Lecture - 1

Date: 21.11.21

computation : is any type of calculation that follows an algorithm
: is finding an answer by using an algo.

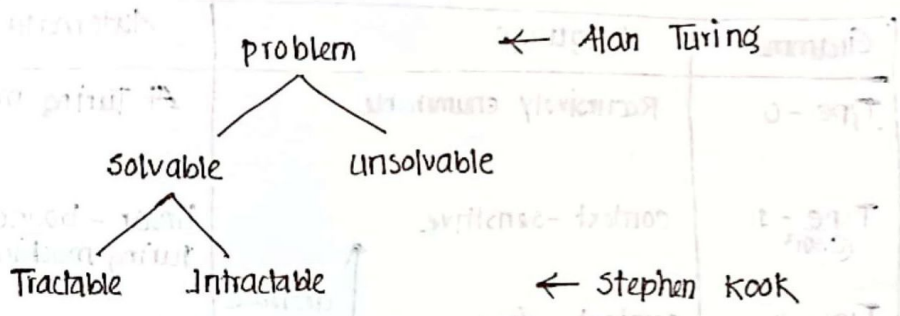
Historical Perspective :

1930 s → "Alan Turing" studies Turing machine

1930-1960 → "Noam chomsky" Finite automata

1969 → "cook" - intractable /NP - Hard problems (time and space)

1970 → Modern computer science compilers, computational and complexity theory



Moore's law: asserts that the speed of computation of a single processor will be double approximately every two years.

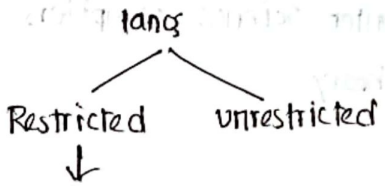
Informally, we can do twice as much work in the same time (or the same amount of work in the half the time).

Applications of Theory of computation :

→ compiler constructing
→ Natural ~~langues~~ language processing
→ Natural lang. understanding
→ Artificial Intelligence
→ Text processing and searching applications
→ cryptography.
→ complex circuit design and so on.

Theory of computation can be divided into 3 main arena.

→ complexity theory (Tractible / Intractible)
→ compability theory (solvable / unsolvable)
→ Automata theory
↳ is the study of abstract machine and the computational problems that can be solved using these machines. These abstract machine are called automata.



Grammer	language	Automata
Type - 0	Recursively enumerable	A Turing machine
Type - 1 (CSG)	context - sensitive	linear - bounded non - deterministic Turing machine
Type - 2 (CFG)	context - free	Push - down automata ; non - deter...
Type - 3 (regular)	Regular (higher restriction)	Finite state ~~att~~ machin

↑ decrease

symbol : smallest unit
: Building block

Alphabet : set of symbols (finite) / collection of symbols

Σ = sigma (capital)

string : sequence of symbols over an alphabet

01, 10, 00, 11

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

$s_1 = 01 \quad s_2 = 10$

$s_1 \cdot s_2 = 01 \cdot 10$
$= 0110$ } string concatination

Σ^P = power of alphabet

$\Sigma^1 \cdot \Sigma^1 = \{0,1\} \times \{0,1\}$
$= \{00, 01, 10, 11\}$

star of alphabet
$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

closure operation
$= \{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$ ← infinite
= set of all possible strings of all lengths over $\{0,1\}$

$\epsilon^0 = \{\epsilon\}$ → epsilon
$\epsilon^1 = \Sigma^1 = \{0,1\}$
$\epsilon^2 = \{00, 01, 10, 11\}$

$\Sigma = \{0,1\}$
$\Sigma^1 = \{0,1\}$ size of string
$\Sigma^2 = \{00, 01, 10, 11\}$

Σ^P = all possible combination of length P

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

cardinality : no. of elements in a set
↳ $\epsilon^n = 2^n$ (0,1)

language : set of collected strings over an alphabet (Σ) with a certain condition.

L = set of all string over alphabet $\{0,1\}$ with length 2.

$= \{00, 01, 10, 11\}$ → finite language

$l_2 = \{00, 01, 10, 11, 0, 1\}$ if length is ≤ 2

$l_3 = \{00, 01, 10, 11, 000, 001, \ldots\}$ infinite lang if (≥ 2)

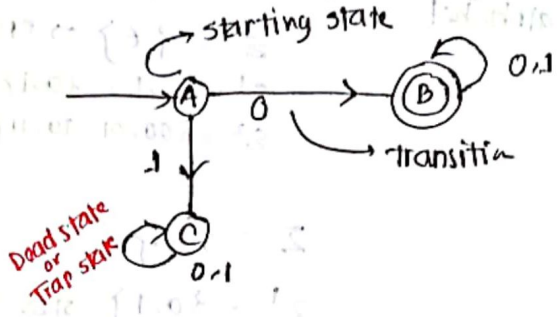
Eg starting with '0' over {0,1}

$L = \{0, 00, 01, 001, 010 \ldots\}$

↓

Abstract machine (FR)

FA accepts the regular language



finite state → accepted state

non-accepted state → A, C

Transitional diagram of FA

0 0,

A → B → B

1 0 0

A → C → C → C (rejected)

$F(L) = \{Q, \Sigma, \delta, q_0, F\}$ [FA mathmatical representation]

Q = ~~Finite~~ set of all states

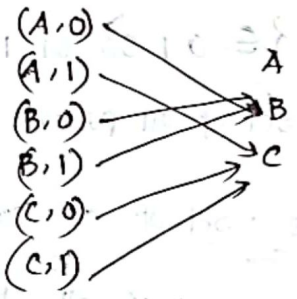
Σ = { , , , symbols / Alphabet }

q_0 = starting state

F = { set of finite → final states }

→ subset of Q $F \subseteq Q$

δ = transition function



$\delta : Q \times \Sigma \longrightarrow Q$

(A,0) → B

(A,1) → C

(B,0) → B

(B,1) → B

(C,0) → C

(C,1) → C

$Q = \{A, B, C\} \times \{0, 1\}$

$\delta : Q \times \Sigma \longmapsto Q$

Representation of finite Automata

→ Tabular representation / Transition table

→ math. model representation

→ pictorial representation

Lecture - 3

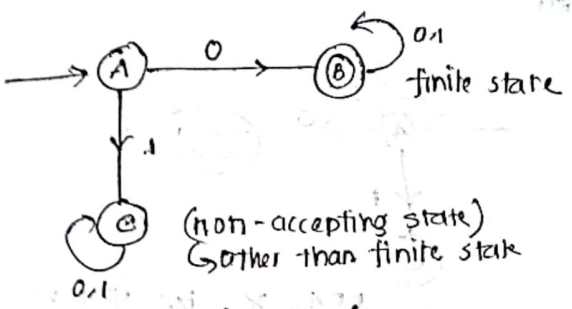
Date: 20.12.21

Transition Table / dig

Tabular representation of FA

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

0 accepted
1 rejected

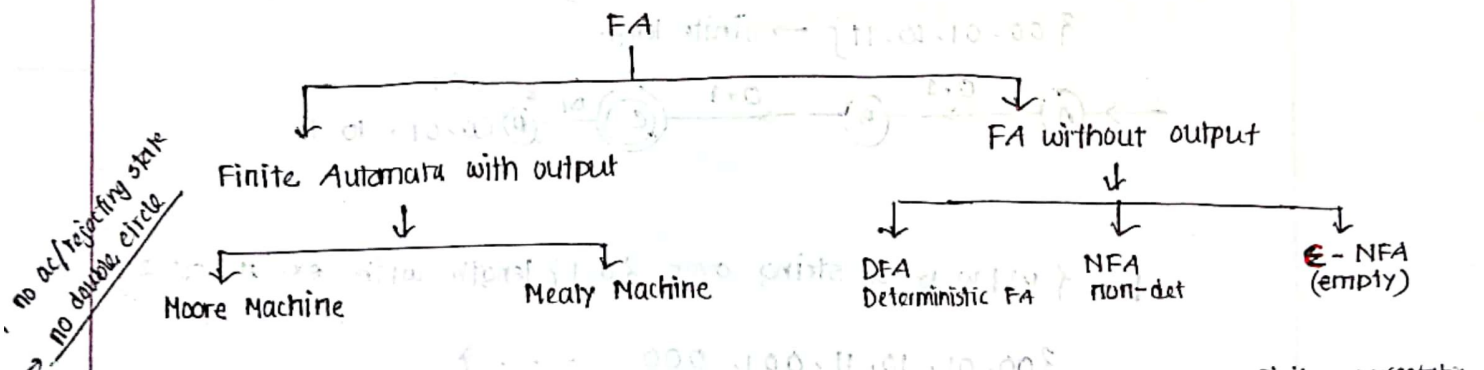


pictorial / transition diagram

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

		0	1
start →	A	B	C
f.s *	B	B	B
	C	C	C

Classification / Types of Finite Automata / Finite state Machine



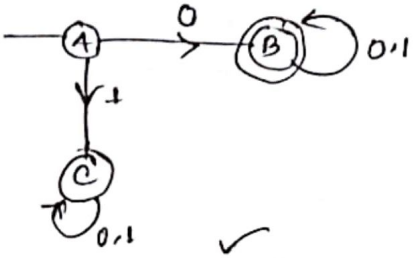
generative device ← lang. generator ← Grammar
lang. acceptor ← Machine
Lang

Finite representation
Regular language
Regular Expression (RE)
↳ represent the regular lang.

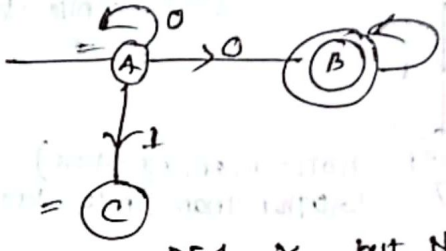
Deterministic Finite Automata (DFA):

→ Simplest model of computation
→ It has a very limited memory

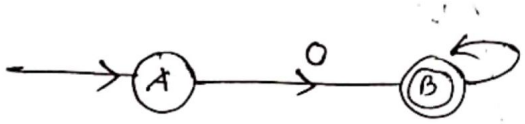
✓(i) There is only one transition for each input
✓(ii) All transitions must be defined for the input alphabet



✓



DFA ✗ but NFA ✓



DFA → ✗

100.1 ✗

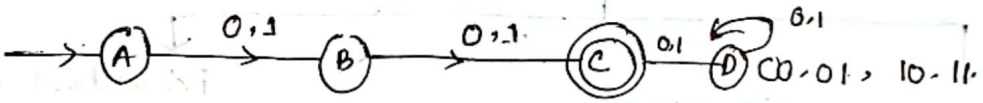
2 conditions (ac)
→ all input scanned
→ at the last stage → Finite state accepting

Σ - NFA
another transition for ε

construction of DFA :

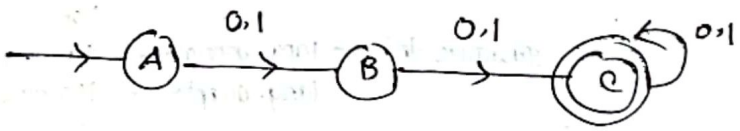
$L = \{ w \mid w \text{ is a string over } \{0,1\} \text{ with exactly 2 length} \}$

$\{00, 01, 10, 11\}$ → finite lang.



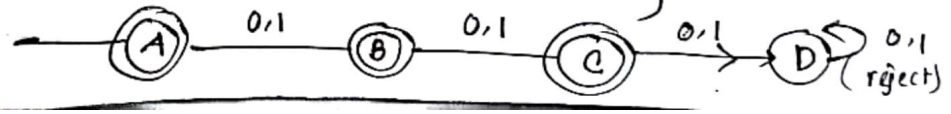
$L = \{ w \mid w \text{ is a string over } \{0,1\} \text{ length with at least 2} \}$

$\{00, 01, 10, 11, 001, 000, \dots\}$



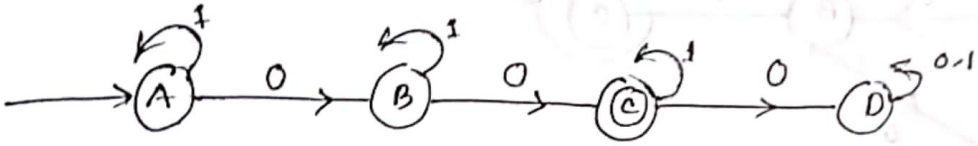
$L = \{ \text{'at most 2'} \}$

$\{ \varepsilon, 0, 1, 00, 01, 10, 11 \}$

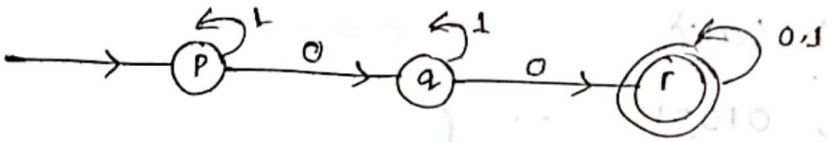


L = { " no. of '0' is exactly 2 }

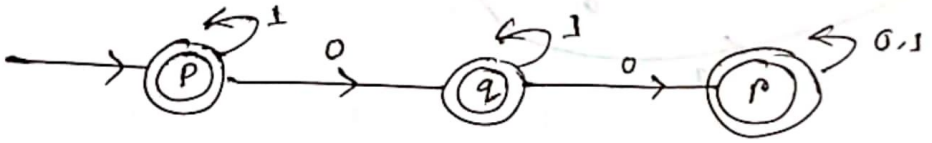
{ 00, 001, 010, 0011, 100, . . . } infinite lang.



L = { no. of 0 at least 2 }



L = { at most 2 }

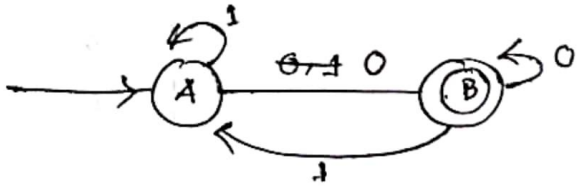


L = { w | w is a string over {0,1} ~~with~~ starting with '0' }

{ 0, 00, 001, 0011, . . . }

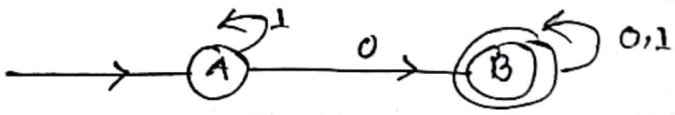
→ ending with '0'.

L = { 0, 00, 10, 010, 110, . . . }



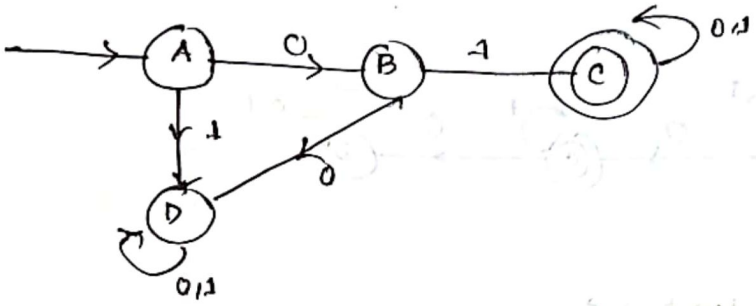
010010
11001

L = { containg '0' }



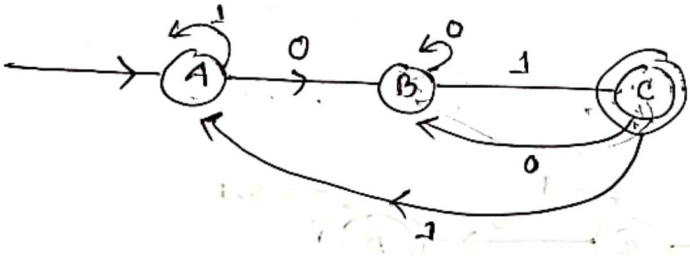
L = {starting with '01'}

{01, 010, 010011, ...}



L = {ending with '01'}

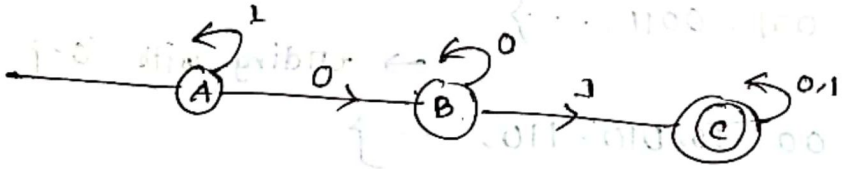
{01, 0101, 01001, ...}



01000110 ✓ Ac

01000110 → reject

L = {containg '01'}



length of string

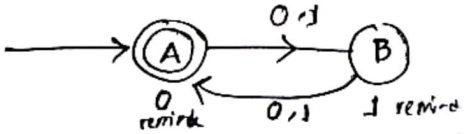
~~Let~~ L = {divided by 2}

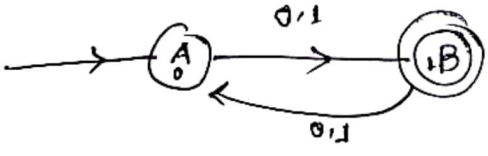
$\frac{0}{2} = 0$

$\frac{1}{2} = 1$

$\frac{2}{2} = 0$

$\frac{3}{2} = 1$





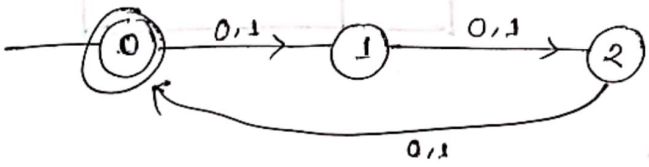
not divided by 2

001 , 01

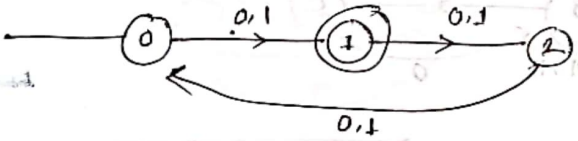
L = { length div by 3 }

{ ε, 000, 010, ... }

Reminder
$\frac{0}{3} = 0$
$\frac{0}{1} = 1$
$\frac{2}{3} = 2$
$\frac{3}{3} = 0$

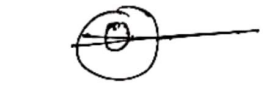


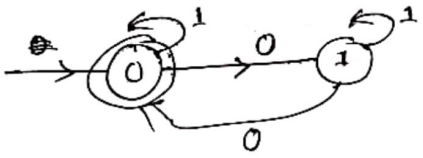
L = not divisible + mod 3 = 1



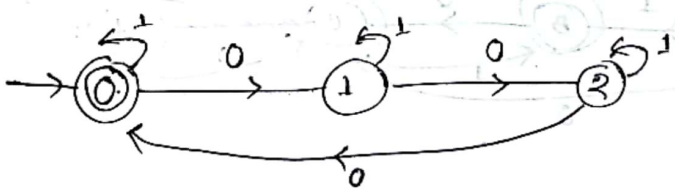
mod 3 = 2,1

L = { no. of '0' divided by 2 }





L = { no. of '0' divisible by 3 }
mod 3 = 0

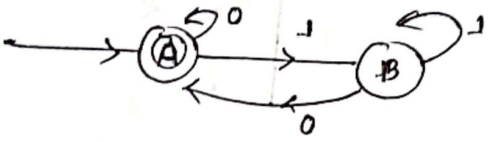


mod 3 = 1, 2

mod 3 = 1

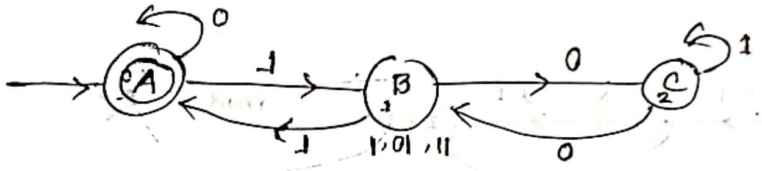
L = { w | w is a string of binary number which decimal equivalent is divisible by 2 mod 2 = 0 }

{ ε, 0, 00 ...



	0	1
→*A	A	B
B	A	B

L = { divisible by 3 }

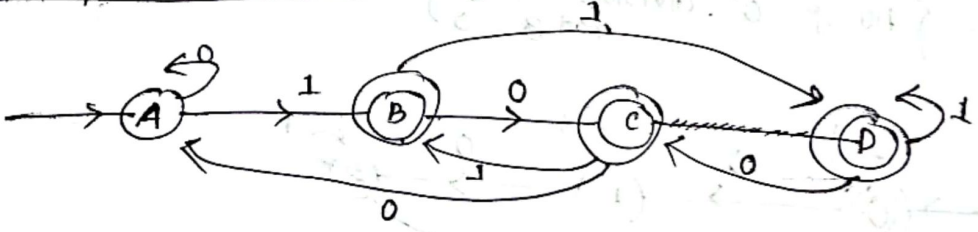


	0	1
→*A	A	B
B	C	A
C	B	C

Binary equivalent of certain

L = { not divisible by 4 }

	0	1
→ A	A	B
* B	C	D
* C	A	B
* D	C	D



Base 3 = {0,1,2}

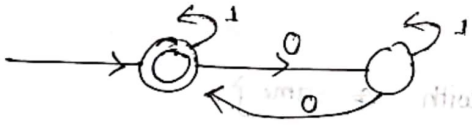
L = { ternary no. (0,1,2)

divisible by 4

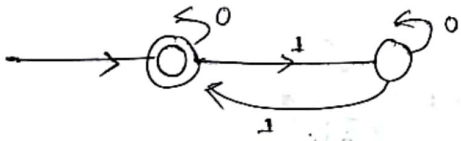
	0	1	2
→ * A	A	B	C
B	D	A	B
C	C	D	A
D	B	C	D

L = { no. of 0 and 1 is even }

{ ε, 00, 11, 0011, 1100, 0101 }



no. of 0 even

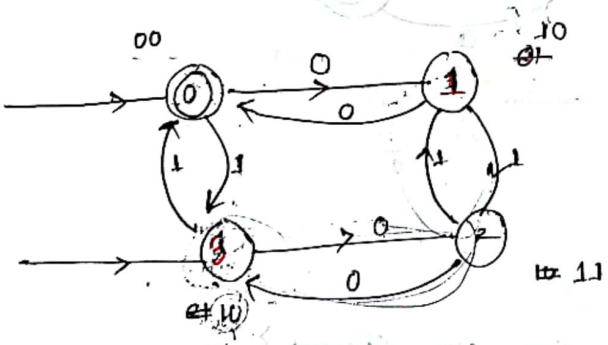


no. of 1 even

no. of 0/1 0 = even, 1 = odd

{0,1}

00
01
10
11



00110

01

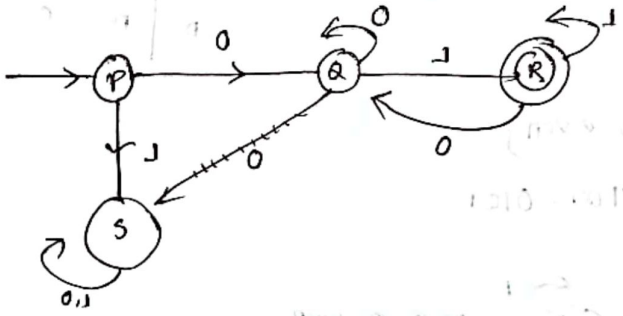
010

0100110 → 3

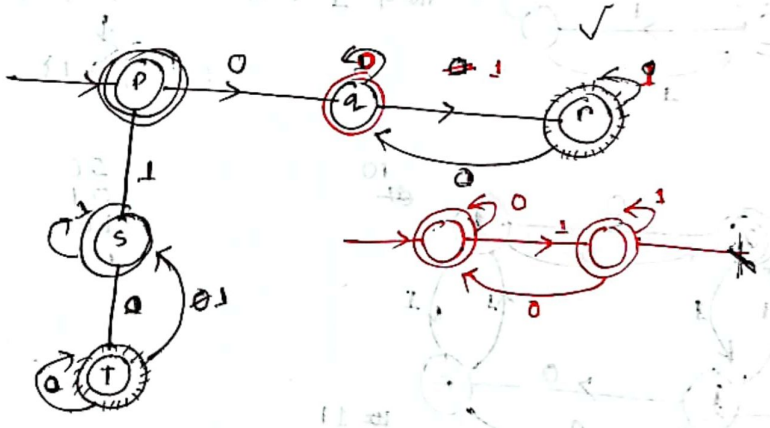
0	1	
X	X	X - even
X	Y	Y - odd
Y	X	
Y	Y	

L = {no. of 0 mod 2 = 1 }
1 mod 2 = 0

L = { starting '0' ending '1' }

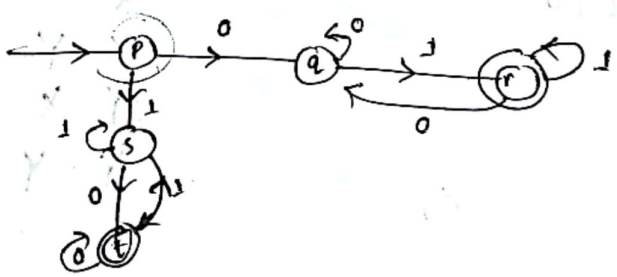


L = { starting and ending with same }
Σ, 0, 1, 0 0
1 1



L = { starting and ending with different }
0 1
1 0

Correction

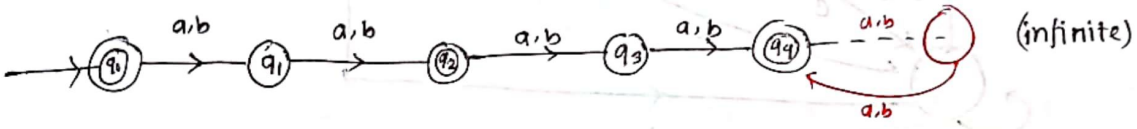


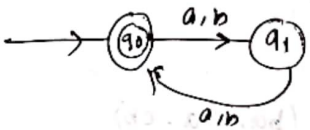
DFA =
→FSA Finite state ~~Machine~~ Acceptor
→FSM " " Machine

Finite { Turing machine, Push-down, Finite Automata

{a,b} even length string

$L = \{\varepsilon, aa, ab, bb, ba, aabb, \ldots\}$



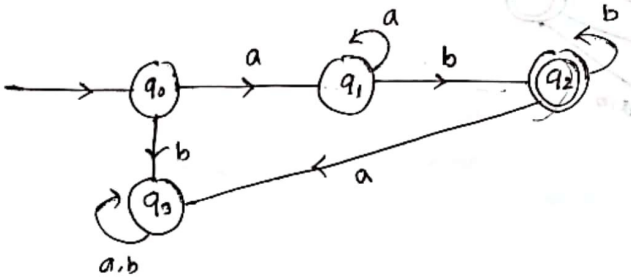


→ Minimal DFA

$L = \{a^m b^n \mid m, n \geq 1\}$

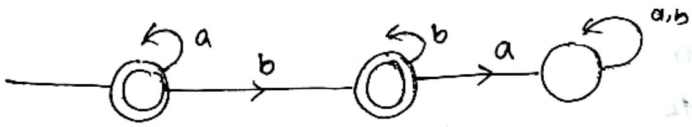
ab, aabb, ...

M(L)



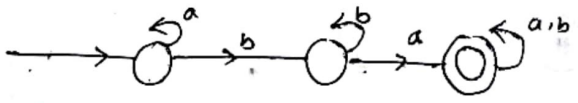
$L = \{a^m b^n \mid m, n \geq 0\}$

$\{\varepsilon, a, b, aaa, bbb, ab, aabb, \ldots\}$



$L = \{w \mid w \text{ is a string of } \Sigma = \{a,b\} \text{ containing 'ba'}\}$

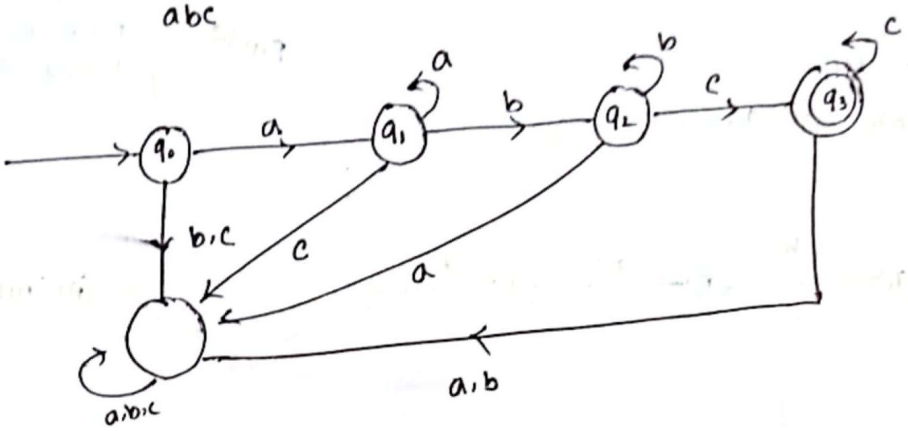
L = containing 'ba'



complementary to each other

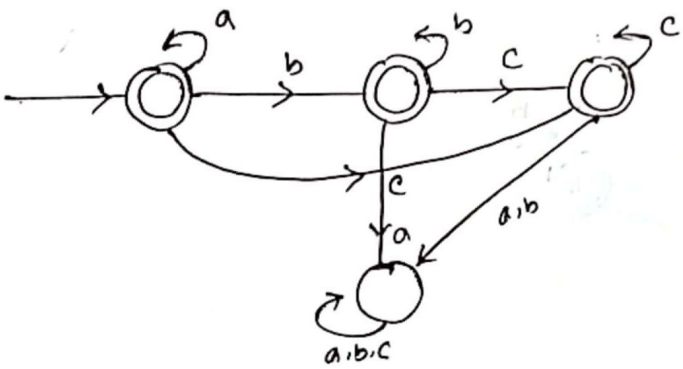
$L = \{ a^m b^n c^l \mid m,n,l \geq 1 \}$

abc



$L = \{ a^m b^n c^l \mid m,n,l \geq 0 \}$ → not containing (ba, ca, cb)

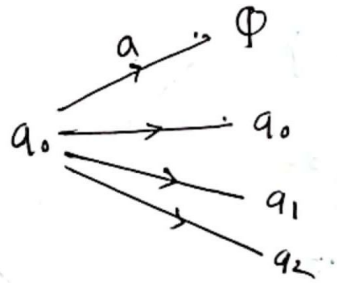
$\{ \Sigma, a, aa, b, bb, c, cc, abc, ab, ac, bc, \ldots \}$



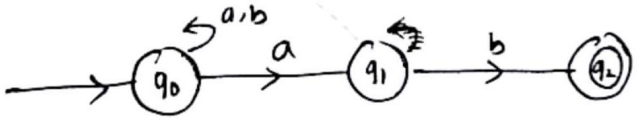
NFA : Non-deterministic FA

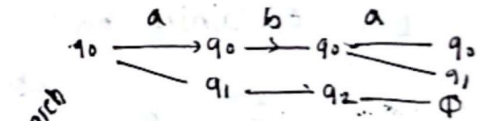
→ no Σ transition

→ no trap state



$L = \{a,b\}^*$ | ending with 'ab' $\}$ $\{ab, aaab, \ldots\}$





search

$q_0 \cup q_1 \cup \Phi$

$\{q_0, q_1\}$

Death configuration

rejected

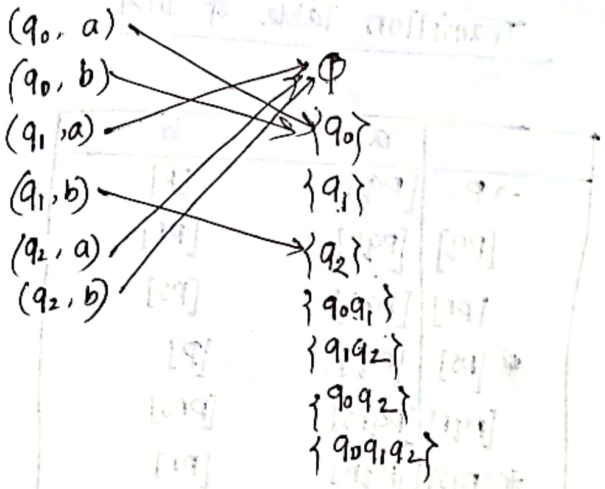
↳ if any state is in finite state then → accepted

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

$Q \rightarrow$
$\Sigma \rightarrow$
$q_0 \rightarrow$
$F \rightarrow$

$\{q_0, q_1, q_2\}$
$\{\}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}$
$2^3 = 8$

$\delta : Q \times \Sigma \longrightarrow 2^Q$ (power set of Q)

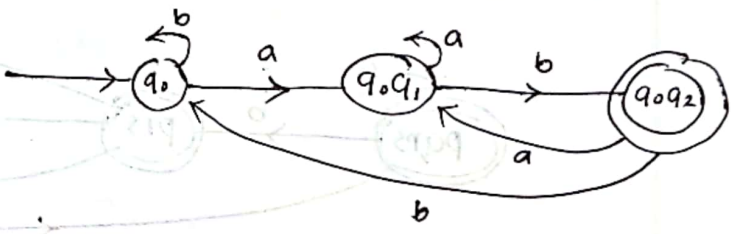


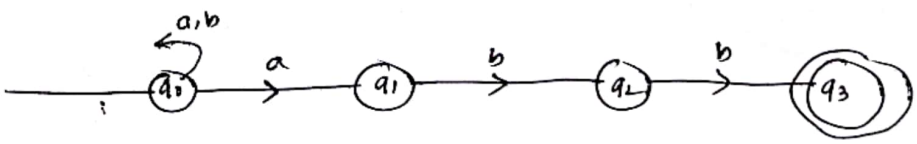
	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	Φ	$\{q_2\}$
* q_2	Φ	Φ

	NFA		DFA	
power :	same	⟷	same	⇄ if and only if
convoision :	✓ subset construction Method		✗ (by default NFA)	

DFA of ending with 'ab'

	a	b
→ q_0	$[q_0 q_1]$	q_0
$[q_0 q_1]$	$[q_0 q_1]$	$[q_0 q_2]$
* $[q_0 q_2]$	$[q_0 q_1]$	$[q_0]$





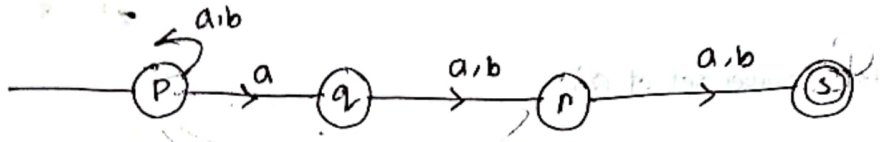
containing 'abb'

Lecture - 5

Date: 26.12.21

Subset construction method

Q to 2^Q

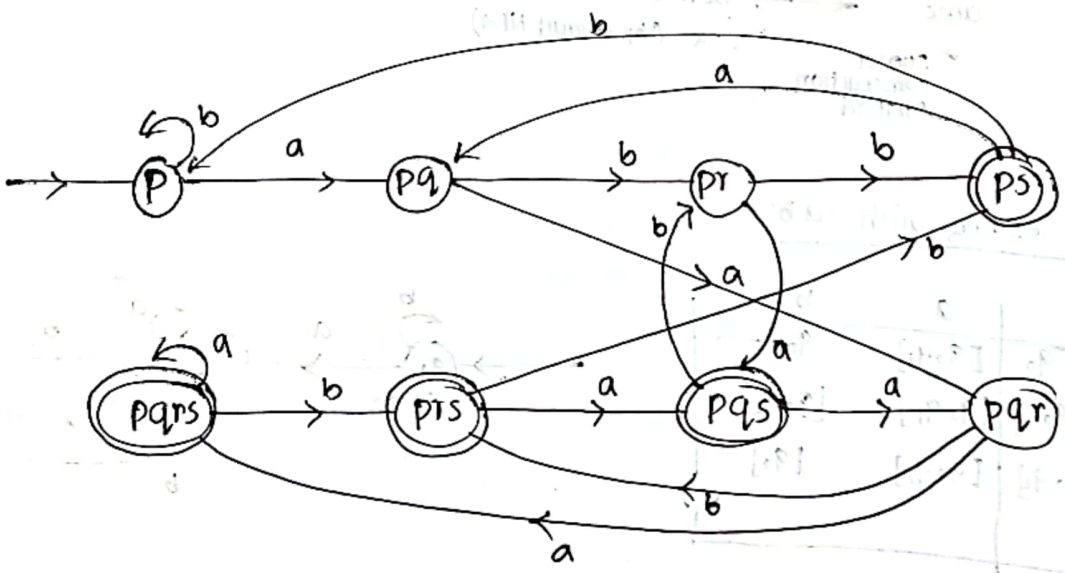


Transitional Table of NFA

	a	b
→p	{p,q}	{p}
q	{r}	{r}
r	{s}	{s}
*s	Φ	Φ

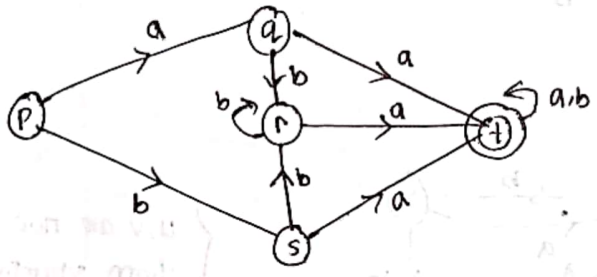
Transition Table of DFA

	a	b
→p	[pq]	[p]
[pq]	[pqr]	[pr]
[pr]	[pqs]	[ps]
*[ps]	[pq]	[p]
[pqr]	[pqrs]	[prs]
*[pqs]	[pqr]	[pr]
*[prs]	[pqs]	[ps]
*[pqrs]	[pqrs]	[prs]



Minimization of DFA

→ partitioning method
→ Table Filling method



Partitioning method

|w| = 0 ; 0 equivalent
|w| = 1 ; 1 "
⋮
|w| = n ; n "

Steps

1 : Remove the element which cannot be reachable from the initial (starting) state

Redundency

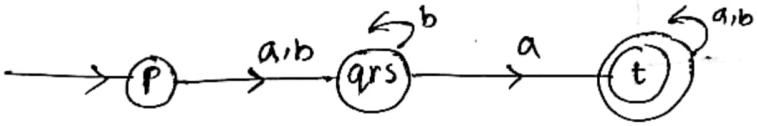
(q,r)

(q,a) → F → (r,a) → F
(q,b) ⇏ F (r,b) ⇏ F

q ⇔ r

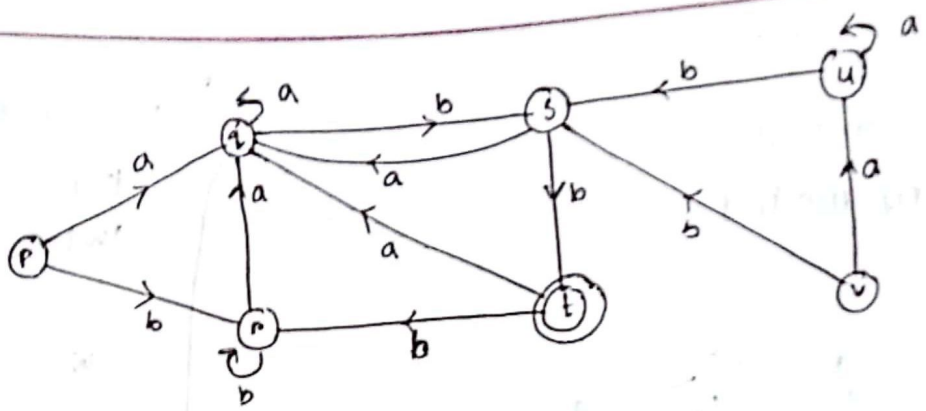
	a	b
→ p	q	s
q	*t	r
r	*t	r
s	*t	r
*t	*t	*t

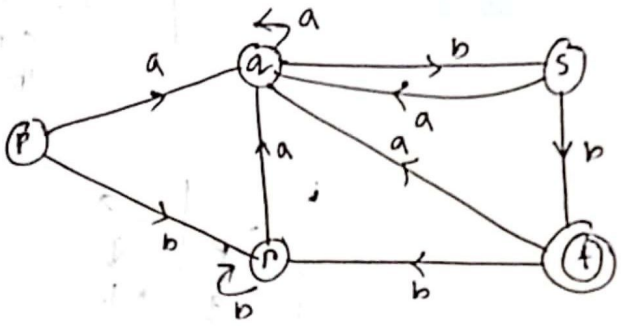
DFA



'0' equivalent : [pqrs] [t]
'1' equivalent : [p] [qrs] [t]
'2' equivalent : [p] [qrs] [t]

← accepting states and non-accepting states are on different partition

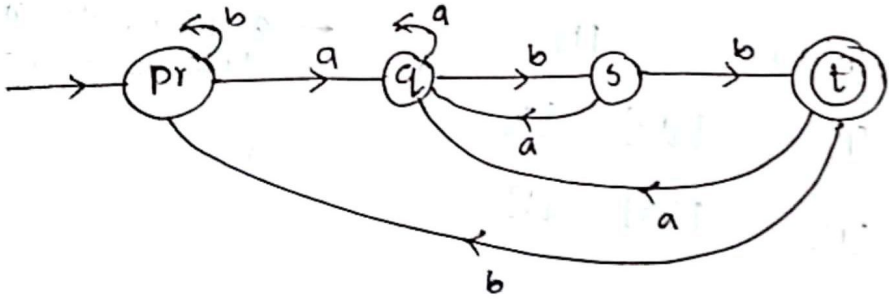




u, v are not reachable from starting state

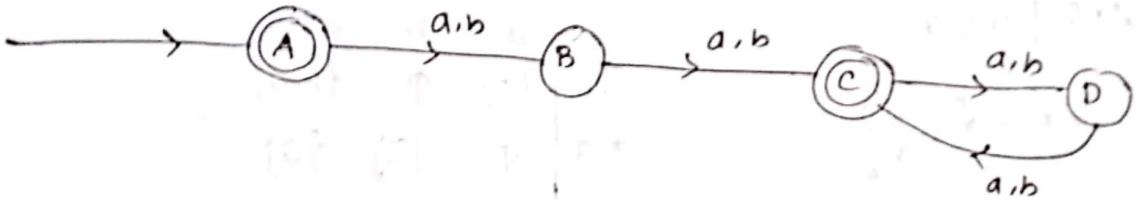
	a	b
→p	q	r
q	q	s
r	q	r
s	q	*t
*t	q	r

'0' equiv : [pqrs] [t]
'1' equiv : [pqr] [s] [t]
'2' equiv : [pr] [q] [s] [t]
'3' equiv : [pr] [q] [s] [t]



Minimal DFA

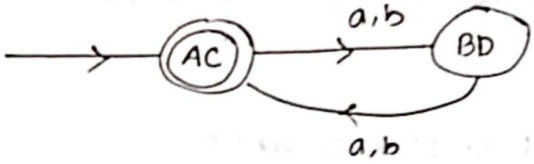
String with even length



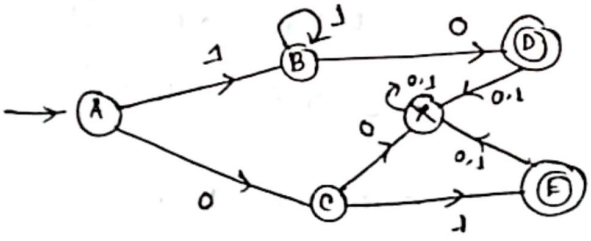
	a	b
→*A	B	B
B	*C	*C
*C	D	D
D	*C	*C

'0' equival : [AC] [BD]

'1' equiv : [AC] [BD]



×



10 ✓

$\frac{1}{A}\ \frac{10}{B}$ ✓

01 ✓

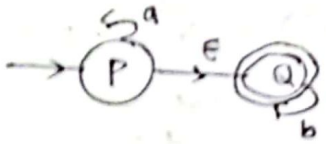
01

{ at least one '1' followed by '0'

ϵ-NFA : NFA with ϵ-transition

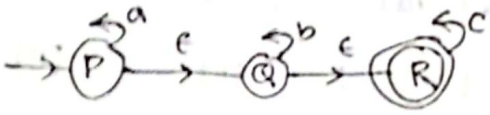
$$\delta : Q \times [\Sigma \cup \{\epsilon\}] \to 2^Q$$

→ $a^n b^n \mid n \geq 0$



	a	b	ϵ
→P	{P}	Φ	{P,Q}
*Q	Φ	{Q}	{Q}

→ $a^n b^n c^n \mid n \geq 0$



- Atleast one final stage to be accepted

P → P →a P →b P →a Φ

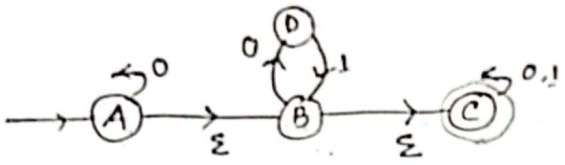
Q →b q →a q →b Φ

r →c r →a r →b Φ

Φ ∪ Φ ∪ Φ = Φ

rejected

ϵNFA to NFA conversion:



	0	1	Σ
→A	{A}	Φ	{B,A}
B	{D}	Φ	{B,C}
*C	{C}	{C}	{C}
D	Φ	{B}	{D}

∴ Σ-NFA transition table

Σ-closure (A) : Σ*

A → A (0 no. of Σ)

A → B (1 no. of Σ)

A → C (2 no. of Σ)

Σ-closure (B)

B → B

B → C

..... (C) = {C}

..... (D) = {D}

corresponding NFA T. table :

	0	1
→*A	{ABCD}	{C}
*B	{C,D}	{C}
*C	{C}	{C}
D	{ }	{B,C}

Σ-closure (Σ-closure (δ, a)) state - a, input - a

Σ → state → Σ

A → A → A

A → B → D → D

A → C → C → C

ABCD

Σ^* — 0 — Σ^*

B → B → {D} → D
→ C → C → C

Σ^* — 0 — Σ^*

A → A, B, C
→ B → D
→ C → C

Σ^* — 0, 1 — Σ^*

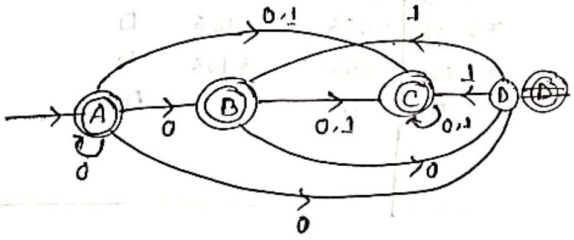
A → Φ
→ B → Φ
→ C → C → C

Σ^* — 1 — Σ^*

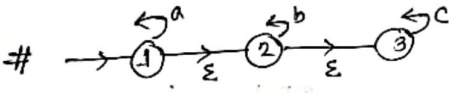
B → B → Φ
→ C → C → C

Σ^* — 0 — Σ^*

C → C → {C} → C



{All Σ-move to go to final state will be final state – NFA

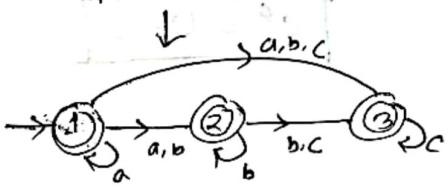


convert to NFA

	a	b	c
→*1	{1,2,3}	{2,3}	{3}
*2	Φ	{2,3}	{3}
*3	Φ	Φ	{3}

←

	a	b	c	Σ
→ 1	{1}	Φ	Φ	{1,2}
2	Φ	{2}	Φ	{2,3}
*3	Φ	Φ	{3}	{3}



0 → Σ^* → 0 → Σ^*

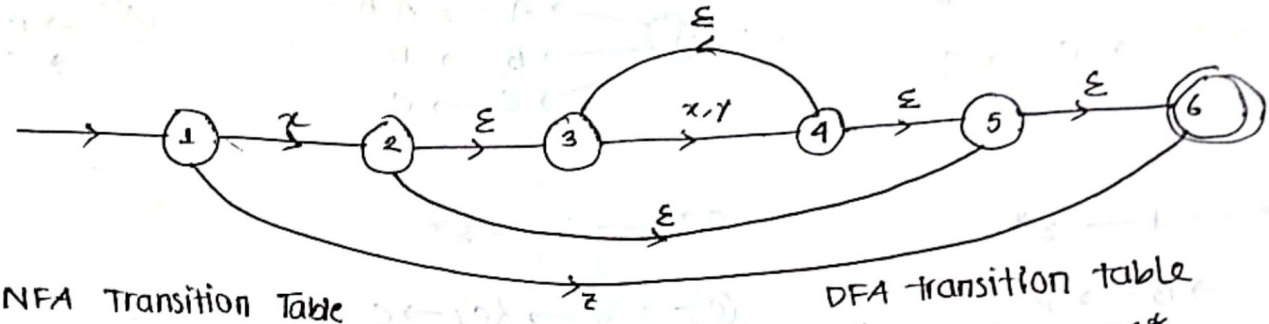
1 → 1 → 1 → {1, 2, 3}
→ 2 → Φ
→ 3 → Φ

↳ for state 'a'

1 → Σ^* → state a → Σ^*

1 → 1 → 1 → 1, 2, 3
→ 2 → Φ
→ 3 → Φ

ε-NFA to DFA

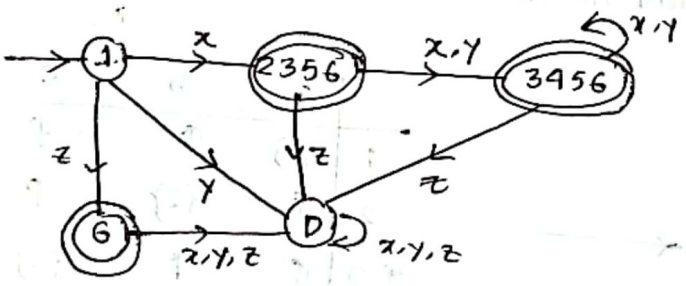


ε-NFA Transition Table

δ	x	y	z	ε*
→ 1	{2}	Φ	6	{1}
2	Φ	Φ	Φ	{2, 3, 5, 6}
3	{4}	{4}	Φ	{3}
4	Φ	Φ	Φ	{3, 4, 5, 6}
5	Φ	Φ	Φ	{5, 6}
* 6	Φ	Φ	Φ	{6}

DFA transition table

$\hat{\delta}$	$x\varepsilon^*$	$y\varepsilon^*$	$z\varepsilon^*$
→ 1	[2356]	D	[6]
* 6	D	D	D
* 2356	3456	3456	D
* 3456	3456	3456	D
D	D	D	D



DFA
NFA
ε-NFA
} Equal power

operation of Finite Automata (s)

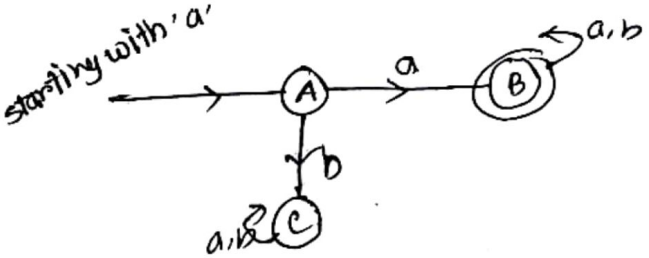
1) complement of FA
2) union of two finite Automata
3) intersection of two finite FA

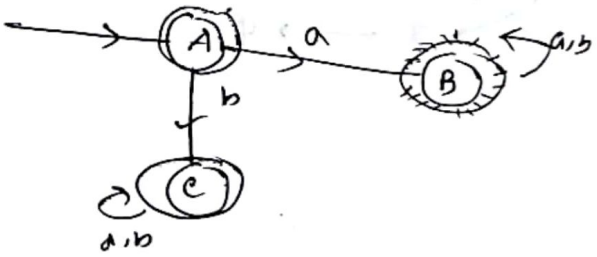
$$C^c = U - C$$

complement

L = {w | w is a string over {a, b} starting with 'a'} {a, aa, ab, ...}

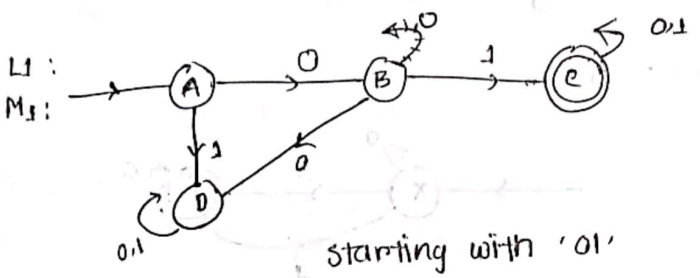
L^c = { ———— not starting with 'a'} {ε, b, a, bb, bba, ...}



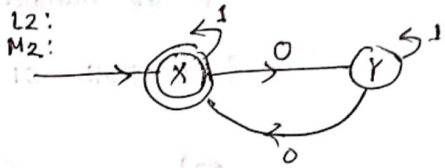


Question Intersection

$L = \{ w \mid w$ is a ~~language~~ string starting with '01' and ~~the length of~~ w is even number of '0' $\}$



starting with '01'



number of '0' is even

M1 :

$Q_1 = \{A, B, C, D\}$

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

$q_{01} = A$

$F_1 = \{C\}$

M2 :

$Q_2 = \{X, Y\}$

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

$q_{02} = X$

$F = \{X\}$

⇒ **M**

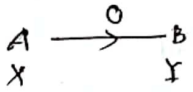
$Q_M = Q_1 \times Q_2$

$= \{AX, AY, BX, BY, CX, CY, DX, DY\}$

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

$q_0 = AX$

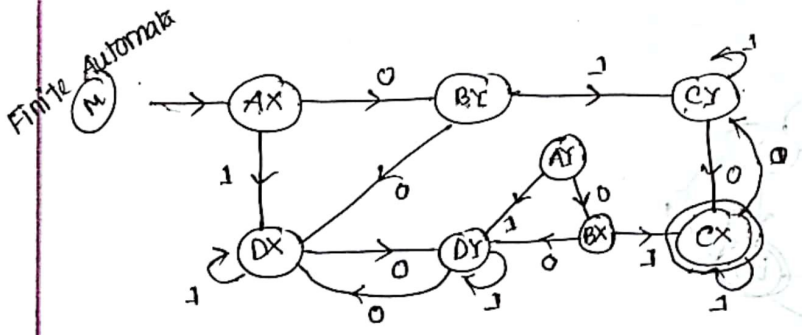
$F = \{CX\}$



Transition Table

	0	1
→ AX	BY	DX
AY	BX	DY
BX	DY	CX
BY	DX	CY
* CX	CY	CX
CY	CX	CY
DX	DY	DX
DY	DX	DY

Finite Automata



← Not minimized

AY

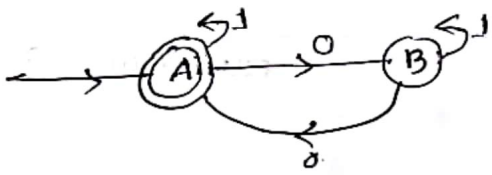
01 0010 ✓

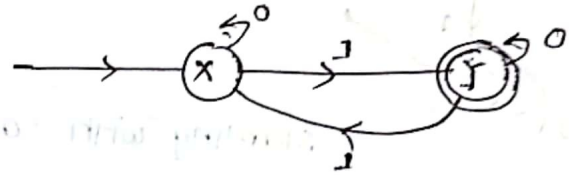
01 010 ✗

Union

$L = \{ w \mid w$ is a string of $\{0,1\}$ in which the number of 0's is even or the number 1's is odd $\}$

$= \{ 00, 0011, 01, 1, \ldots \}$





$Q_1 = \{A, B\}$

$\Sigma_1 = \{0,1\}$

$q_{01} = A$

$F_1 = \{A\}$

$Q_2 = \{X, Y\}$

$\Sigma_2 = \{0,1\}$

$q_{02} = X$

$F_2 = \{Y\}$

$M_1 \cup M_2 = M \Rightarrow$

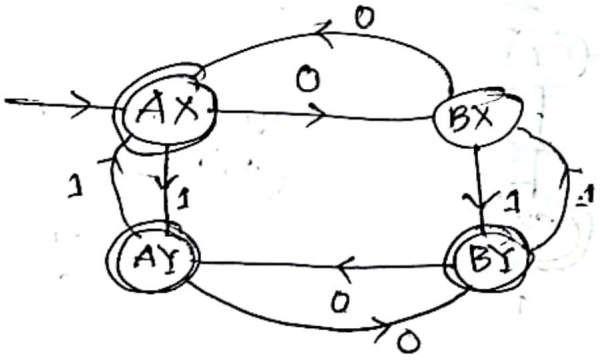
$Q = \{AX, AY, BX, BY\}$

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

$q_0 = AX$

$F = \{AX, AY, BY\}$

	0	1
→*AX	BX	AY
* AY	BY	AX
BX	AX	BY
* BY	AY	BX



FA (with output)

- Moore Machine
- Mealy Machine

For both

$M = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$

Q = Finite set of states

Σ = Finite set of input symbols (Alphabet)

δ = set of input transition function

$\delta : Q \times \Sigma \to (Q)$ → Deterministic

q_0 = Initial state

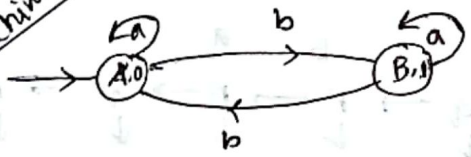
Δ = Finite set of output symbols

λ = set of output Transition function

$\lambda : Q \to \Delta$ (Moore machine)

$\lambda : Q \times \Sigma \to \Delta$ (Mealy machine)

Moore machine



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

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

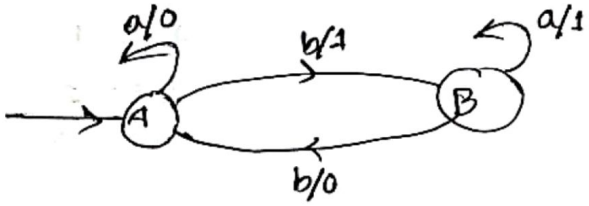
$(A, a) \to A$

$A \to 0$

	a	b	a	a	b
	A →	A →	B →	B →	B → A
	↓	↓	↓	↓	↓
	0	0	1	1	1 0

← output string (length 6)

length + 1 = length of ou



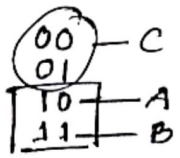
	a	b	a	a	b
	A →	A →	B →	B →	B → a
	↓	↓	↓	↓	↓
	0	1	1	1	0

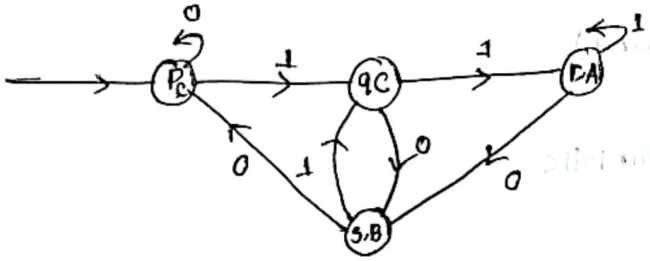
← length 5

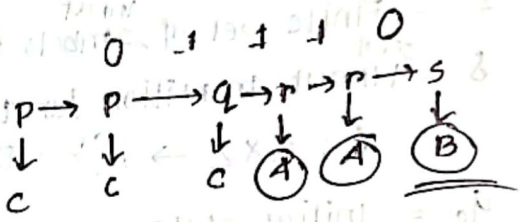
Question: Construct a moore machine that takes strings over {0,1} as input produce as output A if the input string ending with '11'
B if the input string ending with '10'

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

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



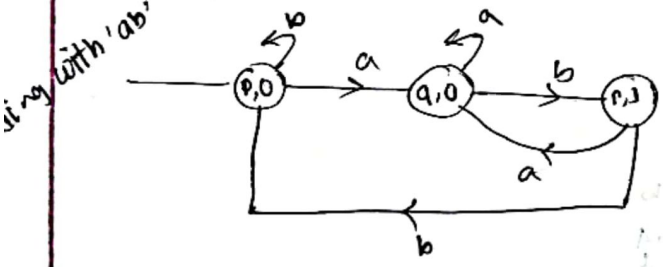


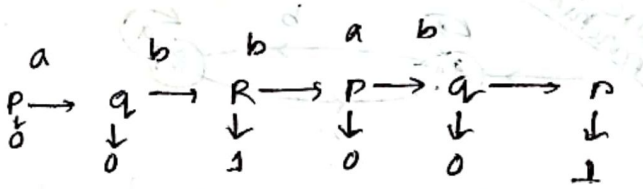


Question: construct a Moore Machine that takes strings over {a,b}. count the no. of occurance of 'ab' as substring

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

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





1+1=2 no. of occurance.

Question: construct a Moore Machine that takes strings of base-4 number produce O/P ~~Mode~~ residue Mod-3.

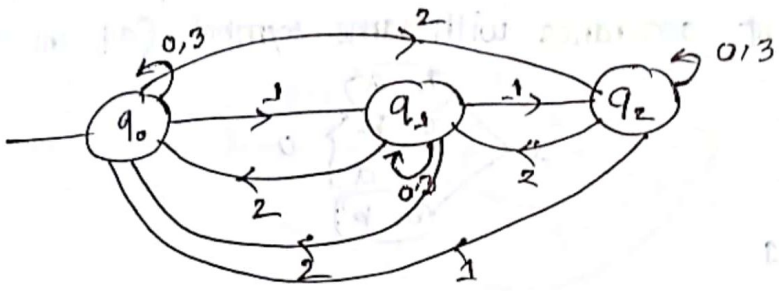
$\Sigma = \{0,1,2,3\}$

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

~~Remainder~~

0/3 = 0
1/3 = 1
2/3 = 2
3/3 = 0

	0	1	2	3
→ q_0	q_0	q_1	q_2	q_0
q_1	q_1	q_2	q_0	q_1
q_2	q_2	q_0	q_1	q_2



Question: Construct a Mealy Machine that takes input ~~string with {0,1}~~ binary as input and produce 2's complement as O/P

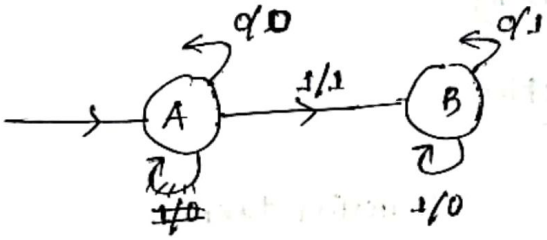
$\Sigma = \{0,1\}$

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

1	1	0	0
↓	↓	↓	↓
0	0	1	1
		+	1
0	1	0	0 (2's)

1	0	1	0	0
0	1	0	1	1
				+
0	1	1	0	0

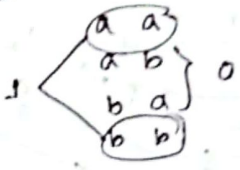
0	1	1	0
1	0	0	1
		+	1
1	0	1	0



A ~~used~~ ← B ← A ← A ← A

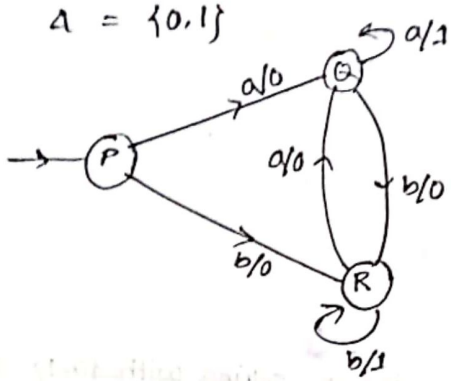
0 1 1 0 0

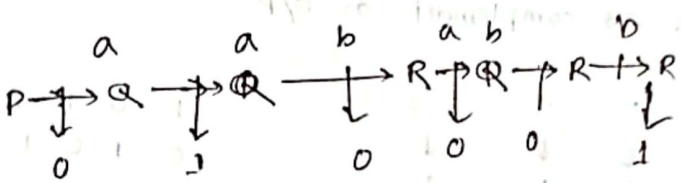
question: construct a Mealy Machine that takes string over {a,b} as input. produces the no. of occurance with same symbol (adjacent symbol)



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

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





Lecture-9

Date: 16.1.22

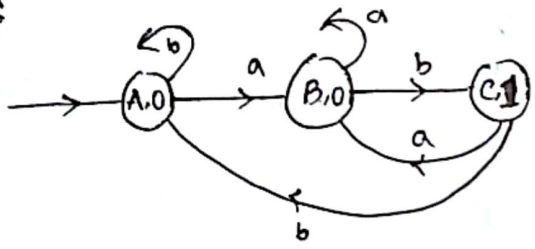
Moore / Mealy

→ ~~pow~~ equally powerful → equivalent (conversion)

Conversion ~~to~~ Moore Machine to Mealy machine

Moore Machine

occurance of 'ab'



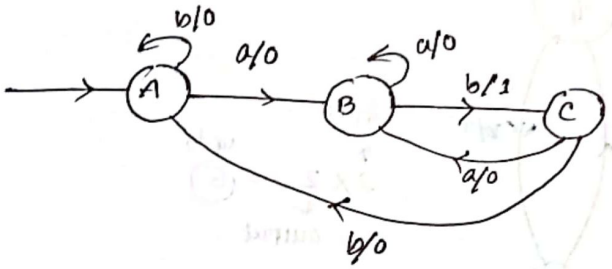
Transition table

	a	b	Δ (output)
→ A	B	A	0
B	B	C	0
C	B	A	1

no. of transition is always greater than Mealy Machine

straight forward

Mealy Machine

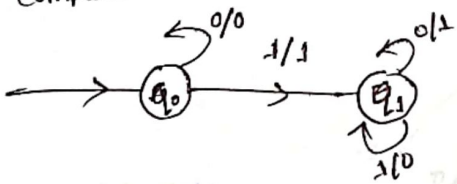


	a	b	Δ
→ A	(B, 0)	(A, 0)	0
B	(B, 0)	(C, 1)	0
C	(B, 0)	(A, 0)	1

conversion from Mealy Machine to Moore Machine :

2's complement :

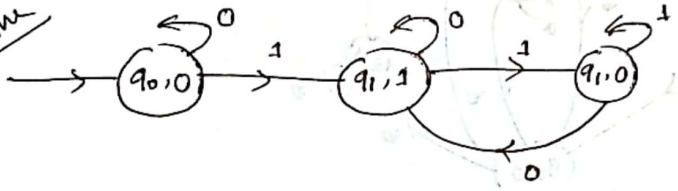
Mealy Machine



→ TT for Mealy

	0	1
→ q_0	$(q_0, 0)$	$(q_1, 1)$
q_1	$(q_1, 1)$	$(q_1, 0)$

Moore Machine

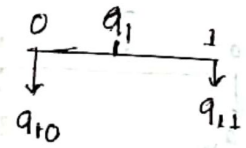


	0	1
q_0	$(q_0, 0)$	$(q_{11}, 1)$
q_{10}	$(q_{11}, 1)$	$(q_{10}, 0)$
q_{11}	$(q_{11}, 1)$	$(q_{10}, 0)$

→ TT for Moore Machine

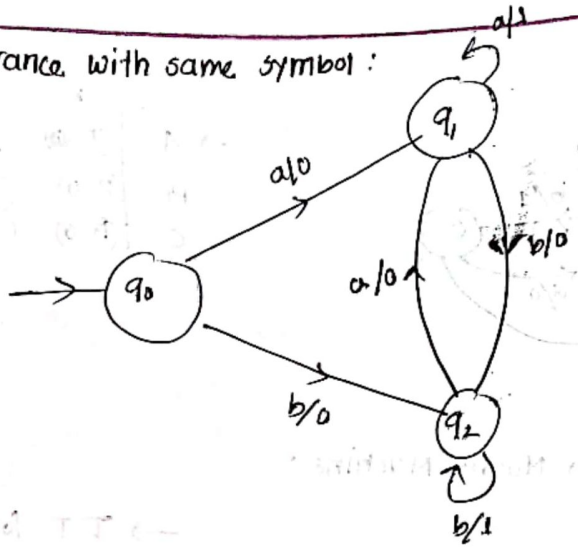
	0	1	Δ
→ q_0	q_0	q_{11}	0
q_{10}	q_{11}	q_{10}	0
q_{11}	q_{11}	q_{10}	1

→ ignore LSB 1 bit of LSB

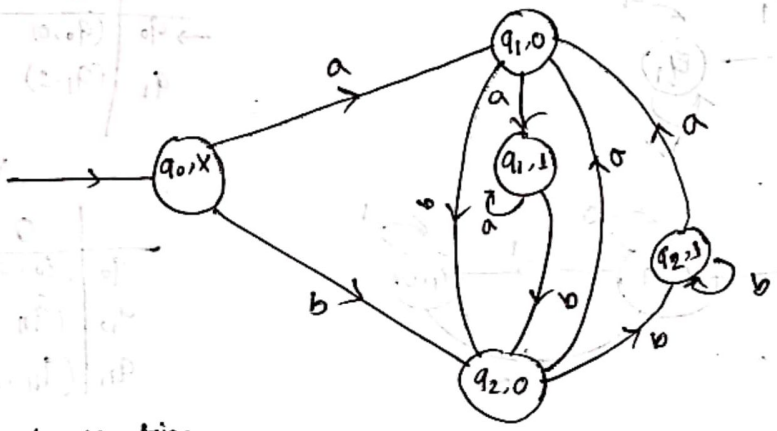


no. of state in Moore Machine = N
no. of output = X
no. of state : N to (NXX)
(NX.X) →
2 · to (2×2) = 4

occurance with same symbol :



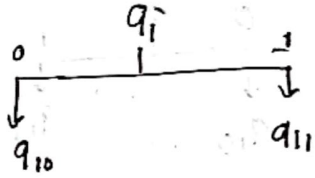
state
$3 \times 2 = 6$
up to
output

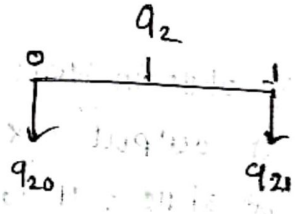


state → 5

TT of Mealy Machine

	a	b
→ q_0	$q_1, 0$	$q_1, 0$
q_1	$q_1, 1$	$q_2, 0$
q_2	$q_1, 0$	$q_2, 1$

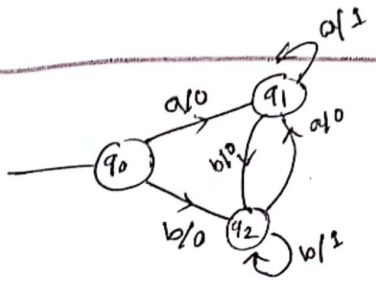




	a	b
→ q_0	$q_{10}, 0$	$q_{20}, 0$
q_{10}	$q_{11}, 1$	$q_{20}, 0$
q_{11}	$q_{11}, 1$	$q_{20}, 0$
q_{20}	$q_{10}, 0$	$q_{21}, 1$
q_{21}	$q_{10}, 0$	$q_{21}, 1$

Moore Machine

	a	b	
q_0	q_{10}	q_{20}	X
q_{10}	q_{11}	q_{20}	0
q_{11}	q_{11}	q_{20}	1
q_{20}	q_{10}	q_{21}	0
q_{21}	q_{10}	q_{21}	1



Date: 17.1.22

Regular lang ac by → Finite Automa

RL can be represented by some expression that is called RE

Regular Expression

RE ⟶ Σ (operand)

+ Union/ ~~conta~~
• concate
* Kleen closure } 3 operators
+ positive closure

(a) Primitive Regular Expression

- ϕ is a RE → $\{\}$
- ε is also a RE → $\{\varepsilon\}$
- $a \in \Sigma$ is a RE
- $\{a\}$ (any lang. with string 'a')

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

- $a+b$ — $\{a, b\}$
- ab — $\{ab\}$
- a^* — $\{\varepsilon, a, aa \ldots\}$

Σ, operators, ( )

$L = \{\varepsilon, a, aa, \ldots\}$
$= \{a^n \mid n \geq 0\}$
$= a^* \quad \Sigma = \{a, b\}$

$L = \{\varepsilon, a, aa, b, ab, bb, \ldots\}$
$= (a+b)^* = (a+b)(a+b)$

exactly 2

$\{aa, ab, ba, bb\}$
$RE = aa + ab + ba + bb$ ✓
$= a(a+b) + b(a+b)$ ✓
$= (a+b)(a+b)$ ✓
$= (a+b)^2$ ✗
$= (a+b)^*$ ✓

At least two

$L = \{aa, ab, ba, bb, \ldots\}$

$RE = (a+b)(a+b)(a+b)^*$ ✓

At most two (length)

$L = \{\varepsilon, a, b, aa, ab, bb, ba\}$

$RE = \varepsilon + a + b + aa + ab + bb + ba$ ✓

$= (\varepsilon + a + b)(\varepsilon + a + b)$ ✓

length is even : 0, 2, 4, 6, ...

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

length is odd

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

$\cong$ 2 mod 3

$(a+b)^{3n}$

$((a+b)(a+b)(a+b))^*$ → divisible by 3

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

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

no. of a's is exactly 2

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

at most two

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

no. of a's is even

$(b^* a b^* a b^*)^* + b^*$ ✓

$(b^* a b^* a b^*)^* \cdot b^*$ ✓

starting with a

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

Ending with a

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

Starting with a and ending with b

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

Starting and ending diff

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

Starting and ending same

$RE = a(a+b)^* a + b(a+b)^* b + (\varepsilon + a + b)$

→ $\{a^n b^n \mid m,n >, 0\}$

$a^* b^*$

→ $\{a^m b^n \mid m,n >, 1\}$

$aa^* bb^*$

→ $\{a^n b^m c^l \mid m,n,l >, 0\}$

$a^* b^* c^*$

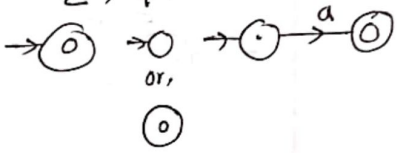
Date: 19.1.22

Lexical Analyzer
↳ FA

RE → FSA/FA (Finite state Acceptor)
ε-NFA

primitive type regular expression

ε, ϕ, $a \in \Sigma$



union/Alteration
concatenation
closure

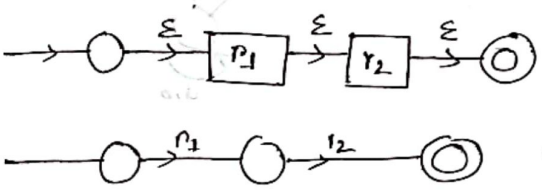
r_1 r_2

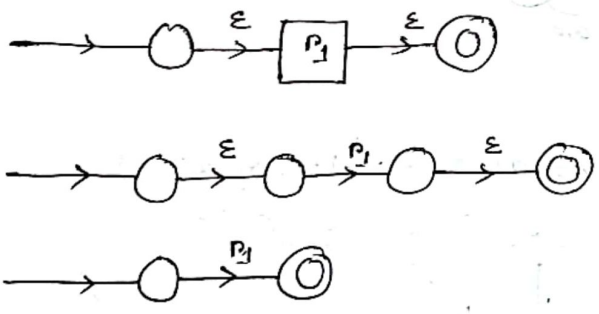
(a) $r_1 r_2$ (conc)

(b) r_1+r_2 / $r_1 | r_2$

(c) r_1^*

Building block

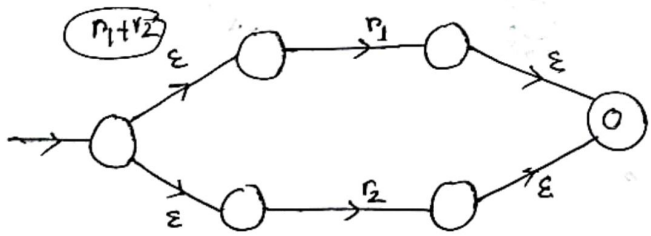


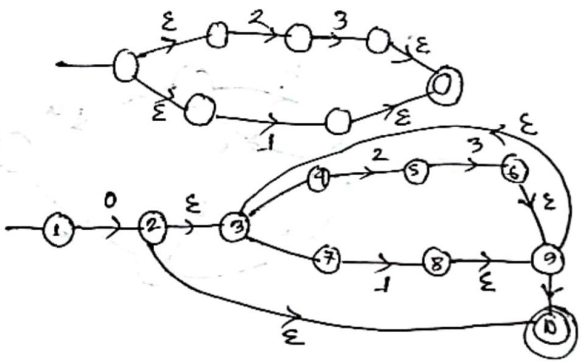


$r_1, r_2 = r_1 + r_2$

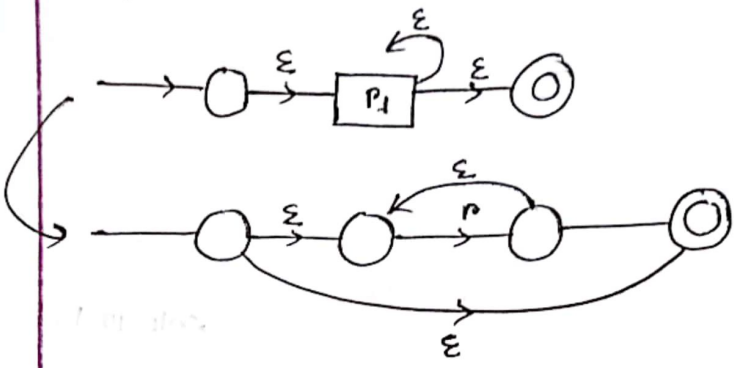
$0\{1+2.3\}$ ← closure { }

$0.\{1+2.3\}^*$



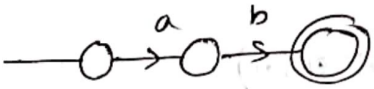


P_1 *

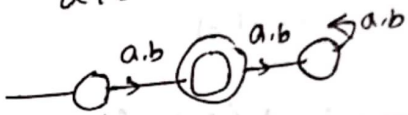


a*

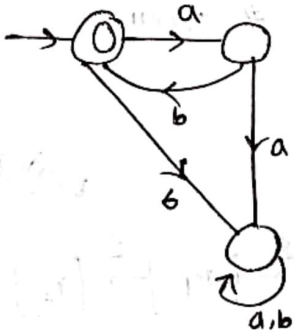
ab



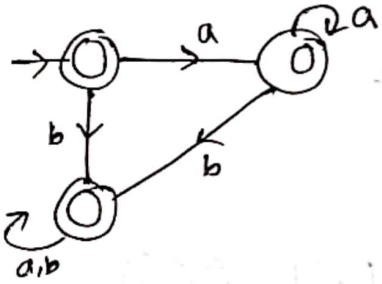
a+b



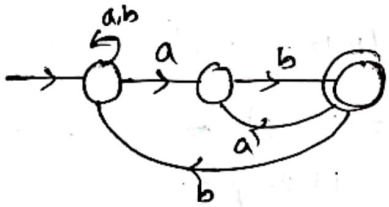
ab*



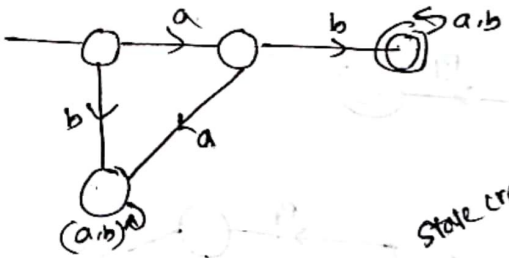
(a+b)* (ε, a, b, ab, aabb. . . .)



(a+b)* ab

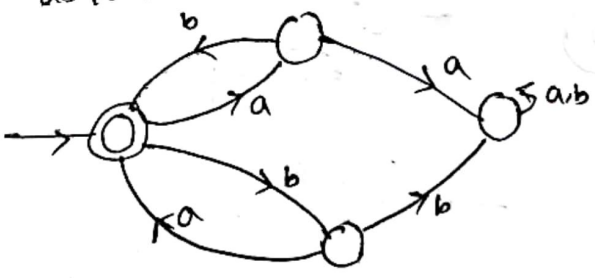


ab (a+b)* → starting with 'ab'



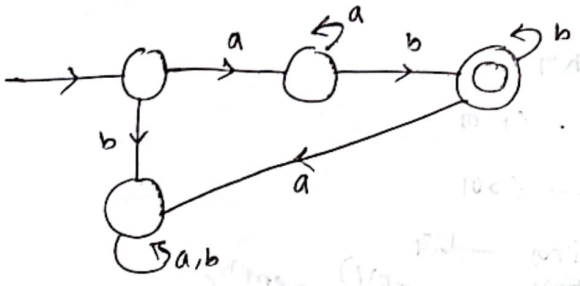
State creation method

ab+ba



Date: 28.2.22

$L = \{a^n b^m \mid n, m \geq 1\}$



$L = \{a^n b^n \mid n \geq 1\}$ ← need to count the number of 'a' and compare with 'b'

* Machine with memory
* Memoryless machine

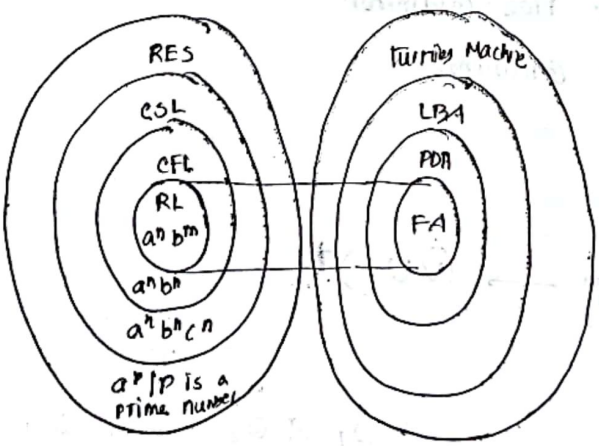
Finite Automata with Memory – PDA

Programming language or other language must defined must be precisely specified.

(1) A set of symbols (Alphabet) that can be used to construct PL - OL.

(2) A set of syntactic rules that ensures the syntactic correctness of sentences or programs.

(3) All the ~~meagr~~ meaningful program sentences.



PDA → pushdown automata

LBA → linear bounded non-deterministic autom

CSL → context sensitive language

RES → Recursively Enumerable sets.

Generative Device :

Grammer is used as the generative device in TOC.

RL ⟶ $a^n b^m$ — FA — RG

CFL ⟶ $a^n b^n$ — PDA — CFG

CSL ⟶ $a^n b^n c^n$ — LBA — CSG

REL ⟶ a^p / p is prime ⟶ Turning Machine — UG

Grammer :

4-tuple = (V, Σ, Φ, S)

- V → set of non terminal symbols (nonempty)
- Σ → set of terminal symbols (non empty)
- Φ → set of production rules / rewriting rules
- S → standing symbol

$V \cup \Sigma$ → vocabulary

Φ : $\alpha \longrightarrow \beta$

$(V \cup \Sigma)^* A (V \cup \Sigma)^* \longrightarrow (V \cup \Sigma)^*$

$A, B \in V$

(a) Unrestricted Grammer

(b) Restricted Grammer

(1) CSG ⟶ content Sensitive Grammer

(2) CFG ⟶ content Free Grammer

(3) RG ⟶ Regular Grammer

* closure
+ ⟶ except ε all can be

CSG

$\alpha \longrightarrow \beta$

$(V \cup \Sigma)^* A (V \cup \Sigma)^* \longrightarrow (V \cup \Sigma)^+$

$|\alpha| \leq |\beta|$

$\alpha = Q_1 A Q_2$

$Q_1 A Q_2 \longrightarrow Q_1 B Q_2$

CSG = $(a^n b^n c^n)$

(1) $S \longrightarrow aSBC$

(2) $S \longrightarrow abc$

(3) $bB \longrightarrow bb$

(4) $bC \longrightarrow bc$

(5) $CB \longrightarrow BC$

(6) $CC = cc$

small → Terminal symbol

(no $\in$ should not be there)

CFG :

$\alpha \in V$ [α only non-terminal symbol]

$\alpha = Q_1 A Q_2$

$\beta = Q_1 B Q_2$

$L(G) = \{a^n b^n \mid n \geq 1\}$

Φ = (i) $S \longrightarrow aSB$

(ii) $S \longrightarrow aB$

(iii) $B \longrightarrow b$

$V = \{S, B\}$

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

S

$S \Longrightarrow aSB$ [i)

$\Longrightarrow aaBB$ [ii)]

$\Longrightarrow aabB$ [(iii)]

} sentential form

$\Longrightarrow aabb$ [iv] ← sentence

Regular Grammer :

$\alpha \longrightarrow \beta$

$\alpha \in V$, β is a form of a, ~~$\alpha\beta$~~ aB, Ba, ϵ

$a \in \Sigma$

$B \in V$

$(\ ) N (\ ) \longrightarrow (\ )$

RLG → Right Linear Grammer (expand in right)

LLG → Left Linear Grammer (expand left)

$S \longrightarrow aB$

$B \longrightarrow cC$

$C \longrightarrow dD$

$D \longrightarrow d$

$S \Longrightarrow aB$

$\Longrightarrow acC$

$\Longrightarrow acdD$

$\Longrightarrow acdd$

T_0, T_1, T_2, T_3

(TM) (CSG) (CFG) (RG)

L_0 L_1 L_2 L_3

Expressive

Regular expression

$L_0 \supseteq L_1 \supseteq L_2 \supseteq L_3$ →

Date: 6.3.22

2 tools for specification
→ Grammer : To generate the whole lang
→ Machine : check the string → Accept/reject

process of Getting strings of a lang by an way is called derivation

2-types :
→ Left most
→ Right most

$L\{a^n b^n \mid n \geq 1\}$

$S \to a\,S\,B$

$S \to a\,B$

$B \to b$

aaabbb

$S \Rightarrow a\,\underline{S}\,B$

$\Rightarrow a\,\underline{a\,S\,B}\,B$

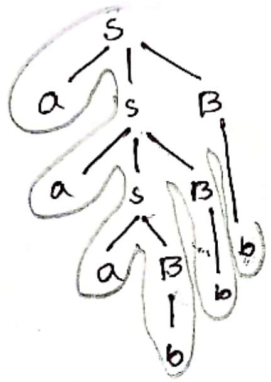
$\Rightarrow a\,a\,\underline{a\,B}\,B\,B$

$\Rightarrow a\,a\,a\,\underline{b}\,B\,B$

$\Rightarrow a\,a\,a\,b\,b\,B$

$\Rightarrow a\,a\,a\,b\,b\,b$

Regular Expression : Generate lang

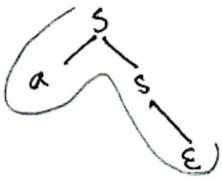


pictorial representation

$\#\{a^n \mid n \geq 1\}$

$a\,a^*$

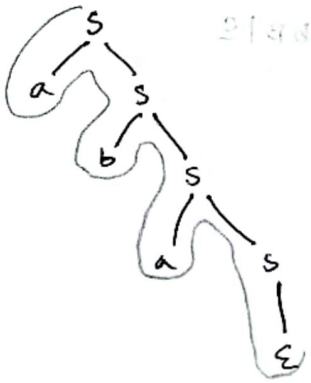
$S \to a\,S \mid \varepsilon$



$\#\{a^n b^m \mid n, m \geq 0\}$ → lang. over, or everything over a and b

$(a+b)^*$ ← Regular expression

$S \to a\,S \mid \cancel{a\,B}\; b\,S \mid \varepsilon$



Context free Grammer

length of string over {a,b} is exactly '2'

RE = $\underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A}$ ↳ {aa, ab, ba, bb}

$S \rightarrow AA$ (for RL/LL)

$A \rightarrow$ ~~a/b~~ a | b

$S \rightarrow aa \mid ab \mid ba \mid bb$ → Grammer

Generate same lang

$\alpha \rightarrow \beta$

$\alpha \in V$, β is the form of $a\beta$, a, ε

at most '2'

$S \rightarrow a \mid b \mid aa \mid bb \mid ab \mid ba \mid \epsilon$

$\underbrace{(a+b+\varepsilon)}_{A}\underbrace{(a+b+\varepsilon)}_{A}$

$S \rightarrow AA$

$A \rightarrow a \mid b \mid \varepsilon$

at least '2'

$\underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A}\underbrace{(a+b)^*}_{B}$

$S \rightarrow AAB$

$A \rightarrow a \mid b$

$B \rightarrow aB \mid bB \mid \varepsilon$

context free grammar

length of string over {a,b} is exactly '2'

RE = $\underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A}$ ↳ {aa, ab, ba, bb}

$S \rightarrow AA$ (for RL/LL)

$A \rightarrow ~~a/b~~ \; a \mid b$

$S \rightarrow aa \mid ab \mid ba \mid bb$ → Grammer

Generate same lang

$\alpha \rightarrow \beta$

$\alpha \in V$, β is the form of $a\beta, a, \varepsilon$

at most '2'

$S \rightarrow a \mid b \mid aa \mid bb \mid ab \mid ba \mid \epsilon$

$\underbrace{(a+b+\varepsilon)}_{A}\underbrace{(a+b+\varepsilon)}_{A}$

$S \rightarrow AA$

$A \rightarrow a \mid b \mid \varepsilon$

at least '2'

$\underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A}\underbrace{(a+b)^*}_{B}$

$S \rightarrow AAB$

$A \rightarrow a \mid b$

$B \rightarrow aB \mid bB \mid \varepsilon$

Length of string is 'even'

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

S

~~$S \to A\,S \mid \varepsilon \quad A\,S \mid \varepsilon$~~

~~$S \to AA \mid BB$~~

~~$A \to a \mid b \mid aB \mid bB$~~

$S \to A\,S \mid \varepsilon$

$A \to BB$

$B \to aB \mid bB$

Palindrome

$\varepsilon, a, b, aa, bb, aba, bab, \ldots$

WW^R (even) $\quad wRw \mid waw^R \mid w^R a w$ (even, odd, odd)

~~$S \to SS \mid SAS \mid$~~

$S \to SAS \mid SAS \mid SAS$

$A \to a \mid b \mid \varepsilon$

for odd ww^R

$\{a^m b^m \mid m, m \geq 1\}$

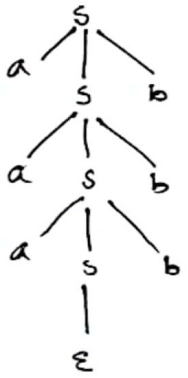
$S \to aSB$

$S \to aB$

$B \to b$

$\{a^n b^n \mid n \geq 1\}$

$S \to asb \mid ab$



$\{a^n b^n c^m \mid n, m \geq 1\}$

$S \rightarrow BC$

$B \rightarrow aBb \mid ab$

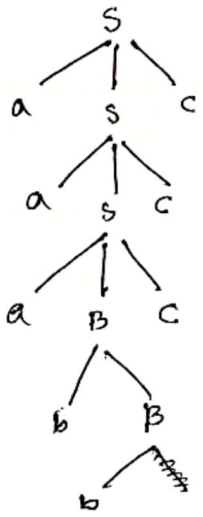
$C \rightarrow cC \mid c$

$\{a^n b^m c^n \mid n, m \geq 1\}$

$S \rightarrow aBc \mid abc \mid asc$

$B \rightarrow bB \mid b$

aaabbccc



$a^n b^{2n} \mid n \geq 1$

abb

aa bbbb

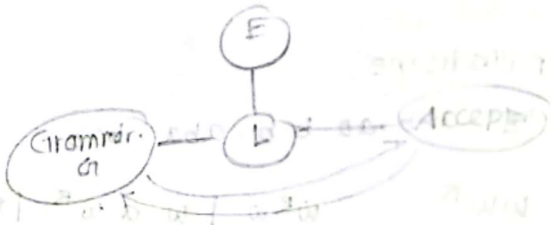
$S \rightarrow aSbb \mid abb$

$\{a^n b^m c^n \mid n \geq 1\}$

~~$a^n b^n \mid n \geq 1$~~

context sensitive

$S \rightarrow abSc \mid abc$



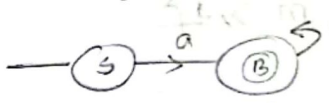
conversion to FA to regular exp
via versa

aa^*

$S \rightarrow aS \mid a$

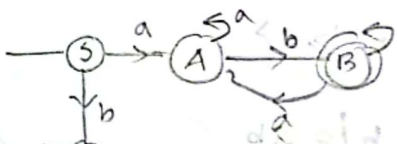
$S \rightarrow aS$

$S \rightarrow a$



$a^n b^m$

starting and ending with different symbol



Right linear

$S \rightarrow aA \mid bC$

$A \rightarrow aA \mid bB$

$B \rightarrow bB \mid aA \mid \varepsilon$

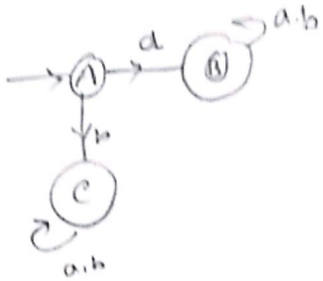
$C \rightarrow aD \mid bC$

$D \rightarrow aD \mid bC \mid \varepsilon$

starting with 'a'

Right linear

$A \rightarrow aB \mid bC$
$B \rightarrow aB \mid bB \mid \varepsilon$
$C \rightarrow aC \mid bC$

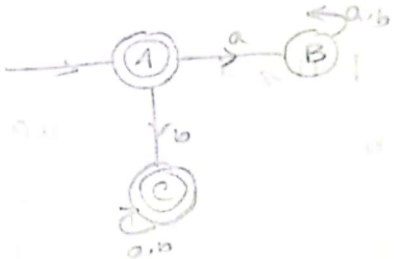


Left linear

$A \rightarrow Ba \mid Cb$
$B \rightarrow Ba \mid Bb \mid \varepsilon$
$C \rightarrow Ca \mid Cb$

will generate the ~~lang~~ inverse lang (not starting with 'a')

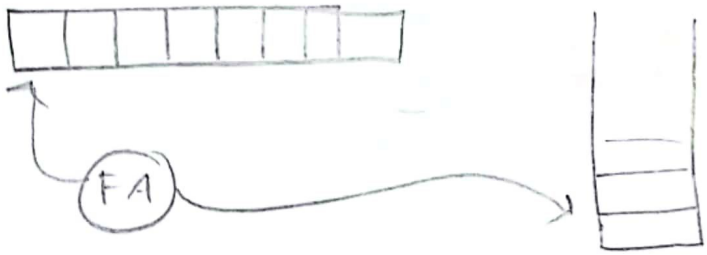
$\{\varepsilon, b, ba, bab, \ldots\}$



Context free Grammer

→ Push-down Automata (PDA)

- Acceptor
- PDA = FA + memory (infinite)
  = FA + stack → can handle infinite memory



PDA

7-tuple

$$P = (Q, \Sigma, \delta, q_0, Z_0, Z, \mu)$$

Q = nonempty set of state
Σ = Alphabet / string
δ = transition function
q_0 = starting state
Z_0 Z_0 = bottom of stack
Z = set of ~~finite~~ final state (may / may not)
μ = top of stack (input)

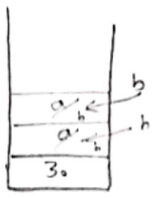
$\delta : Q \times (\Sigma \cup \varepsilon) \times \mu \longrightarrow Q \times \mu^*$ → Deterministic PDA

$\delta :$

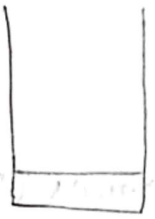
$2^Q \times \mu^* \longrightarrow$ Non-det PDA

$a^n b^n \mid n \geq 1$

aabb



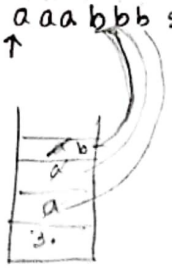
aabb(b)ε → reject

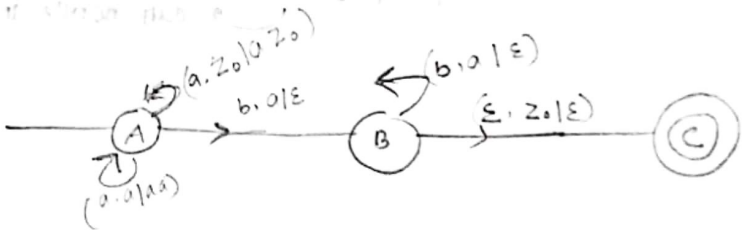


2 condition for Accepting
→ stack should be empty
→ should be of ε

- Accepted by empty stack
- " " a final state

aaabbbε





abaabb
↑?
cannot accept anymore

Z_0/Z_0

aabε

cannot accept

$(A, a, Z_0) \longrightarrow (A, a, Z_0)$
$(A, a, a) \longrightarrow (A, aa)$
$(A, b, a) \longrightarrow (B, \varepsilon)$
$(B, b, a) \longrightarrow (B, \varepsilon)$
$(B, \varepsilon, Z_0) \longrightarrow (C, \varepsilon)$

→ Design a PDA to accept a specific lang