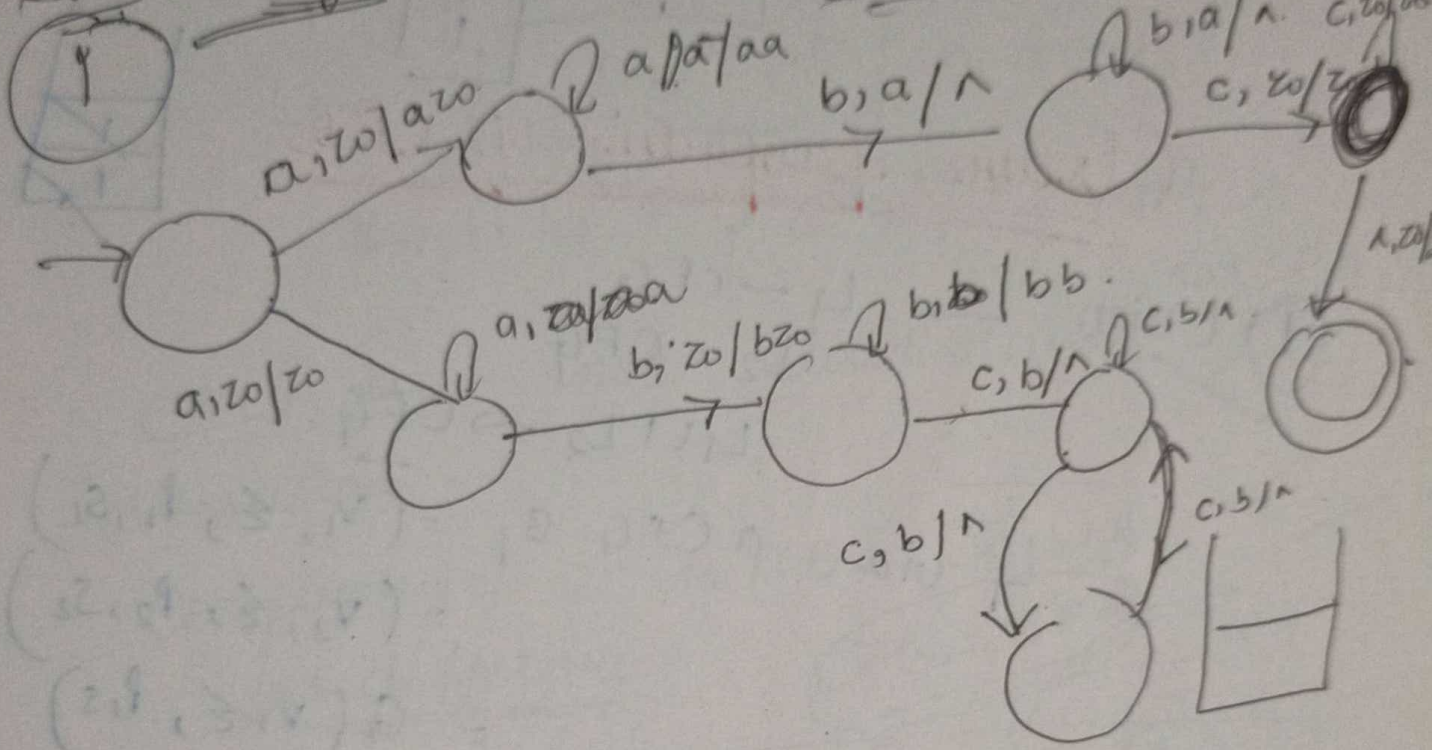
Example

$L = \{ 0^n 1^n \mid n \geq 0 \}$

$L = \{ 0^n 1^n \mid n \geq 0 \}$



Lec 23 eg $L = \{ a^i b^j c^k \mid i=j \text{ or } k \leq j \leq 2k \}$

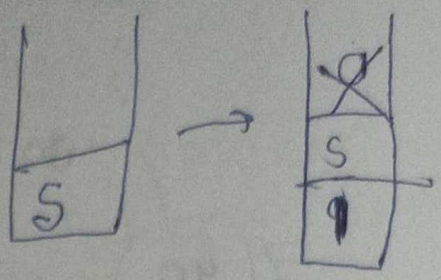


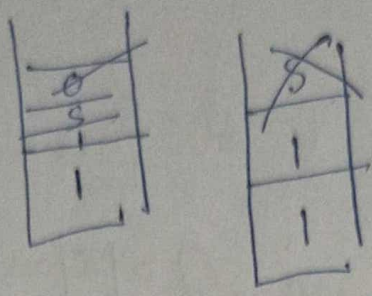
(2) $L_2 = \{ w \in \{a, b\}^* \mid w = rev(w) \}$.

abbba
abba

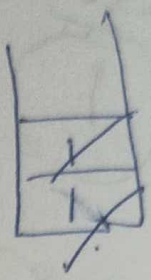
$\{0^n 1^n \mid n \neq 0\}$

$S \to 0S1 \to 00S11$





$00S11$



Closure properties of CFLs

(1) Union $\longrightarrow$ $L_1 \to CFG$

$L_2 = CFG$

$L_1 \cup L_2 \to CFG$.

L_1 via a CFG $G_1 = (V_1, \Sigma, P_1, S_1)$

$= (V_2, \Sigma, P_2, S_2)$

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

$\Rightarrow$ s.t $L(G) = L_1 \cup L_2$.

(2) Concatenation

(3) Star $\quad L(G) = L_1^*$

(4) Reverse $\quad L(G') = rev(L(G))$.

CFLs are closed under homomorphisms and inverse homomorphisms.

Intersection →

$L_1 \rightarrow$ CFG 30/175

$L_2 \rightarrow$ CFG ✓ $L_1 \cap L_2$

$L_1 \rightarrow$ CFG

$L_2 \rightarrow$ RG.

$L_1 \cap L_2 \rightarrow$ CFG.

$L_1 = \{ a^n b^n c^m \mid n, m \geq 0 \}$

$L_2 = \{ a^n b^m c^m \mid n, m \geq 0 \}$ $a^n b^n$ $a b^n c^n$

$L_1 \cap L_2 = \{ a^n b^n c^n \mid n \geq 0 \}$ is c^n not CFG a^n

CFL are not closed under intersection

Complement:

Set difference

$A \cap B = \overline{\overline{A} \cup \overline{B}}$

$a^i b^j$ $i \neq j$ $i < j$ $i > j$

CFG − CFG → CFL

CFL

CA

A − B

$a a^n b^n c c$

$a a^n b b c^n c$

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

$a^{n+1} b^{n+1} 0$

0

0

Z0

0, Z0/0

1, Z0/1

F

1

Z0

0

Deterministic

A deterministic Push Down Automaton (DPDA) is a six tuple

seven

$(Q, \Sigma, \delta, q_0, Z_0, \Gamma, F)$

such that for all $q \in Q$, $a \in \Sigma$ and $X \in \Gamma$, $\delta(q, a, x)$ has at most one element

$q \in Q$, $X \in \Gamma$, if $\delta(q, \varepsilon, x) \neq \phi$

A language L is said to be a deterministic context-free language if ∃ a DPDA M such that

$L = L(M)$

$L = \{0^n 1^n \mid n \geq 0\}$ is a DCFL

---

eg of CFL that is not a DCFL

$\{w \in \{a, b\}^* \mid w = rev(w)\}$

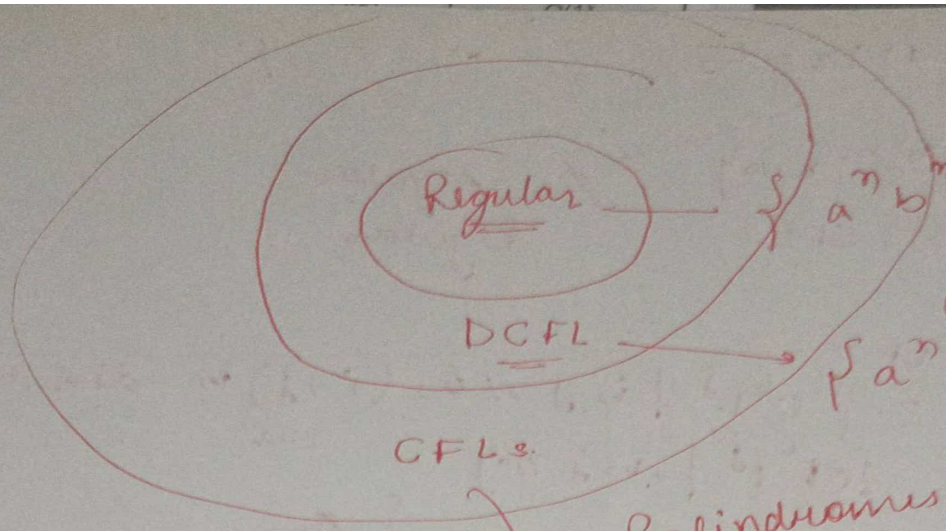
$L = \{a^n b^n c^n \mid n \geq 0$ is not a CFL

L is a CFL

It is a not a DCFL.

* DCFL are closed under complementation

DCFLs are not closed under union, intersection.



Palindromes

L = Consides the regular language $L(a^* b^* c^*)$

$L \cap L(a^* b^* c^*) =$

$L = \{w \in \{a, b, c\}^* \mid$ no of a's in w = no of b's in w = no of c's in w

Consider the regular language, $L(a^* b^1 c^2)$

$L \cap L(a^* b^* c^*)$ — regular

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

not a CFL

L is not CFL.

Bubble Sort

$L_1 \cap L_2 = L_3$ → ϕ

$\{a^n b^n c^n \mid n \geq 0\}$ → ϕ regular

not CFL regular

CFL

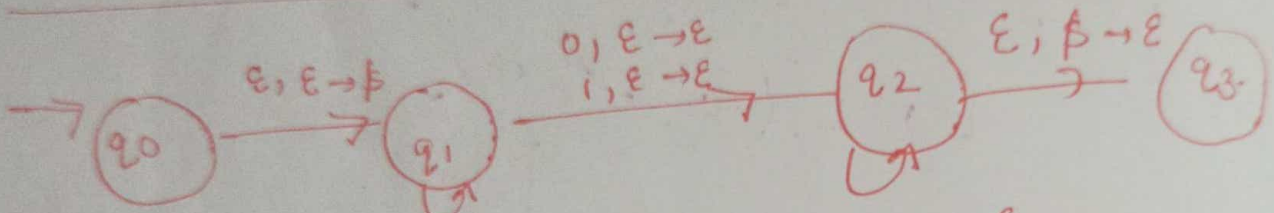
(2) $A = \{0^i 1^j \mid i, j \geq 0, (i+j)$ is divisible by $2\}$

$B = \{0^i 1^j \mid i, j \geq 0\}$

$\{A,$

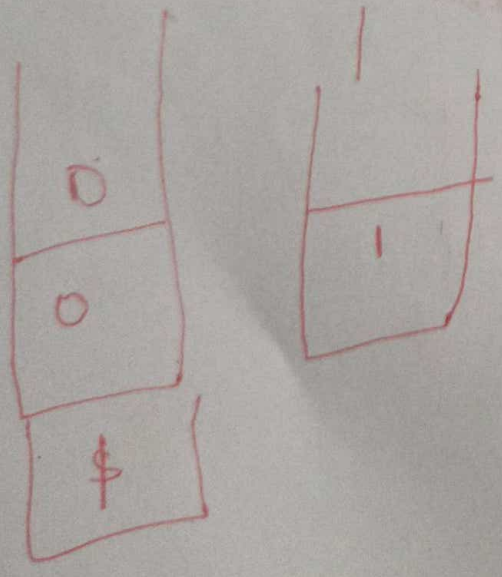
$B - A$

CFL CFL



01 10 1 0 0 1

0 1 1 0



$S \rightarrow a S_1 b S_3 c \mid a S_4 b S_2 c$

$S_1 \rightarrow a S_1 b \mid \wedge$

$S_2 \rightarrow b S_2 c \mid \wedge$

$S_3 \rightarrow S_3 c \mid \wedge$

$S_4 \rightarrow S_4 a \mid \wedge$

$S \rightarrow a a S_1 b b S_3 c c$

$\rightarrow aa\ bb\ cc$

$a\ S_4\ a\ b\ b\ S_2\ c\ \ S_2\ c$

$aa\ bbcc \qquad a$

$S \rightarrow a\ bc \mid$

$A = \{ a^i b^j \mid i \geq j \} \rightarrow$ CFG

$B = \{ b^k a^l \mid k \leq l \}$. CFG.

$a^2 b,\ a^4 b,\ a^3 b^2.$

$A \begin{cases} aab, aaaab, aaabb \ldots \\ bba, bbba \ldots \end{cases}$

$B \{$

$A = \{ \quad \rightarrow ww$ is not CFL

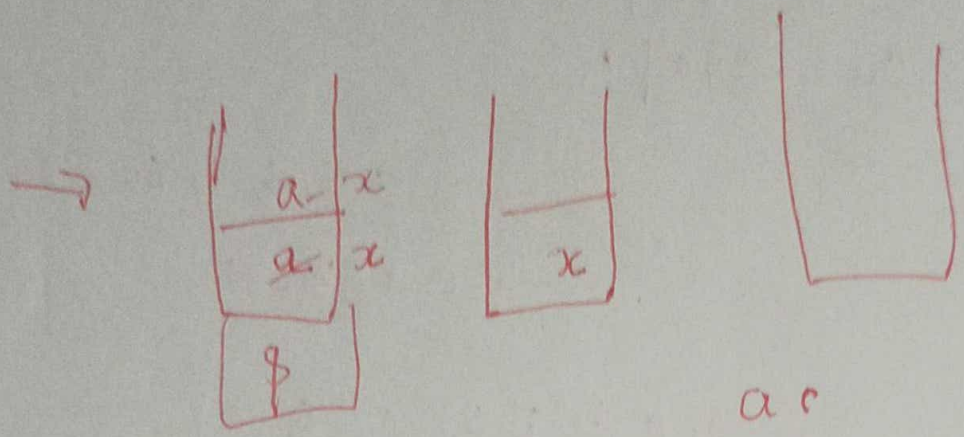
comp is CFL

A = $ww^r w^r w$ ✗ NFA

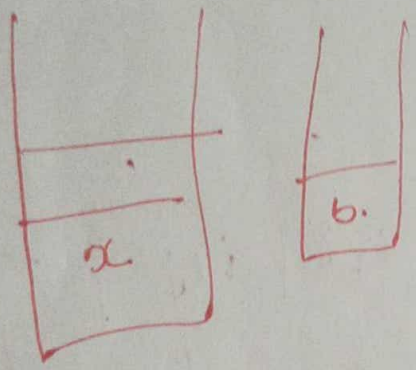
B = $ww^r x x^r$

C = $\{a^i b^j a^k b^l$

abba baab.



ac



abcc

x. xc

ab.

$ww^r \# ww^r$

abba # abba

$a^n b^n a^m b^m$ $ww^r \# ww^r$

00 01 11 10

$L_1 \& L_2 - L_1 \subseteq L_2$.

$a^2 b^2 \subseteq a^n b^n$ ✗

$a^n b^n \subseteq (a+b)^*$

$a^n b^n \subseteq a^*$.

$S \to A \mid Sb \mid a \mid b$

$A \to aS \mid Sb.$

$S \to A$

$\to aS$

$\Rightarrow aSb$

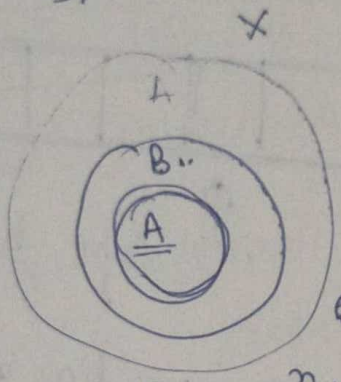
$\Rightarrow a$

$a^5 b^5$

$S \to Sb$

$\to Sbb$

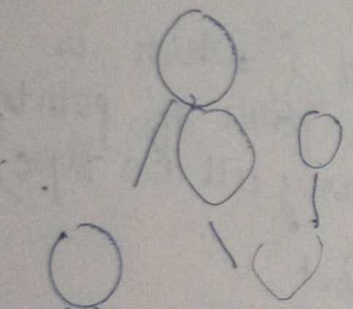
$\Rightarrow$

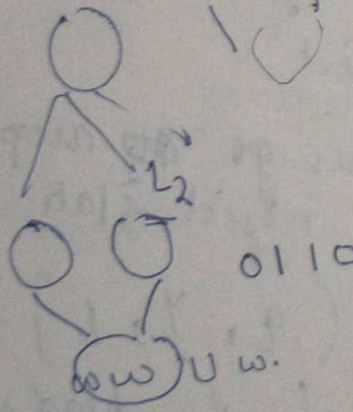


$A \cap B \to$ CFL

$A \cup B \to$ CFL

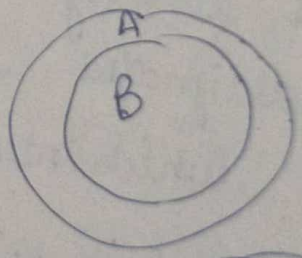
$a^n b^n$

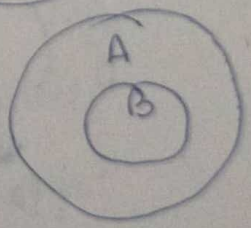




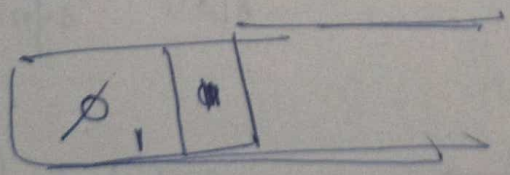
$L_3 - L_1$

$A \cap B \to$





$A \cup B$

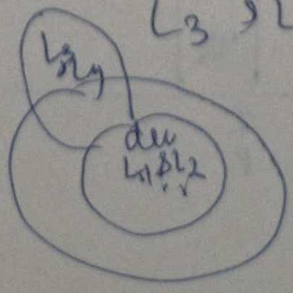


$L_1 \& L_2$ — decidable

$L_3 \& L_4$ — rec

$L_1 - (L_3 \cup L_4)$

$L_3 - L_1$



$L_3 - L_1$

$L_3 - L_1$

$(L_1 \cup L_2) - L_3.$

Turing Machines

Finite Automata :- finite control + no memory

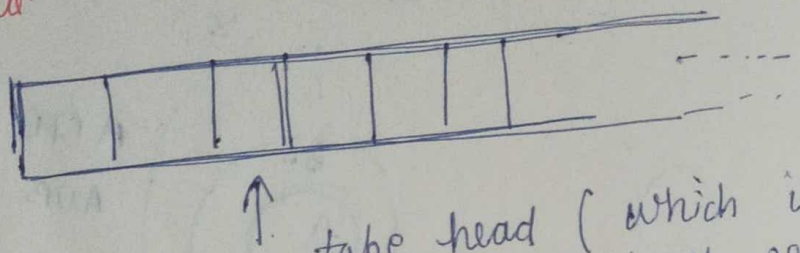
Push Down Automata → finite control + stack

Turing M/C → finite control + ~~tape~~ tape

Tape - infinite data structure

A finite control + tape.

Tape is unbounded and any cell of the tape can be accessed by using a tape head.



↑ tape head (which is capable of pointing any cell on tape)

FA

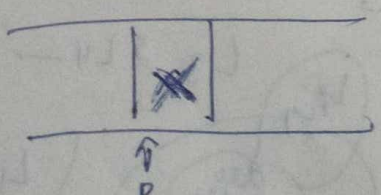
A turing machine has a designated accept and reject

(1) which never halt → neither go to accept or reject state

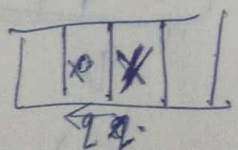
$$\delta(p, X) \vdash (q, Y, L)$$

p → state, X → tape symbol

q → state, Y → tape symbol, L → head moves to left



P



q

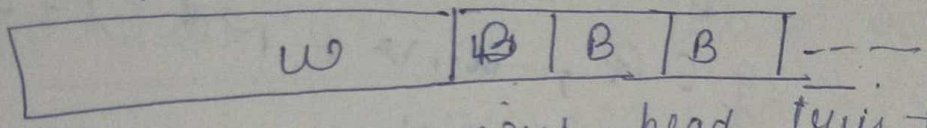
$\delta: Q \times \Gamma \to Q \times \Gamma \times (L, R)$

Input alphabet, Σ is a subset of tape alphabet.

$$\Sigma \subseteq \Gamma$$

There is a blank symbol $\sqcup \in \Gamma - \Sigma$

Initially the tape contains the input w and the remaining cells are filled up with symbol $\sqcup$



If the input head tries to move to the left of leftmost cell of tape then it stays at its current position.

Defn A turing machine (TM) is the tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$

$Q \to$ set of finite states

$\Sigma \to$ Input alphabet.

$\Gamma \to$ tape alphabet

$\delta \to$ transition function.

$\delta: Q \times \Gamma \to Q \times \Gamma \times (L, R)$

$q_0 \to$ initial state

$q_A \to$ accept state

$q_R \to$ reject state.

More on Turing Machines

A configuration of a Turing M/c M with respect to an input w is a snapshot of the machine consisting of

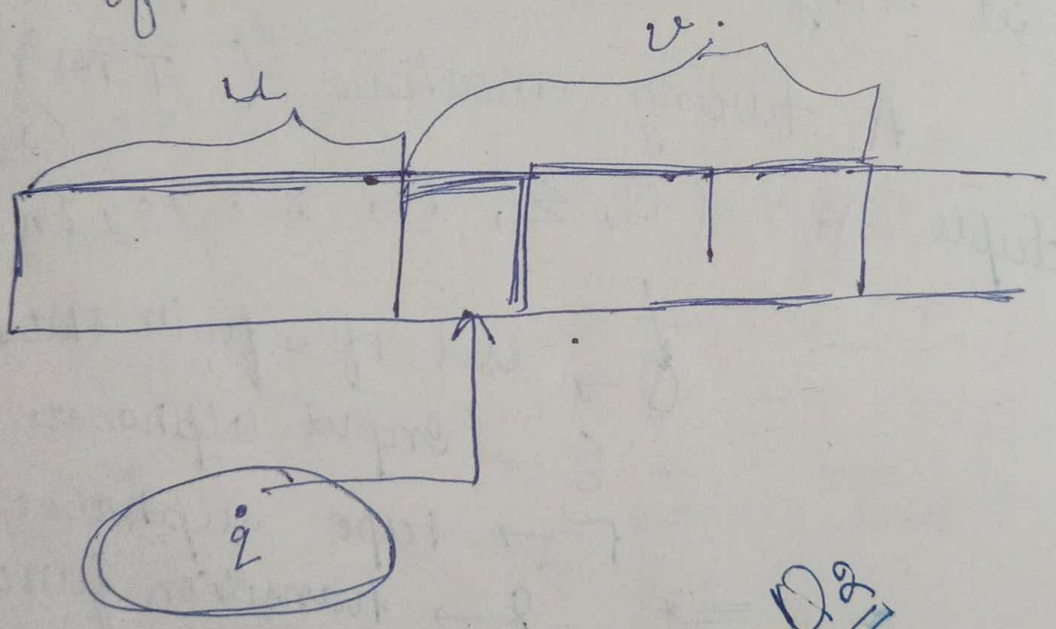
(1) The current state
(2) The current tape contents
(3) The position of tape head.

We represent a configuration

$$u\,q\,v$$

$q \in Q$, $u, v \in \Sigma^*$ s.t.

→ q is the current state
→ string uv is the current contents of tape
→ tape head points to the first symbol of v.



02//

Start configuration

$q_0 w \rightarrow$ where w is input

accept configuration $u q_A v$ where $u, v \in \Gamma^*$

$\rightarrow u q_R v$, $u, v \in \Gamma^*$

$q_A \neq q_R$ are never same.

$a^n b^n$ $a^* b^*$

M halts on w if either M accepts w or M rejects w.

M is said to be a halting turing machine if $\forall w \in \Sigma^*$ M halts on w. (accept or reject)

25.1.

The language of TM, M.

$(12)^* 0 H \rightarrow$

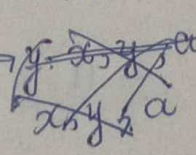
$L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$

A language is Turing recognizable if $\exists$ a Turing Machine s.t $L = L(M)$

A lang is Turing decidable (or just) decidable if $\exists$ a halting TM such that $L = L(M)$.

$(x+y)^* y (a+ab)^*$

0 → 1

1 → 

y, a

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

Description of Turing Machine

checks whether the input is of the form $a^* b^* c^*$. If not then reject.

(2)

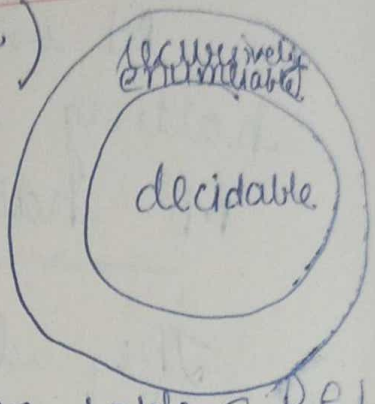
X X X X X X

|

Non-deterministic Turing Machine

halting / decidable (also recursive)

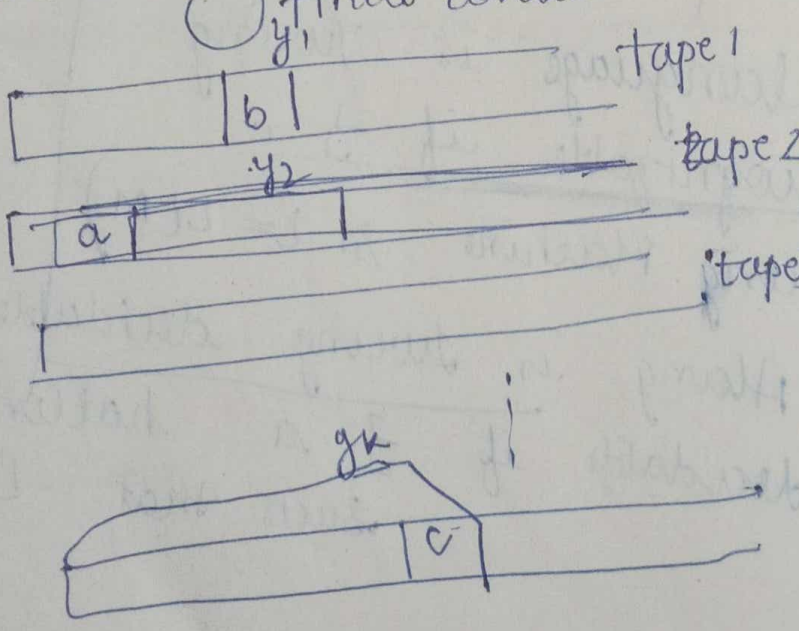
→ Turing recognizable language (recursively enumerable languages)



Decidable ⊆ REL

Variants of Turing Machine

(1) Multi-tape Turing Machine

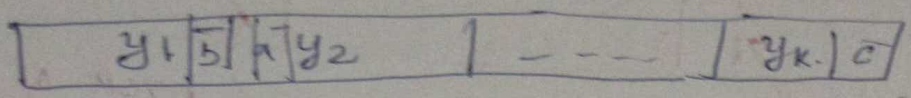


K- tape Turing Machine ≡ 1 tape Turing

Idea We simulate the K- tapes Machine via a single tape

7- tape Turing Machine

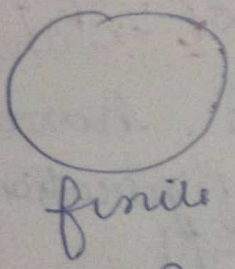
Store the contents of all the tapes.

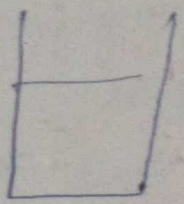


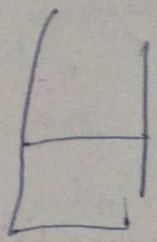
Challenges ① Increase in length of string on a tape.

② Multiple tape heads for every symbol $a \in \Gamma$, add a symbol $\bar{a}$

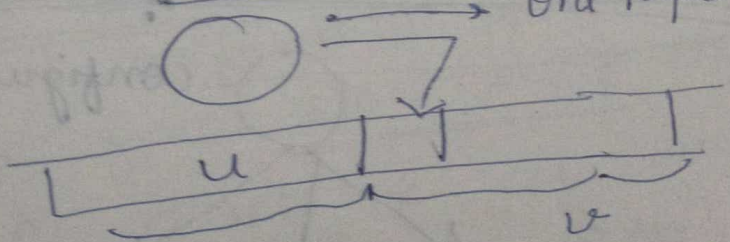
2 Stack Machine


finite control

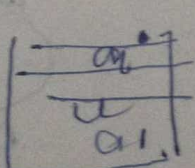

stack 1


stack 2

2 stack machines are subclass of one tape machines



one tape turing m/c.

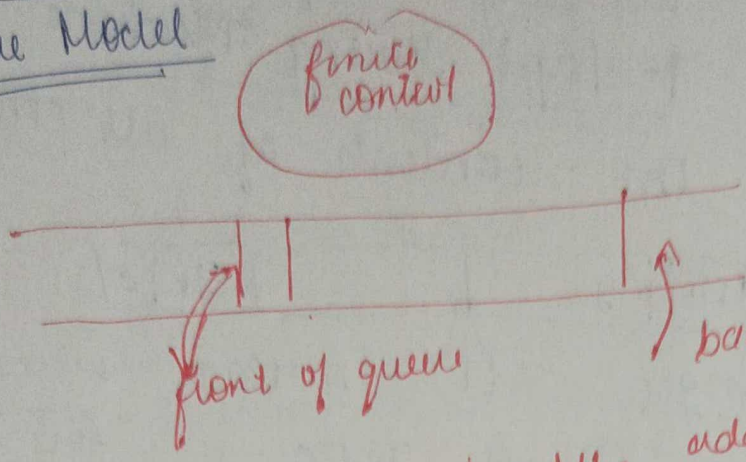


$\in \Gamma$.

2 Stack ≡ TM

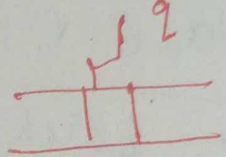
$\frac{25}{100} \times 685$

Queue Model



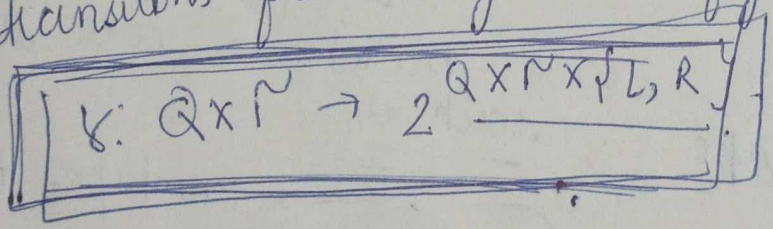
Queue ≡ Turing M/c

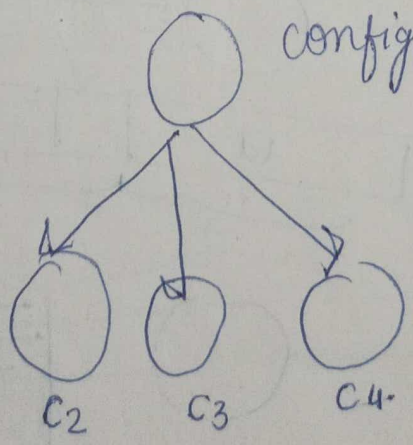
Conf
1. state
2. tape contents
3. tape head.



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

Non-deterministic → one can have mutiple transitions from a given configuration





eg $L = \{ 0^t \mid t$ is a composite number $\}$

Write down n_1 no. of 0_i^s and

n_2 number of 1's.

check if $n_1 \times n_2 = t$

$2 \leq n_1, n_2 < t$

Configuration graphs

A configuration is a tuple of form.

(state, tape contents, position of tape head)

Turing machine M w.r.t an input x

A configuration graph of M on input x (denoted as $G_{M,x}$) is a graph whose vertices are the configuration of G w.r.t x and there is an edge from configuration C_1 to C_2 if the turing machine M can go from C_1 to C_2 in one step

→ graphical representation of the computation of M on x.

A computation path is a path in G_{Mx} from start configuration.

M accepts x if and only if a computation path in G_{Mx} from the start configuration to accept configuration

Properties of configuration graph

configuration graph is defined w.r.t a TM and an input.

(1) → If M is deterministic then out degree of every vertex in $G_{M,x}$ is ≤ 1

(2) If M is non-deterministic then out degree can be arbitrary

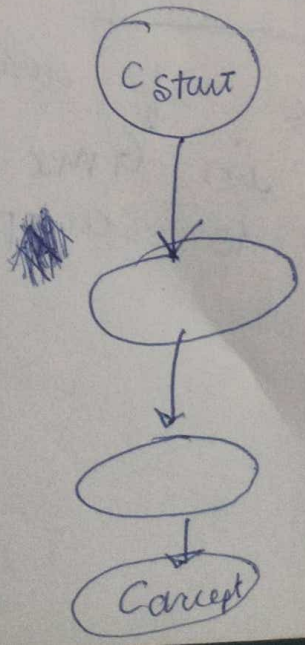
Out degree of accept & reject configuration are zero

Technically $G_{M,x}$ can be infinite but if the size of tape is bounded then $G_{M,x}$ is finite

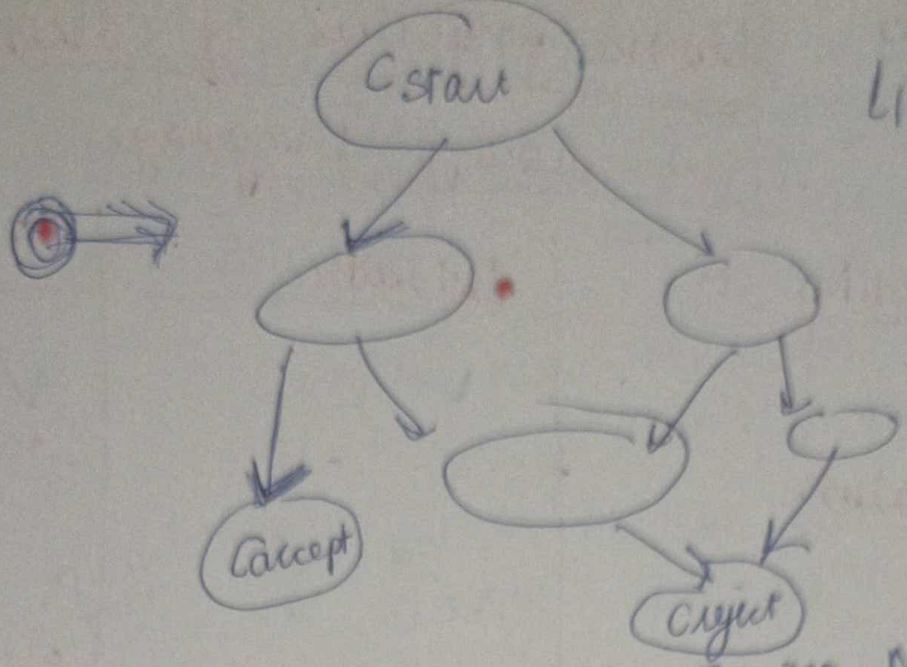
→ Although every configuration is a part of $G_{M,x}$

$G_{M,x}$ has a unique start and accept configuration

Example of configuration graph



Deterministic Turing M1c



L_1 → decidable
L_2 → Tue

L and $\bar{L}$ Thm

M1
M2

Non deterministic Turing Machine

Theorem The class of languages accepted by deterministic and non-deterministic TMs are equal

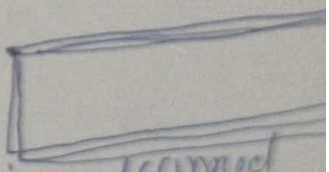
DO a BFS of $G_{M,x}$

Maintain a queue data structure to store the visited configurations of $G_{M,x}$

if an accept configuration is encountered then halt and accept.

→ ~~reject conf is~~

If entire $G_{M,x}$ is scanned without seeing an accept configuration then halt & reject



Bubble Sort

Lec 30 Closure properties of decidable and Turing Languages

Operation	decidable	Turing
(1) Union	✓	✓
(2) concatenation	✓	✓
(3) Star	✓	✓
(4) Intersection	✓	✓
(5) Complement	✓	✗

(1) L_1 & L_2 are decidable via halting turing machines, M_1 & M_2

Yes, $L_1 \cup L_2$ is decidable

M →

Input x

Simulate M_1 on x. If it accept then accept

else simulate M_2 on x,

if M_2 accepts then accept

else reject

$L(M) = L(M_1) + L(M_2)$

Turing Recognizable language

(1) Input x → simultaneously run M_1 & M_2 on input x.

(run M_1 for one step on x, run M_2 for one step on x

(2) If either M_1 or M_2 accepts x, then accept if both reject then reject

L_1 & L_2 are TR →

$L(M) = L(M_1) + L(M_2)$

$L_1 \cup L_2$

L_1 → recursive lang

L_2 → recursive enumerable but not recursive

$\bar{L}$ is recursive

$\bar{L}$ — $L_1 \cap L_2$

L

$L_2 - L_1$ → Turing lang.

$L_1 \cap L_2$: Acceptance problem for DFA

complement problem for DFA

L_1 → decidable

$\bar{L_2}$ → decidable $L_1 - L_2$ →

Decidability →

Acceptability

Equivalence

(1) A_{DFA} is decidable

(2) A NFA is also decidable

(3) A_{RE} is also decidable

(1) $Empty_{DFA} = \{ \langle D \rangle \mid L(D) \text{ is } \emptyset \}$ is also decidable

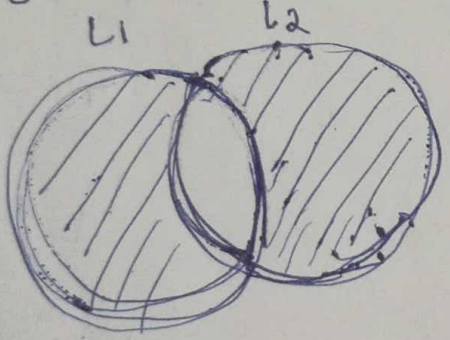
$Empty_{NDFA}$ → decidable

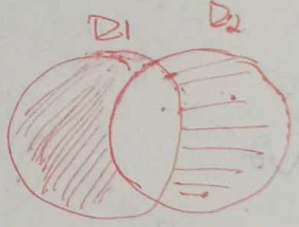
$EQ_{DFA} = \{ \langle D_1, D_2 \rangle \mid D_1 \text{ & } D_2 \text{ are DFA and } L(D_1) = L(D_2) \}$

is also decidable.

$[L(D_1) \setminus L(D_2)] \cup [L(D_2) \setminus L(D_1)] = \emptyset$

$D_1 - D_2$ $D_2 - D_1$





(1) Acceptance of CFG is also decidable.
A_{CFG} → decidable

(2) E_{CFG} → empty $\{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \}$

(3) $EQ_{CFG} = \{ \langle G_1, G_2 \rangle \mid G_1 \text{ and } G_2 \text{ are CFG } L(G_1) = L(G_2) \}$

not decidable.

Equivalence of CFG is undecidable

E_{CFG} → Turing Recognizable

$\overline{E_{CFG}}$ → Turing Recognizable.

False.

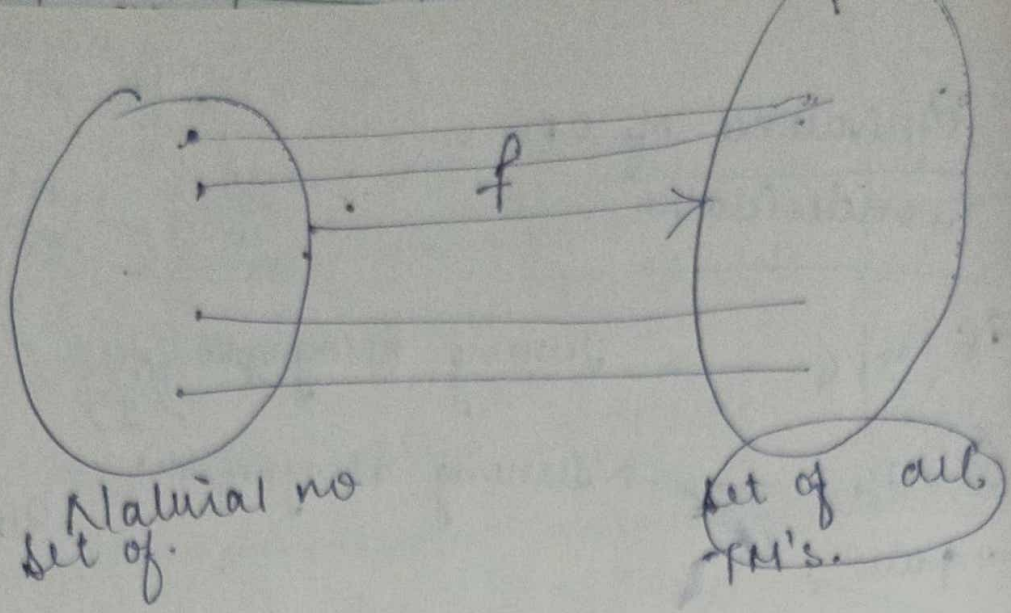
Undecidability → There are languages which can't be computed during Turing machines. These problems are known as undecidable problems.

Church Turing Thesis – Anything that can be computed by a Turing m/c

A Turing Machine can also be represented using a finite string consisting of symbols from some finite alphabet.

Every Turing m/c maps to natural no.

If a natural no does not map to the Turing Machine by earlier representation then we map it to the Turing m/c that accepts all inputs



f is onto function

jth natural number

Let M_j denote the jth TM

Let S_j be the jth binary string

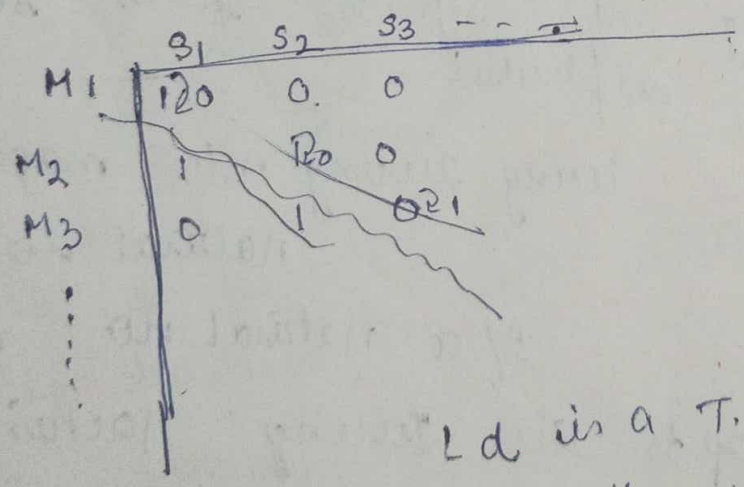
Language

Non-Turing Recognizable Language

Diagonal $L_d = \{ s_j \mid s_j \notin L(M_j) \}$

—jth Turing M:-

$O_{ij} = 1$ if M_i accept s_j
otherwise 0



L_d is a T.M.

$\Rightarrow \exists$ a TM M_i s.t $L_d = L(M_i)$

$s_i \notin L_d$

$s_i \notin L(M_i)$

$\Rightarrow$ M_i does not accept s_i

Mi accepts ⟨i⟩ ..
not Turing Recognizable

An undecidable language

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$

encoding of Turing m/c M ↓ undecidable language ↓ Turing recognizable

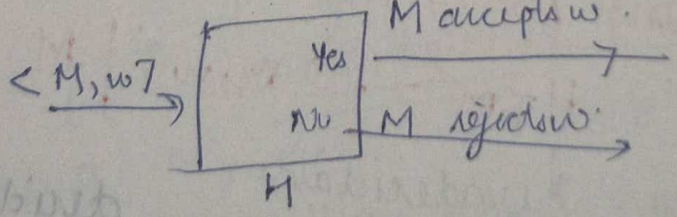
A_{TM} is turing recognizable

If ↓ undecidable language

Undecidability

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$

Suppose A_{TM} is decidable, there exists a Halting Turing M/C H such that $A_{TM} = L(H)$

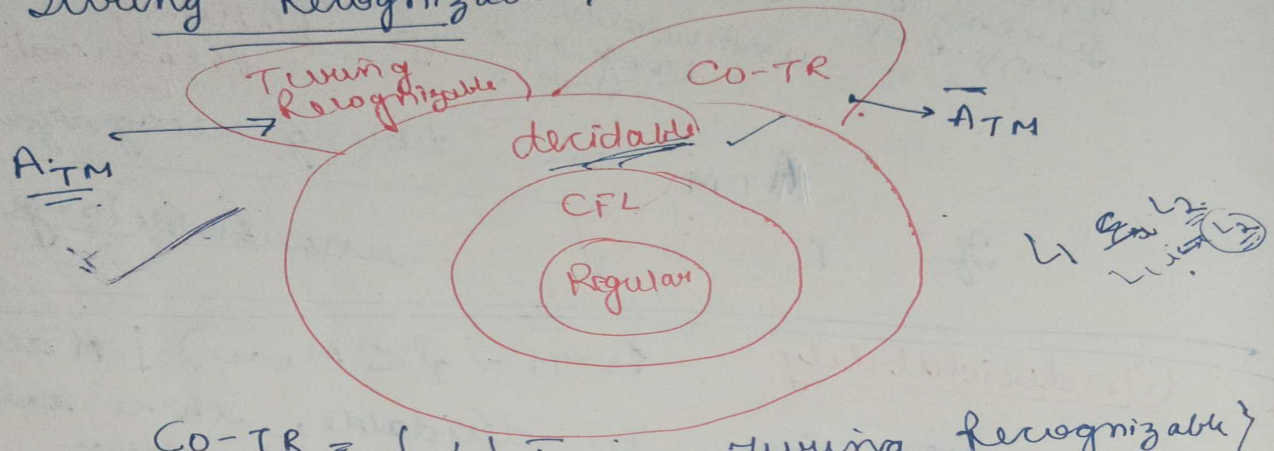


Construct a Turing Machine N as follows.

Input: ⟨M⟩.

$L_1 \subseteq L_2$ ... L_2 regular.

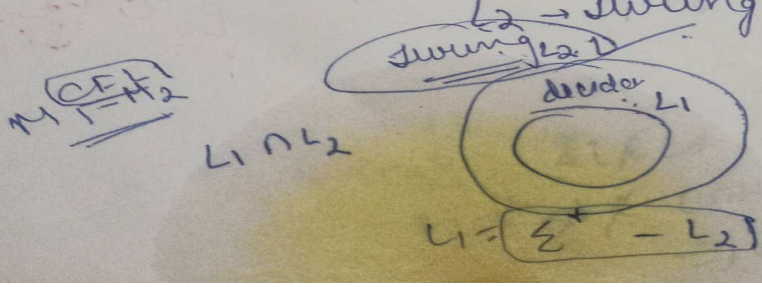
A_{TM} is undecidable
A_{TM} is Turing Recognizable.
$\overline{A_{TM}}$ → not turing recognizable.
$\{\langle M, w\rangle \mid M$ does not accept $w\}$
If L is undecidable and $\bar{L}$ is Turing Recognizable. then L is not T.R.



Co-TR = $\{L \mid \bar{L}$ is Turing Recognizable$\}$

Halting problem ✓ H_{TM} → Turing Recognizable
$H_{TM} = \{\langle M, w\rangle \mid M$ halts on $w\}$
↓ undecidable

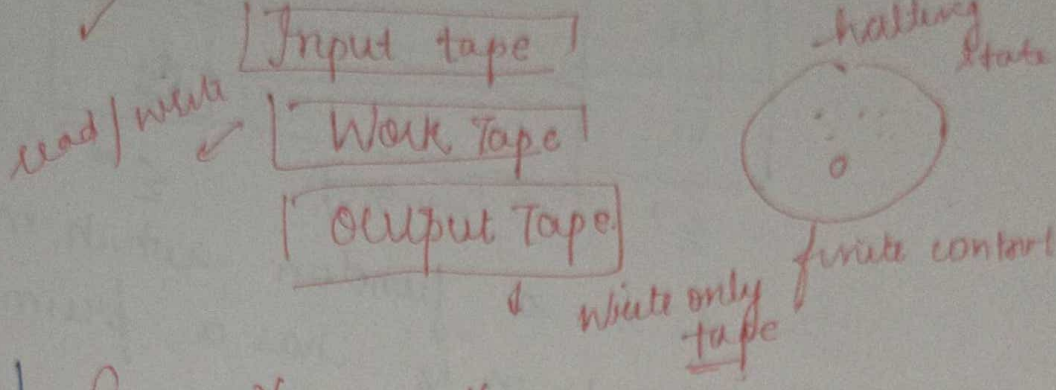
decidable
$M_1 \cap M_2$.
L_1 → decidable
L_2 → Turing Recognizable



$L_1 \cap L_2$
$L_2 - L_1$
$\emptyset \cap \bar{L_2} \rightarrow \bar{L_2}$ →
$L_1 - L_2$.

Reduction → converting one problem into easier problem

can't write



$f: \Sigma^* \to \Sigma^*$

Boolean function

Languages ≡ Boolean functions

$f: \Sigma^* \to \{0, 1\}$

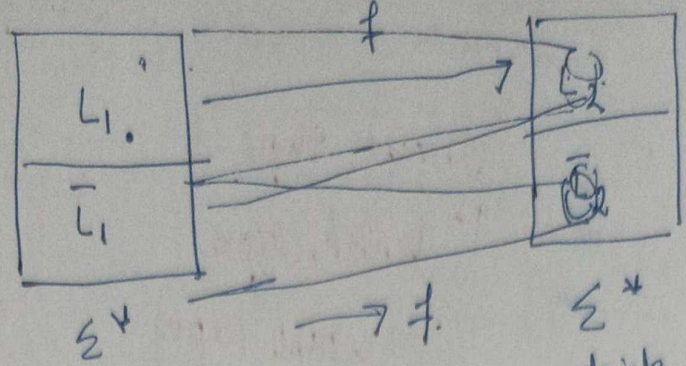
TM with o/p is a TM that has input, work and output tapes as shown s.t for $\forall x \in \Sigma^*$, the Turing m/c computes x. string y ∈ Σ^* before entering the halt state.

→ A function $f: \Sigma^* \to \Sigma^*$ is said to be computable if ∃ a TM with output M, s.t. $\forall x \in \Sigma^*$, M halts with f(x) written on its output tape

Defn $L_1, L_2 \subseteq \Sigma^*$

we say that L_1 reduces to L_2 (denoted as $L_1 \leq_m L_2$) if ∃ a computable function f such that $\forall x \in \Sigma^*$,

$x \in L_1 \iff f(x) \in L_2$

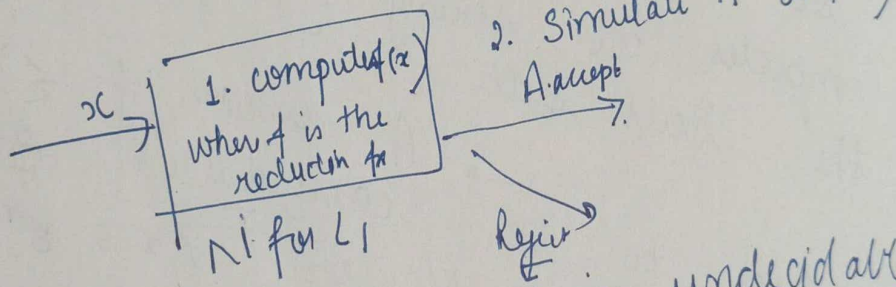


f is a function which means not all element in L_2 or $\bar{L}_2$ has a preimage.

→ many to one $\leq_m$.

Root Property Let $L_1 \leq_m L_2$ and L_2 is decidable then L_1 is also decidable ✓

Root ∃ a halting Turing Machine $L_2 = L(M)$.



① If $L_1 \leq_m L_2$ and L_1 is undecidable then L_2 is also undecidable ✓

② If $L_1 \leq_m L_2$ and L_1 is not Turing Recognizable then L_2 is also not TR. ✓

Key:- $L_1 \leq_m L_2$ then L_2 is at least as hard as L_1

Application.

(1) $E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \phi \}$

encoding is undecidable

We will show that

$A_{TM} \leq_m E_{TM}$.

$L_1 \leq L_2$.

Week 8 Lec

Rice Theorem

undecidability of an infinite set of languages.

A property of language is a f^x

$P = \{$ set of all languages $\} \to \{0, 1\}$

$P(L) = 1$ 1 → language satisfy property

$P(L) = 0$ 0 → " does not satisfy the property

eg of properties

→ L has the string 0110

→ L is empty

→ L has 1000 strings.

Non-trivial property → P is said to be a non-trivial property of languages of TM if

$\exists$ TM_1 and M_2 such that

$P(L(M_1)) = 1$ & $P(L(M_2)) = 0$

Rice's Theorem Let P be a non-trivial property of languages of TM, then the language

$L_P = \{ \langle M \rangle \mid P(L(M)) = 1 \}$

whose language satisfy the property is undecidable.

Proof: Case 1 → ~~$P(\emptyset) = 0$~~

P is a non-trivial property of language of TMs $\exists$ a TM N s.t. $P(L(N)) = 1$.

Using this, we will show that

$A_{TM} \leq_m L_P$

If A_{TM} is undecidable then L_P is undecidable.

The reduction f.

$\langle M, w \rangle \xrightarrow{f} \langle M' \rangle$

Input: $\langle M, w \rangle$

(1) Design a Turing M' that on input x does the following

(M' has a description of N hardcoded into its description together with M & w)

Simulate M on w.

→ If M rejects w then Reject

→ If M accept w then Simulate N on x

If N accept x then Accept. If N rejects then Reject

Output $\langle M' \rangle$. $L(M') = L(N)$

M accept $w \Rightarrow$ $P(L(M')) = 1$

M does not accept $w \Rightarrow L(M') = \phi$.

$\Rightarrow P(L(M')) = 0$.

Therefore $A_{TM} \leq_m L_P$

Case 2: $P(\phi) = 1$

We will "reduce" this to Case 1

Consider the complement property

$\overline{P}$ $(L) = 1 - P(L)$ ← how to reduce M

If $P(L) = 0$
$\overline{P}(L) = 1$.

Since $P(\emptyset) = 1$, $\bar{P}(\emptyset) = 0$.

$A_{TM} \leq_m L_{\bar{P}} \Rightarrow \bar{A}_{TM} \leq_m L_{\bar{P}}$

Observe that

$L_{\bar{P}} = \{ \langle M \rangle \mid P(L(M)) = 0 \}$

$= \{ \langle M \rangle \mid P(L(M)) = 1 \}$.

$\bar{A}_{TM} \leq_m L_P$.

A_{TM} & $\bar{A}_{TM}$ are undecidable

Then L_P is also undecidable.

Applications

$\{ \langle M \rangle \mid L(M)$ is finite $\}$ is regular.

contains the empty is

are undecidable

Complexity Theory

Decidable Languages → we will assume that all our Turing Machines are halting turing machines.

* Defn Let M be a deterministic TM

The running time of M is said to be

$t : N \to N$ if $\forall x \in \Sigma^*$, M halts

in at most $O(t(|x|))$ steps.

The space required by M is

said to be $s : N \to N$ if $\forall x \in \Sigma^*$,

M uses at most $O(s(|x|))$ cells

in its work tape.

We assume the 3-tape model of TM and count the space used in the work tape only

→ we always count the amount of resource used as a function

Since, we use the O-notation we ignore multiplicative constants and lower order terms

Time and space complexity classes

Let $t: \mathbb{N} \to \mathbb{N}$.

Time $TIME(t(n)) = \{L \mid \exists$ a det TM having running time $t(n)$ such $L = L(M)\}$.

$s: \mathbb{N} \to \mathbb{N}$

Space $(SPACE(s(n)) = \{L \mid \exists$ a det TM M requires $s(n)$ space, such that $L = L(M)\}$

Time & space of non-deterministic TMs

Let N be a non-deterministic TM

The running time of N is said to be $t: \mathbb{N} \to \mathbb{N}$ if $\forall x \in \Sigma^*$, every computation path in N halts in at most $O(t(|x|))$ steps

space → at most $O(s(|x|))$ cells

$NTIME(t(x)) = \{ L \mid \exists$ a NDTM N having running time $t(n)$ s.t. $L = L(N) \}$

NSPACE $\{ L \mid \exists$ a NDTM N using $s(n)$ space such that $L = L(N) \}$

Classes P and NP

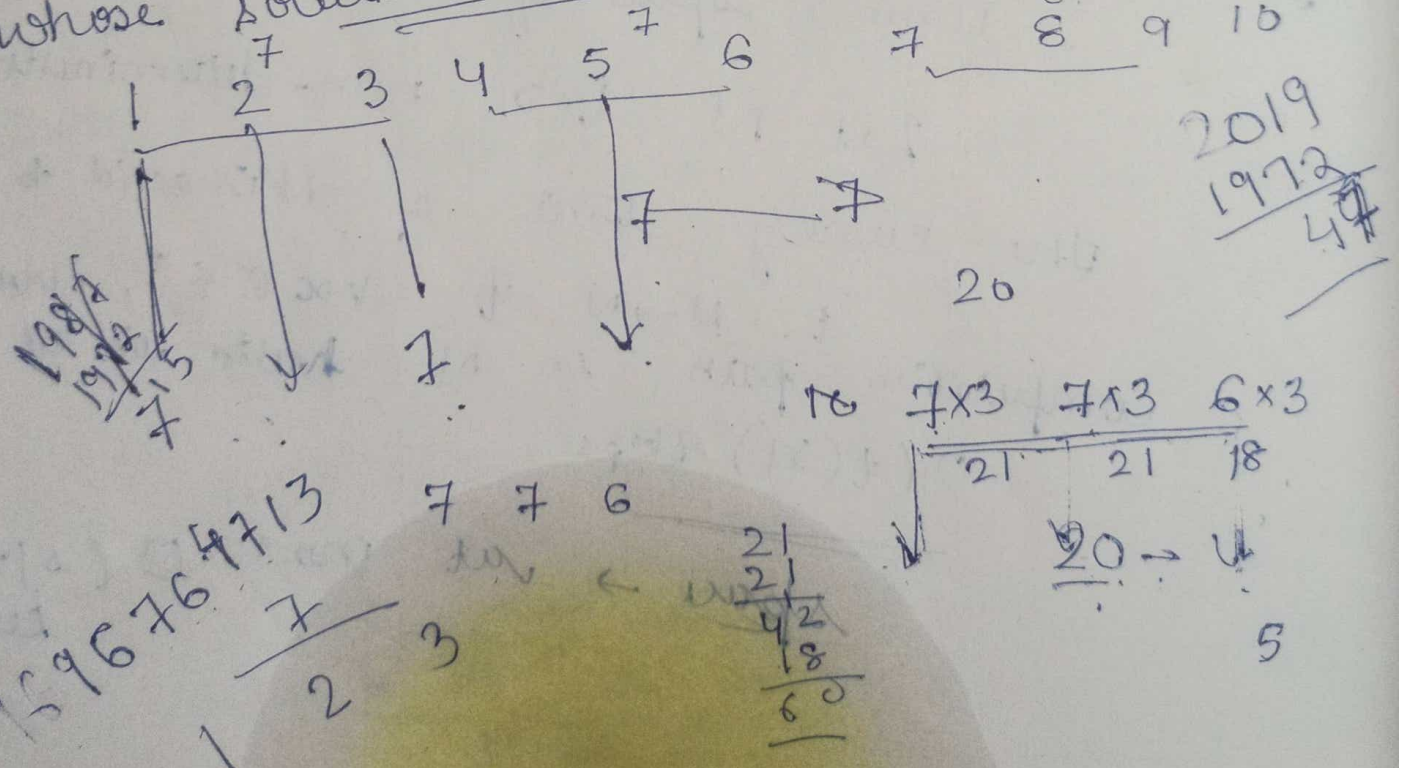
$$P = \bigcup_{k \geq 0} Time(n^k)$$

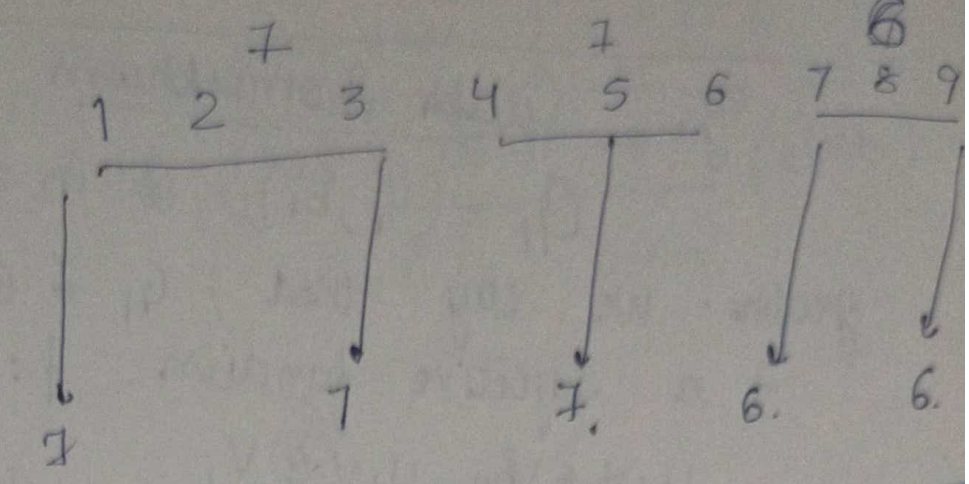
Polynomial eg. $\{ \phi, n, n^2, \ldots \}$

$$NP = \bigcup_{k \geq 0} NTIME(n^k)$$

* P is widely recognized as the class of problems having an efficient soln.

* NP is said to be the class of problems whose solutions can be verified efficiently





More on the class NP.

$$P = \bigcup_{k \geq 0} \text{Time}(n^k)$$

Matrix Multiplication ✓
sorting an array ✓
computing minimum spanning tree
shortest path ✓

$$NP = \bigcup_{k \geq 0} \text{NTime}(n^k)$$

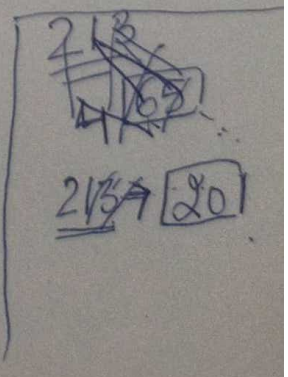
Alternate

$L \subseteq \Sigma^*$ is said to be in NP if $\exists$ constants $c \geq 0$ and $k \geq 0$ and deterministic polytime TM V s.t $\forall x \in \Sigma^*$,

$x \in L \iff \exists y \in \Sigma^*, |y| \leq c x^k$
$V(x,y) = 1$

V (Verifier, deter polytime m/c that takes 2 strings and input x and 20

sky 8 6 7 7 6



y: certificate

eg. Graph Isomorphism

$G_1 = (V_1, E_1)$ & $G_2 = (V_2, E_2)$ be 2 graphs. We say that $G_1 \cong G_2$ if $\exists$ a bijective function $f: V_1 \to V_2$ s.t.

$u, v \in V_1$

$(u, v) \in G_1 \Rightarrow (f(u), f(v)) \in G_2$

non-deterministic

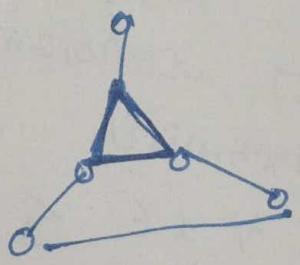
① Graph Isomorphism Problem → NP

$GI = \{ \langle G_1, G_2 \rangle \mid G_1 \cong G_2 \}$

NP algorithm for GI

② Clique problem

$\{ \langle G, k \rangle \mid G$ has a complete graph of size $\geq k \}$

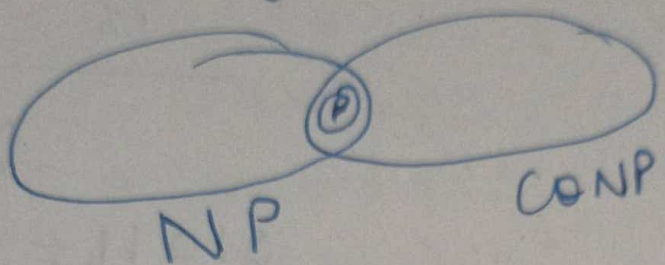


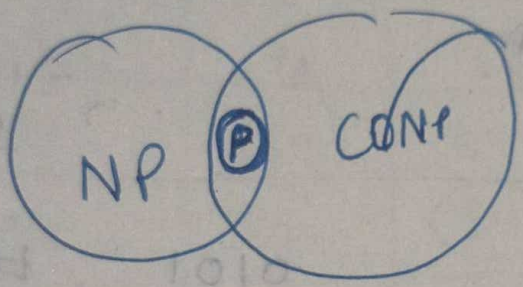
Imp

CO NP (class of problem L such that $L | \bar{L} \in NP$)

Is (NP = CONP → ?) (open)

(NP = P ?) open





P → then NP ∩ CONP

If in P.

NP-completeness → looking at hardest problems in NP.

$\Sigma^* - \{x \in \Sigma^* \mid |x| > 0\}$

$\wedge$ ✓

01 (1 2) · 0101 (1 2 3 4)

00000 (1 2 3 4 5)

01011 (2 3) 3-2=1

GNFA.

→ 101 (4 2 1)

0101 ~~to~~ 011

NFA → DFA. $3m+2$.

DFA → NFA ✓ $3\times0+2=$

$0^*(1(01^*0)^*1)^*0^*0$.

DFA →

10101 (4 2 1) 010 (2 1)

110 (4 2 1)

$q_0 = q_0\wedge + q_00 + \wedge$

$q_1 = q_10 + q_01$

$q_0(0+\wedge)+\wedge$ $q_1 = q_10 + 0^*1$ (RP)

$q_00+\wedge \Rightarrow 0^*$ $q_1 = q_1 0^*10^*$

$(0^*1)0^*$

$S \to AB$

$A \to aa \mid ab \mid ba \mid bb$

$B \to aBa \mid bBb \mid C$

$C \to aa \mid ab \mid ba \mid bb$

$S \to AB$

$\to baBb$

$\Rightarrow babaBbab$

$\Rightarrow babababbab$ ✓

$S \to AB$

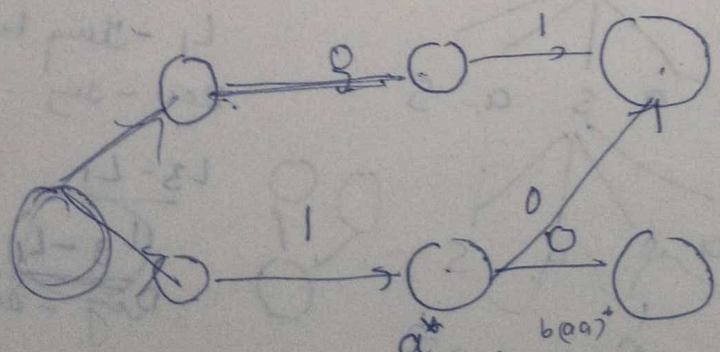
$\to aabb$ ...

	a	b
→q0	q1	q2
q1	q3	q4
q2	q4	q0
q3	q1	q2
q4	q4	q4

$\Pi_0 = \{q_0, q_3\}\ \{q_1, q_2, q_4\}$

$\Pi_1 = \{q_0, q_3\}\ \{q_1\}\ \{q_2\}\ \{q_4\}$

4 States



$S \to aS \mid bT \mid \wedge$

$T \to aaT \mid bT \mid \wedge$

$(aa)^* b^*$

$a^n b^n$

$S \to aSb \mid \wedge$

$S \to aA \mid bB$ (a^*, b^*)

$A \to aA \mid bB \mid \wedge$

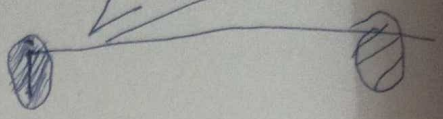
$B \to bB \mid aA \mid \wedge$

b^*

8:30

6:30

2 to 5:30



$S \to aSB \mid aAb$

$A \to aA \mid bA \mid \wedge$.

$L_1 \subseteq L_2$ — Recursive m/c — Decidable

$L_1 \leq_m L_2$ — Turing $\leq L_2$

$L = a^n b^n$ not a.

$L_1 \leq_m L_2$

$L_2 \leq L_3 \Rightarrow$ $\leq_p$ 3 SAT

$L_1 \leq_m$. $L \leq$ X

L is regular →

$L_1 \subseteq L_2$

↓

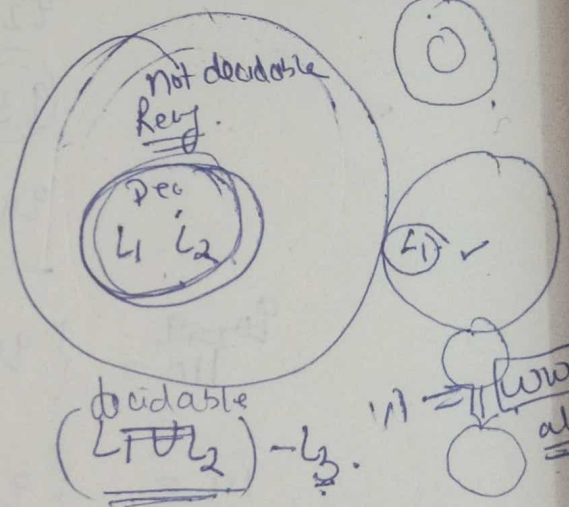
$a^* \not\subseteq$ $a^n b^n \not\subseteq$

$a^2 b^2 \subseteq a^n b^n$ X False

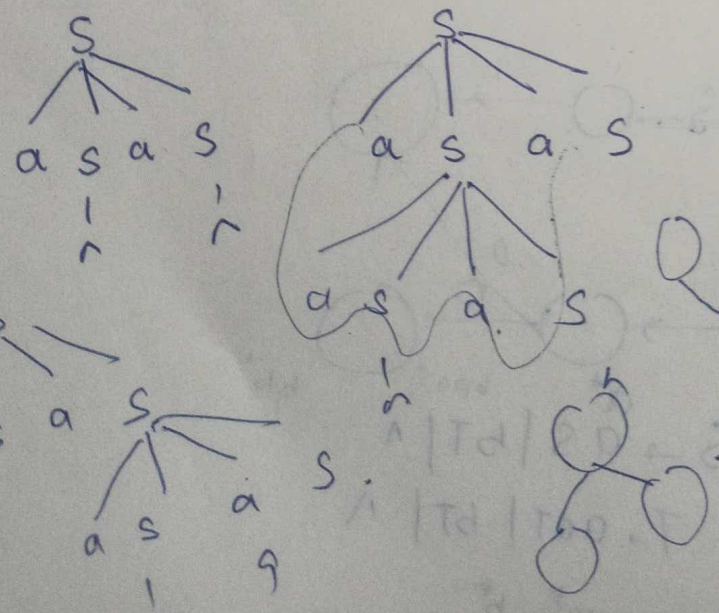
$a^n b^n \subseteq (a+b)^*$ → False.

↓

$a^n b^n \subseteq a$



(3) $S \to aSaS \mid \wedge$



$L_1 - (L_3 \cup L_4)$

L_1 – Turing

Dec – Turing

$L_3 - L_1$

$L_3 - L_1$ → Rec – Decidable

L_4 is Turing

is not

Formal Language

Basic of strings, Alphabets, Formal lang, chomsky classification of lang, Language & their relation, operations on lang, Closure properties of lang. classes

Finite Automata Deterministic FA, Acceptance by Finite Automata, Transition systems Non-Deterministic FA, eq. of DFA and NDFA, Moore & Mealy M/C, eq. of Moore & Mealy, Minimization of F.A, App & limitations of FA

Regular grammar - Regular grammar, Regular expressions, Algebraic method using Arden's Theorem, eq. of FA & Regular expressions, Properties of regular lang, Pumping Lemma.

Context Free Lang

Introduction, Leftmost & Right most Derivation tree, Ambiguity, Simplification of Context Free Grammar, Normal Forms CNF, GNF, Pumping Lemma.

Push Down Automata

Description and defn, Acceptance by PDA, equivalence of PDA & context free grammars & languages

Context sensitive lang

Model of linear bounded automata, Relation between LBA & context sensitive language

Turing Machine Turing machine: def and Model, Representation of Turing Machine. Design of Turing machines

Variants of Turing machines, Recursively enumerable lang, Halting problem, Post correspondence problem, Decidability

M/C,