TOA

Lecture 01

Language

Whenever we built a language we first define its alphabets.

it is represented by Σ

alphabets

$\Sigma = \{a, b\}$ cardinality → no of alphabets in set $= 2$

size = ~~1~~ 2

$\Sigma = \{ab, ba\}$ cardinality $= 2$

size $= 1$

all alphabets have size of 1.

Λ → NULL string (size = 0)

or λ

or ϵ

Σ^1 → size of strings
Σ = no. of strings.
Σ → size(no. of alphabets.

$\Sigma = \{a, b\}$ we can make infinite strings from this

$\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots\ldots$

$\{a,b\}^0 \cup \{a,b\}^1 \cup \{ab\}^2 \cup \ldots$

$\{\Lambda\} \cup \{a,b\} \cup \{aa, ab, ba, bb\}, \cup \ldots$

$\{\Lambda\}$ → size 0 string is NULL.

$\{a,b\}, \{a,b\}$ we will take their cartesian product $\{aa, ab, ba, bb\}$

The strings which becomes part of a language are called "words".

Invalid alphabets :-

The string which can be generated in more than one way. Because it contains ambiguity.

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

it can be generated by tokenizing it a, b (size = 2). and it can also be generated by ab.

$L1 = \{a, b\}$

cardinality $= |L1| = 2 = n(L1) =$ no. of elements in $L1$

length = let length of $x_1 = |x_1| = 1$

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

alphabet کا size شروع میں ڈیفائن ہوتا ہے -

اور یہ اگر ان alphabet کو ملا کر کوئی string

بنا لیں تو size + ہو جاتے ہیں -

e.g $\{a, b, ab\}$ (1) (1) (1)

ab if it come from sebarate a, b then its size is 2

if it comes from ab then its size = 1.

کوئی بھی alphabet کسی دوسرے alphabet کا as whole

prefix میں نہیں ہوگا

تو یہ valid

alphabet ہو گا -

$L = \{ab, ba\}$

Invalid:

$L = \{ab, ba, abc\}$

ab is prefix of abc so its invalid alphabet

invalid string: a string which can't be generated by alphabets. e:g $\Sigma = \{a, b\}$ string = abc invalid, c not alphabet

valid string: which can be generated by alphabets. $\Sigma = \{a, b\}$ string = ab

$\Sigma = \{ab, bab, ac\}$ valid alphabets

$\Sigma = \{a, ab, b\}$ invalid alphabets

$L = \{w / w \text{ over } \Sigma = \{a, b\}, w \text{ contains at most one 'a'}\}$

null string. ← Λ, a, b, ab, ba, bb, bbb, abb, aba, bab ....

0 size string.

size = 2 size = 3

Rejected strings are = aa, aaa, aab, aba, baa ...

how many strings we can generate of size n from set of alphabet with cardinality k

$= k^n$

e:g $\Sigma = \{a, b\}$ how many strings we can generate of size 3

$2^3 = 8$ (it contain both rejected and accepted strings)

Finite Automata

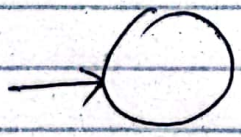
FA

- DFA — Deterministic
- NFA — non-deterministic

It will contain finite alphabets and finite states.

Σ (alphabets), Q (states), S (start state), f_n (set of final states), δ (transition function).

$\Sigma \times Q$ (cartesian product) → action. (we give input it will give output).



starting state (q_0)

final state

Starting state will always be one.

DFA contain:

Q (finite set of states)

Σ (finite set of alphabets)

q_0 (starting state). $q_0 \in Q$ ✓ $q_0 \subset Q$ ✗

$F \subseteq Q$

$F \not\subseteq Q$ ✗

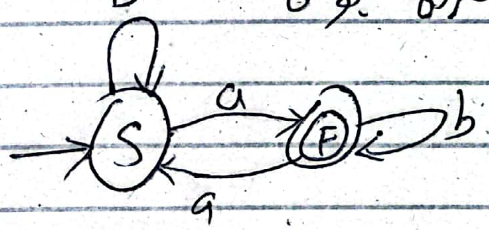
F (finite set of ~~starting~~ final state)

if we do not make final state it will be create empty language

$L = \{ \}$

δ (transition rules)

اگر DFA میں ہم نے دو state بنائی ہوئی ہیں تو ہر alphabet کا 2 state کے لیے rule ہوگا -



$\delta(S, a) = F$

مطلب S پر read a کریں گے تو ایسا ہی جائیں گے

اور ہر alphabet کا rule ہر state کے لیے بنائیں گے -

$\delta(S, b) = S$

$\delta(F, a) = S$

$\delta(F, b) = F$

اگر ہم نے NULL کو string accept کروانی ہے تو ایک ہی state ہوگی جو ہی starting اور Final.

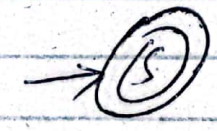
صرف وہ والی strings جو accept ہوں
جن میں ہم start سے starting state
سے شروع ہو کر final state تک پہنچ جائیں۔

from previous question,

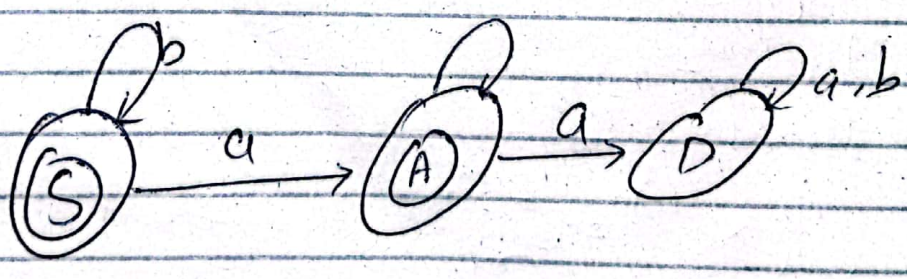
{a, ba, ab, bba, abb, aaa ...}

وہ final جو starting state ہو اور null accept ہو

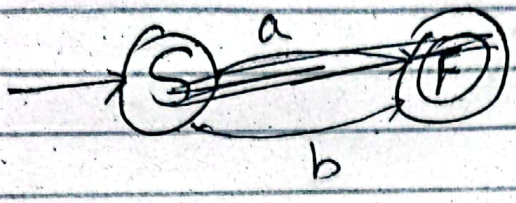
e.g

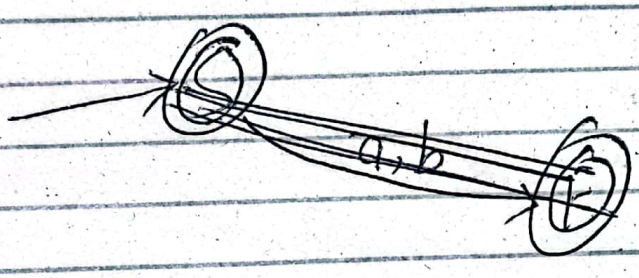


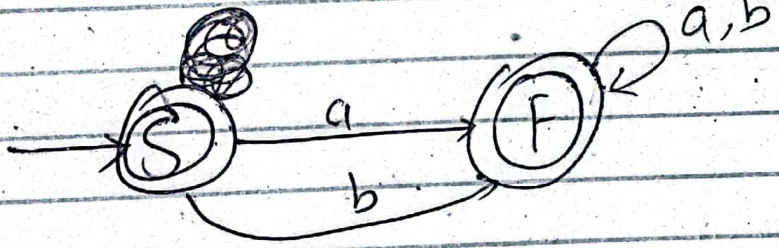
A = Λ, a, b, ab, ba, bb, bbb, abb ....

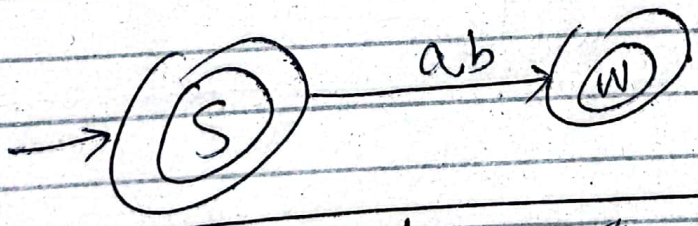


Finite automata is
Basic with no memory



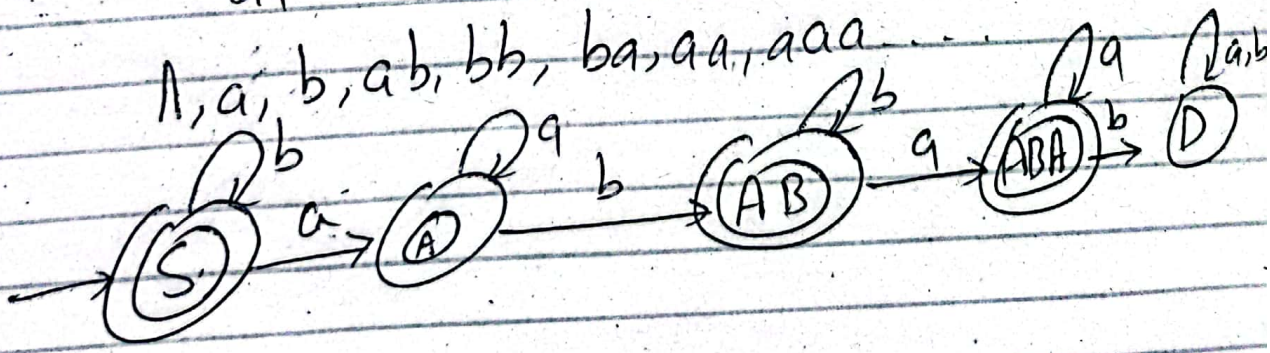






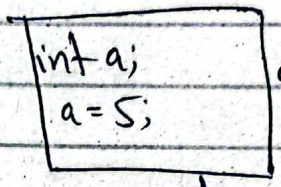
at most one "ab"

Λ, a, b, ab, bb, ba, aa, aaa . . .



ToA

Application



words

int ← keyword

a ← identifier

5 ← number

Computer

→ problem solver

→ Yes | No

Acceptance / Rejection

word is made of alphabets.

collectively it is called language.

"collection of words"

word is collection of alphabets

PDA is finite automata with memory.

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

$L = \{x / x \in \Sigma \; 0 \leq |x| \leq 2\}$

length of $x \geq 0$ and $x \leq 2$

we will make woods in canonical order.

e.g 0, 1, 2, 3

$x = \Lambda$ = NULL = smallest wood with length 0.

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

$L = \{x / x \in \Sigma \; ;$ is a palindrome$\}$

$0 \leq |x| \leq 3$

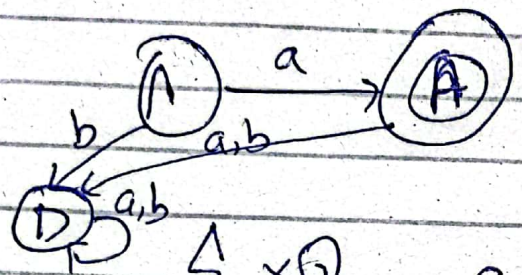
$\Sigma = \{a, b\}$

$L_1 = \{\Lambda, a, b, aa, bb, aaa, aba, bab, bbb\}$

Cardinality $= |L_1| = 9$

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

$Q = \{\Lambda, a\}$



our language is only accepting 1 a. we are starting with null state.

$\Sigma \times Q$ $\quad a, \Lambda \rightarrow \quad a, A \rightarrow$

$b, \Lambda \rightarrow \quad b, A \rightarrow$

$a, D \rightarrow D$

$b, D \rightarrow D$

DA

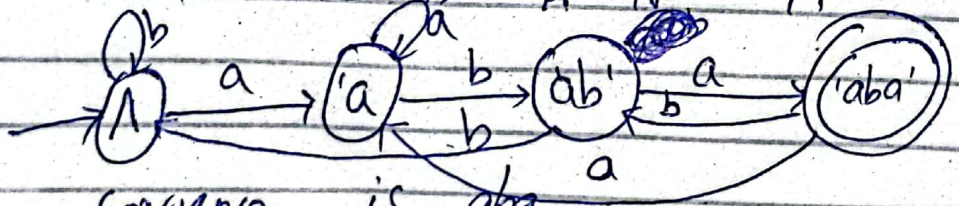
$\Sigma = \{a, b\}$ → case 6 ends with

$L = \{x \mid x \in \Sigma, x \text{ ends with 'aba'}\}$

{aba, aaba, baba, aaaba, ababa, baaba, bbaba ---}

3 4 4 5 . . .

start state is A NULL Λ



Sequence is aba

we divide sequence in prefixes and we observe suffixes of strings.

Sequence: aba

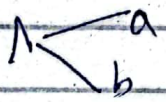
String will be our input.

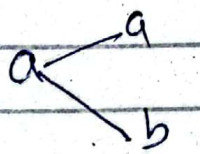
prefixes:

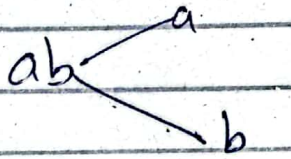
∧
a
ab
aba

· a, ∧ → a

· b, ∧ → ∧







to draw arrow we will look into suffix. e:g at ∧ state we can go only at a if we got a and back to ∧ in case of a

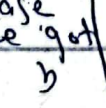
if we are at a state then we have two options what will happen if we got a and what will happen if we got b.

prefixes
∧
a
ab
aba

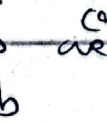
suffix
∧
a
aa

here match is at ∧ and a and longer match is aa so

in case we got b

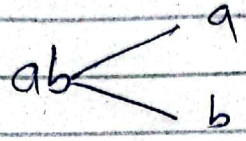


in case we got a



each string has null at end so suffix will be

lets assume we are at state ab



prefixx	suffix
^	^
a	a
ab	ba
aba	aba

match

we got a after ab (aba)

longest match is at aba so we will move to state aba if we got a at "ab"

Now to b at ab

prefixx	suffix
^	^
a	b
ab	bb
aba	abb

match

longest match is at ^ so it we are at ab and we got b then we will move to ab ^ state

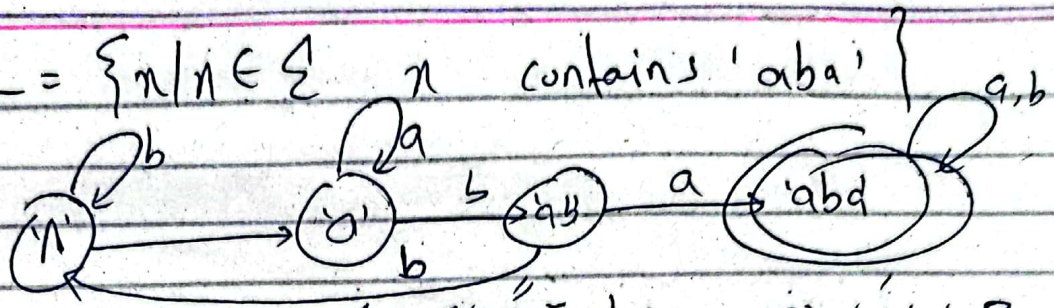
L=

final ~ state

we will

accept

$L = \{x / x \in \Sigma^* \text{ } x \text{ contains 'aba'}\}$



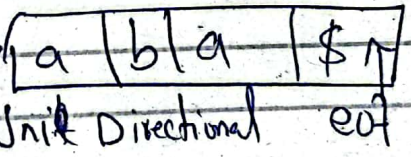
final ~ state

String processing

we want to check wether a string will be accepted or rejected.

aba → string

Yes / No



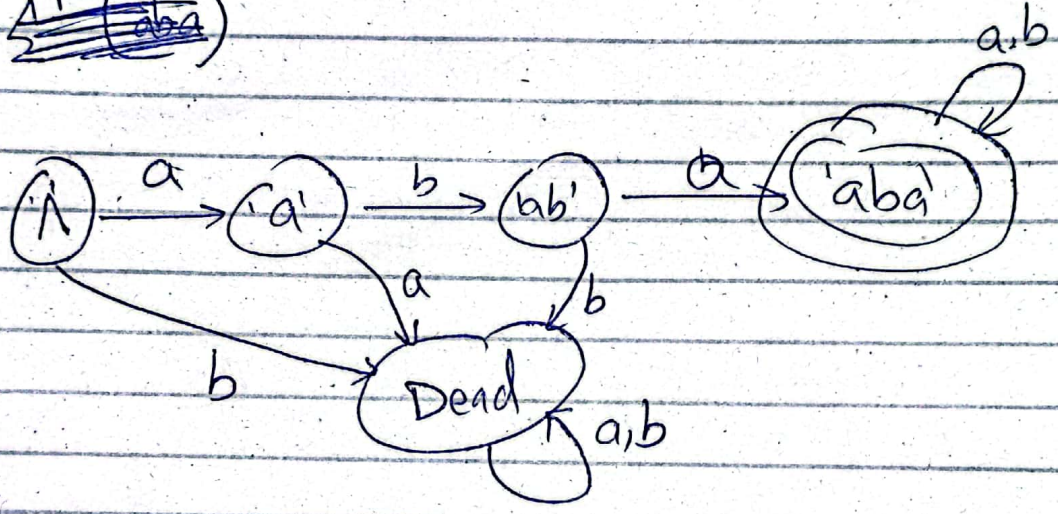
Q	Σ	Action
'Λ'	a	'a'
'a'	b	'ab'
'ab'	a	'aba'

end ~ input tape ~ final state ~ string accept

DFA for language starting with 'aba'.

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

$L = \{\underset{3}{aba}, \underset{4}{abaa}, \underset{3}{abab} \cdots\}$



if minimum size string is of 3 then minimum we need 4 states

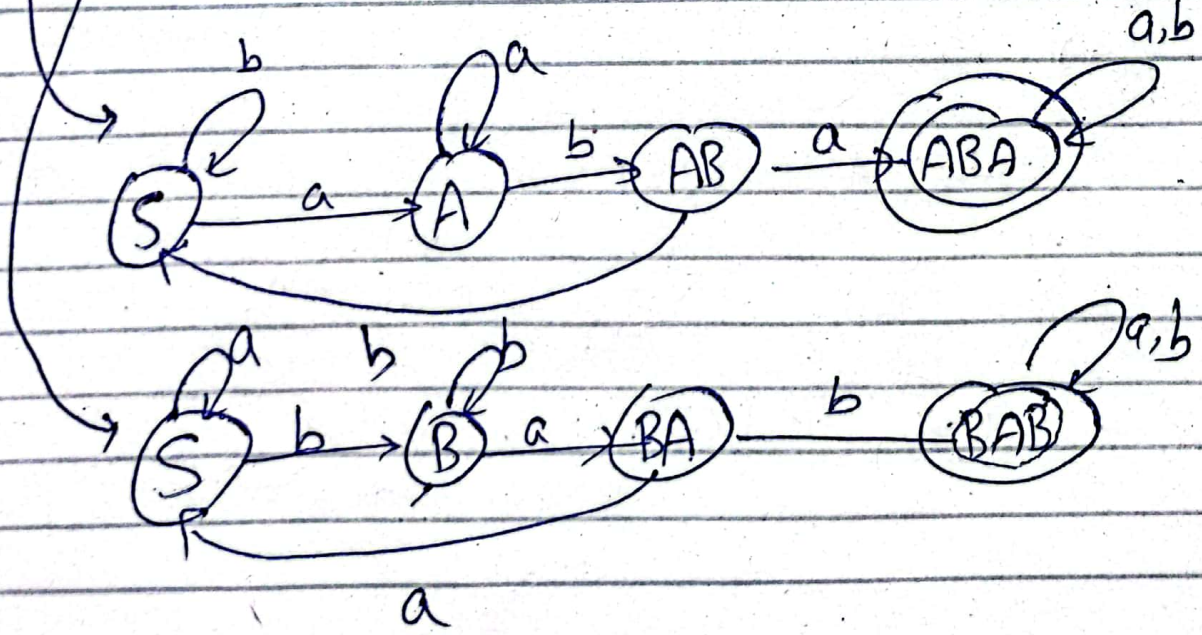
Regular expression (given) → NFA-Λ → NFA → DFA

NFA → R.E
DFA → R.E
NFA-Λ → R.E

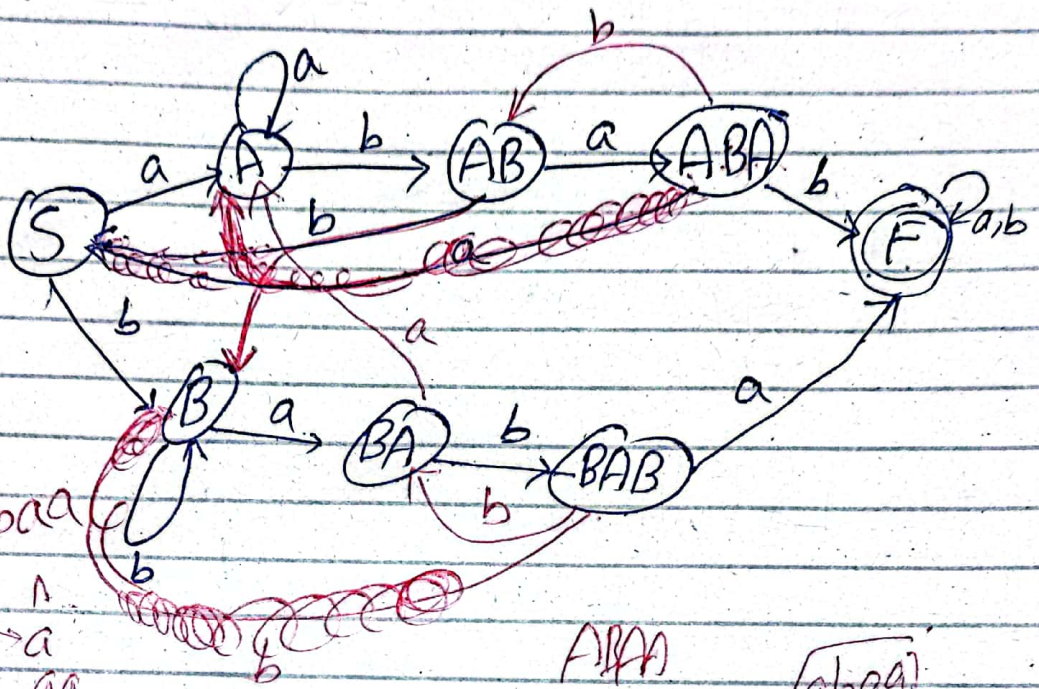
$L = \{w \mid w \text{ over } \Sigma = \{a,b\}$ w contains 'aba' and 'bab'$\}$

we can break it using set operation

we can find w contain aba and then find w contain bab and then take their intersection.



closure property



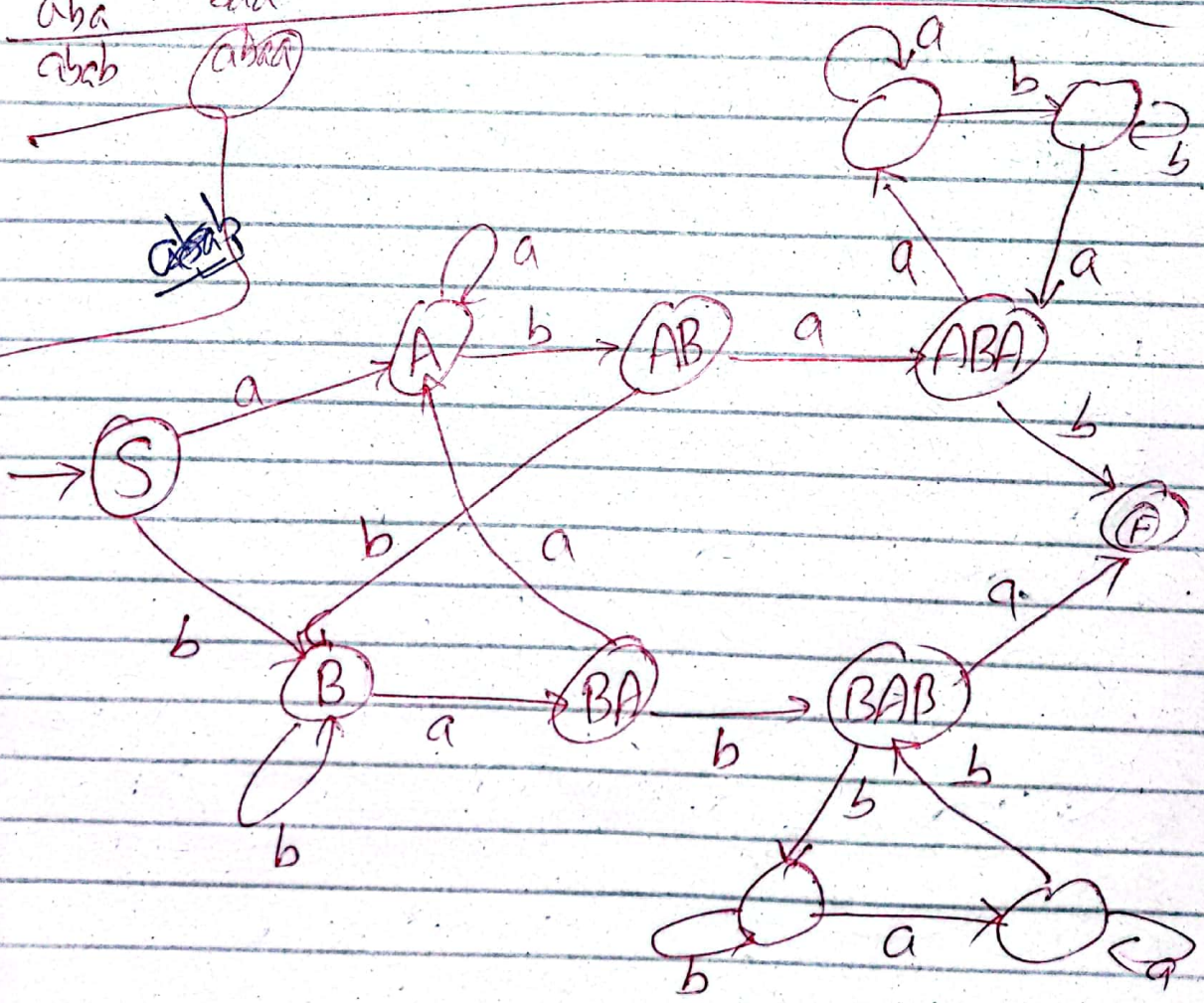
a
ab
aba
abab

a
aa
baa
abaa

abaa

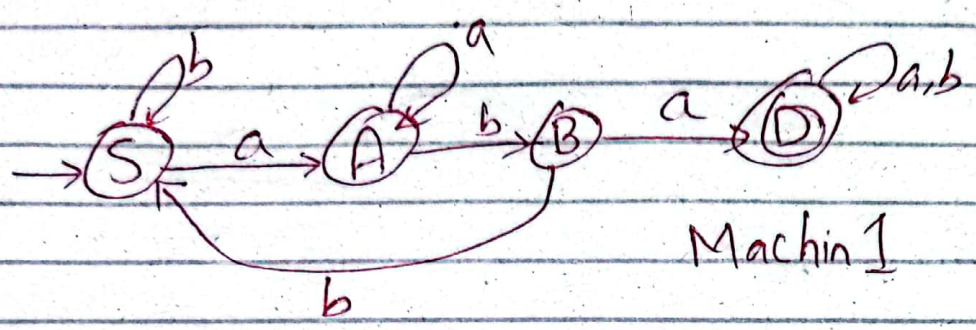
ABA

abaa

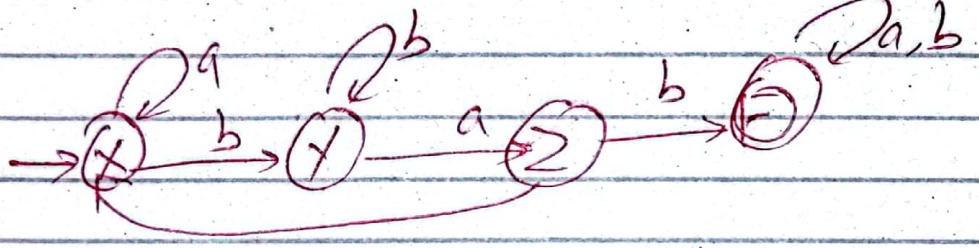


Closure Properties

Divide Complex problem into sub-problems.



Machin 1



Machine 2

M_1 states = {S, A, B, D}

M_2 states = {X, Y2, F}

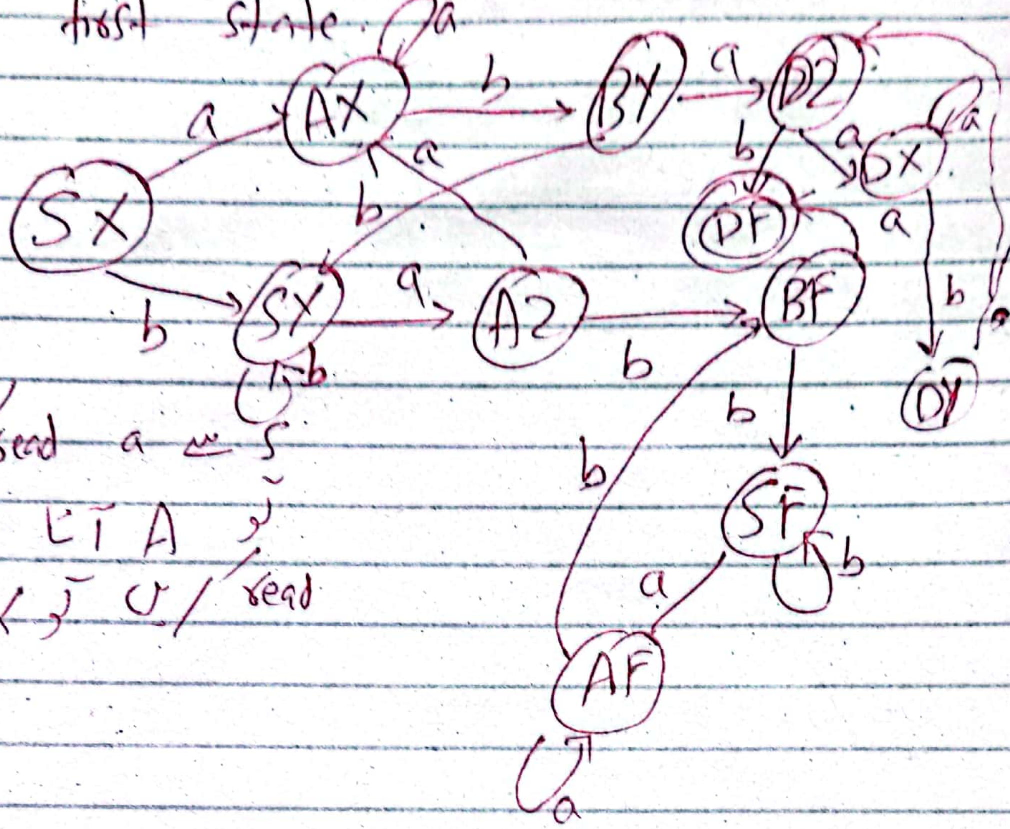
Maximum no. of states will be = Cartision product of

$M_1 \times M_2 = 16$

$4 \times 4 = 16$

SX, SY, S2 - - - -

We will start with first value as first state.



S سے a read ہو/
تو A آتا اور X سے
read ہو/ تو X آتا ہے

ان میں final states وہ ہیں جن میں

M₁ کی final state ہو اور M₂ کی final state ہو۔

intersection of final states of M_1 and M_2

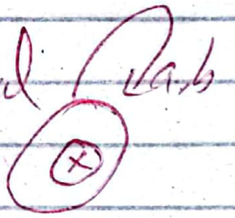
اگر Q میں OR (یا) دیا ہوتا تو کوئی ایک
بھی کوئی final ہو تو وہ ہم لے لیتے۔

Compliment of DFA:

contain ABA

Compliment = does not contain ABA.

contains کی machine بنی ہے جو final states ہیں
ان کو non-final کر دیں اور non-final کو final کر دو
تو وہ compliment بن جائے گا۔



Universal Sigma* all accepted

Universal — Machine

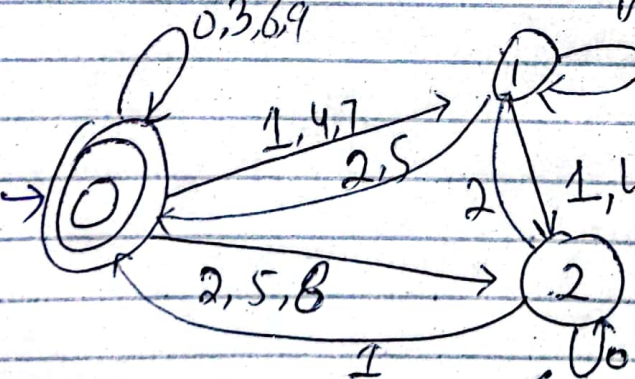
XS, XA, XB, XD

وہ تمام final states جو universal میں ہیں یوں کی جیسی
کو final state
موجود ہو ملیں machine کی final state نہ کرو۔

$L = \{w \mid w \text{ over } \Sigma = \{0,1,2\}$, number in w is divisible by $2\}$

$\Sigma = \{0,1\}$
we can have 3 assumptions from a no.
e.g. $2 \leftarrow 10 \rightarrow$ number ten
sum of digits $= 1+0 = 1 \rightarrow$ binary string 2.

$L = \{w \mid w$ over $\Sigma = \{0,1,2\}$, number in w is divisible by $3\}$



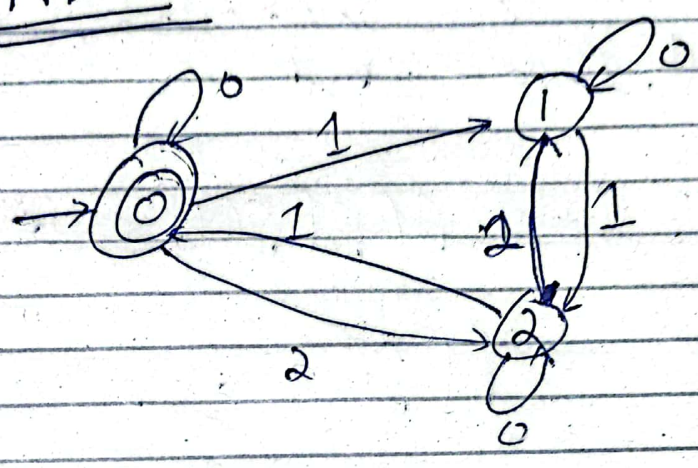
$\frac{3}{3} = 1$ $3\overline{)1}$

$\frac{2}{3}$

01 → $\frac{1}{3} = 1$

read 1 = 0 ↳ remainder.

NFA



Transition Table:

States

	Σ 0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\delta(0,0) = 0$

Suppose our states

0 = A
1 = B
2 = C

	0	1	2
A 0	0(A)	1(B)	2(C)
B 1	1(B)	2(C)	0(A)
C 2	2(C)	0(A)	1(B)

Extended Transition function

$$\delta^*(A, 012) = \delta(\delta(\delta(\delta(A,0),1),0),2)$$

$$\delta(\delta(\delta(A,1),01),2)$$

$$\delta(\delta(B,0),2)$$

$$\delta(B,2),$$

$$= A \quad \text{string acceptable.}$$

NFA

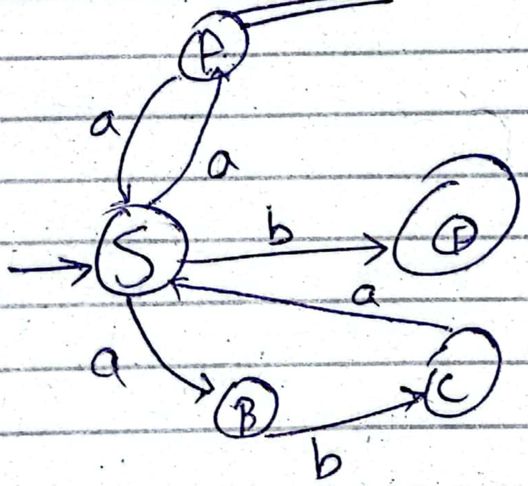
Non-Deterministic Finite Automata

$L = \{aa, aba\}^{*} . b$ ($* \rightarrow 0 \ldots \infty$)

minimum size string is b.

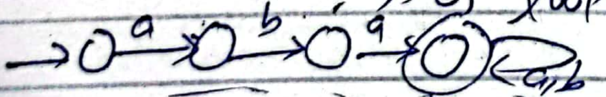
$L = \{ b, aab, abab, aaaab, aaabab \ldots \}$

NFA

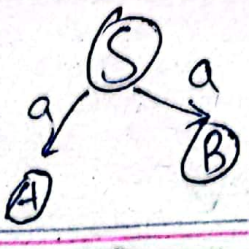


Contain کرنے والے NFA میں minimum size کے بنے states بنا دو اور final اور end پہ loop لگا دو۔

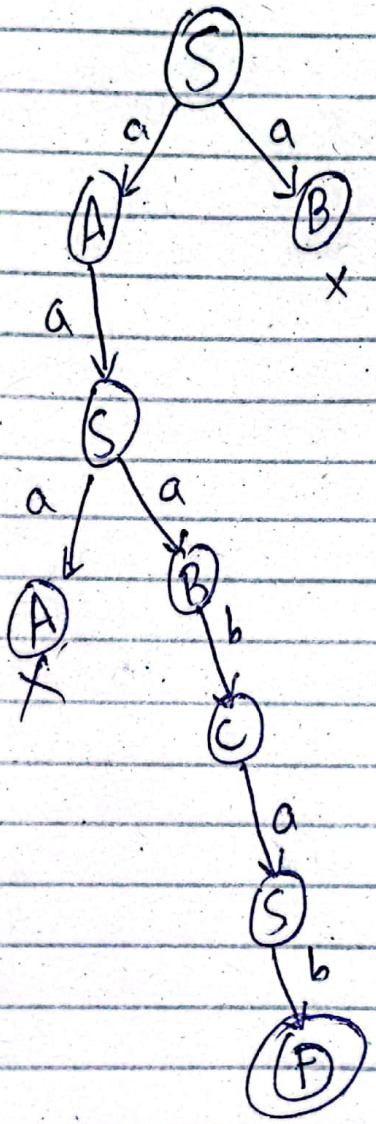
start with میں minimum size کے بنے states بنا دو اور final state پہ loop لگا دو



end with اور start state پہ loop لگا دو۔



$\delta^*(S, aaabab) = \delta^*(\{A, B\}, aabab)$.



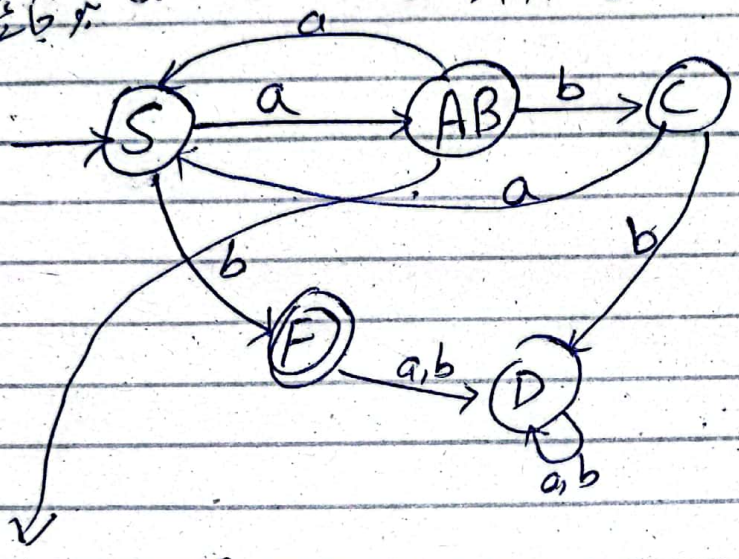
	$\delta^*(q, a)$	$\delta^*(q, b)$
S	{A, B}	{F}
A	{B}	$\phi = \{\}$
B	ϕ	{C}
E	{S}	{C}
F	ϕ	ϕ

NFA to DFA

{S, A, B, C, F} no. of states in DFA = power set
$= 2^5 = 32$

starting state of DFA = starting state of NFA

Find all the states of DFA from NFA



{ } = Dead

ϕ = Dead

$\delta(A, a) = S$

$\delta(B, a) = \phi$

$S \cup \phi = S$

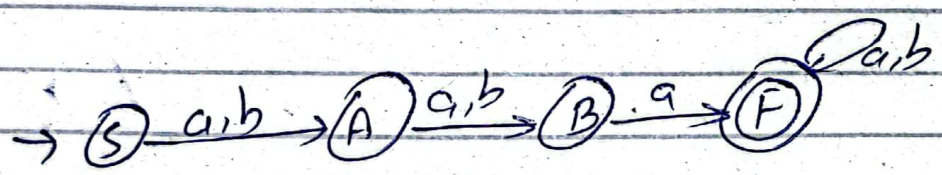
	a	b
S	AB	F
AB	S	C
C	S	D
+F	D	D
D	D	D

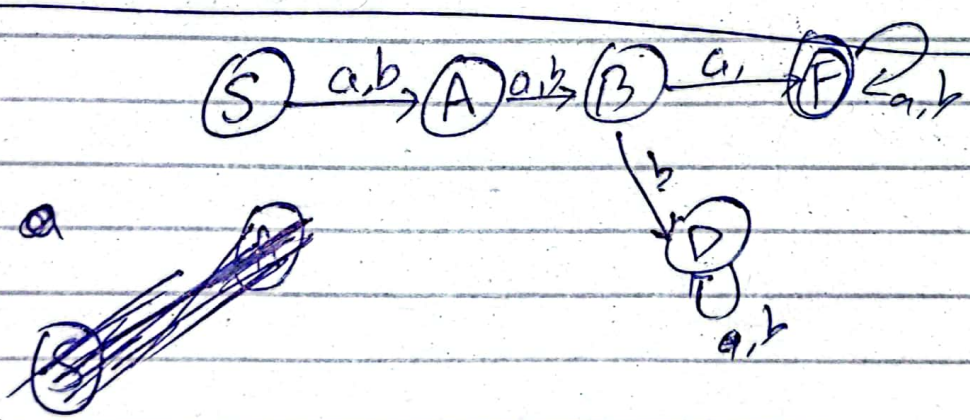
Every DFA is NFA.

$L = \{w | w$ over $\Sigma = \{a, b\}$, third character from start is $a\}$

baa, bba, aaa, ~~aaaa~~, aba,

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

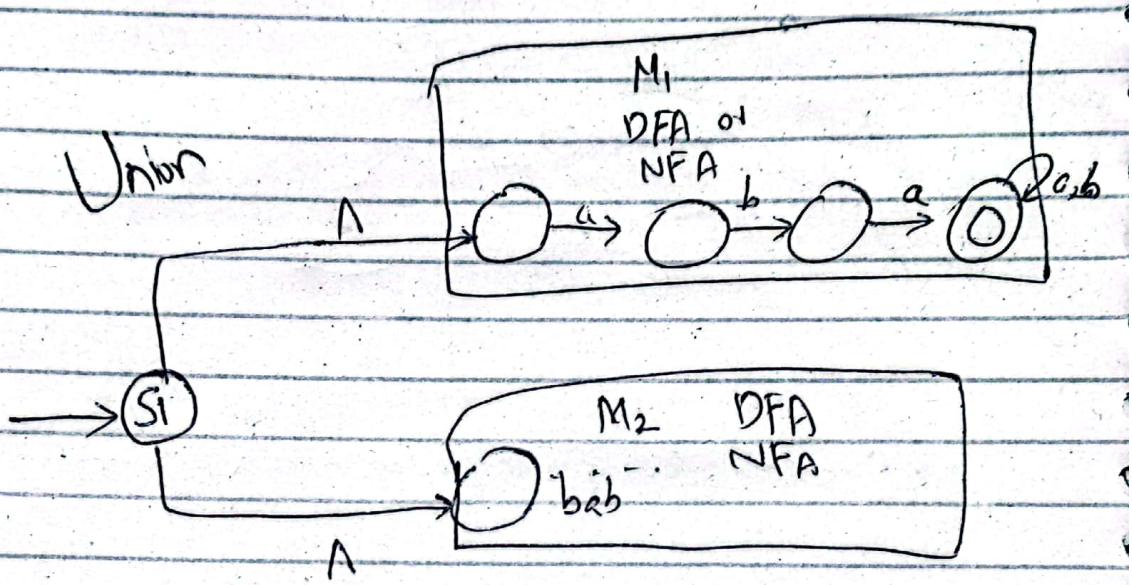




NFA - Λ (NULL)

آل میں:

NFA میں ایک state سے دوسرے state پر move کرنے کے لیے بغیر کسی alphabet کے NULL استعمال کرتے ہیں۔



$L_1 = \{$ contain aba $\}$

$L_2 = \{$ contain bab $\}$

Concate L_1L_2

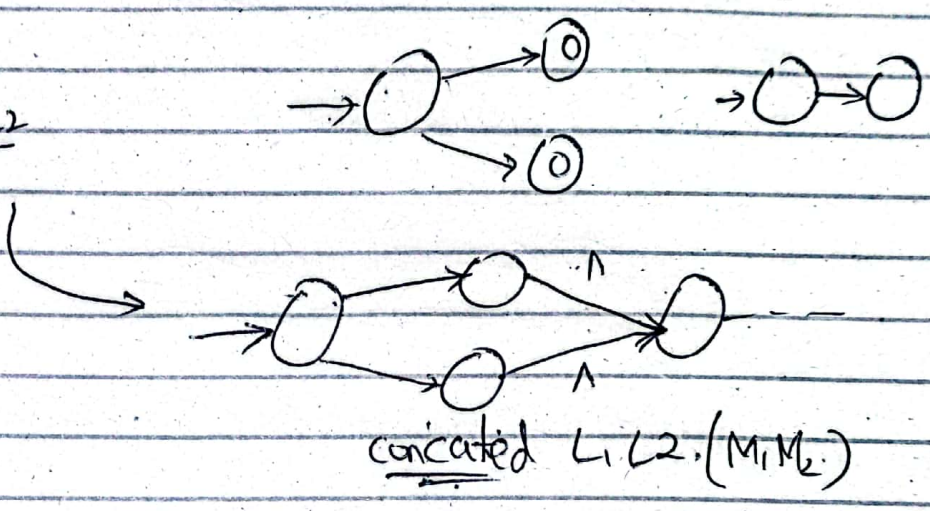
L_1L_2 : aba.bab

آخری machine بنا لیں اور machine کی L_2 کی بنانے کے لیے NFA-Λ پھر

machine L_1 کی جتنی بھی Final States ہیں وہاں سے ایک ایک transition لگا کر machine L_2 کی starting states تک جا سکتے ہیں اور Final states کو non Final کر دیں گے۔

m_1 m_2

$L_1 \text{ concat } L_2$

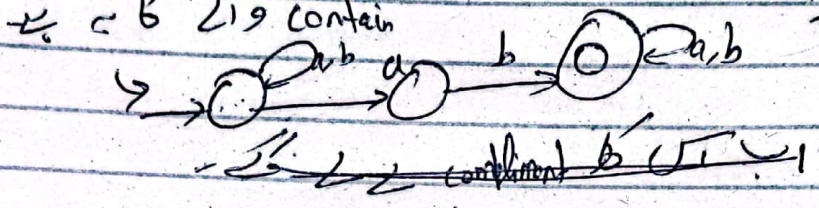


concated $L_1 L_2 . (M_1 M_2)$

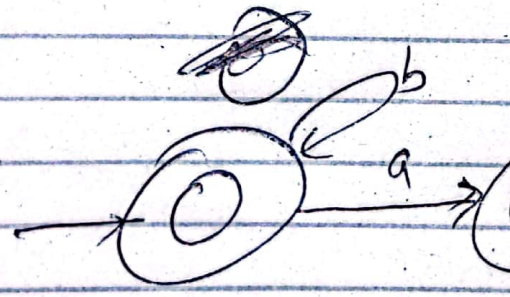
L = {w | w over Σ = {a,b}, w does not contains ab and ends with ba}

does not contains ab

NFA.



DFA



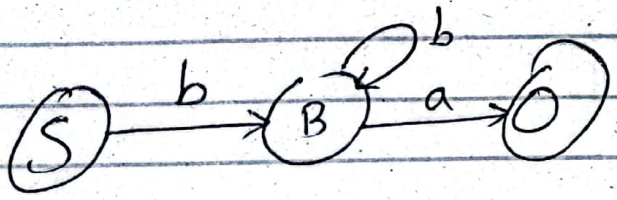
NFA does not contain ab.

Universal set 

ontains

NFA ends with ba

دونوں کا، ب

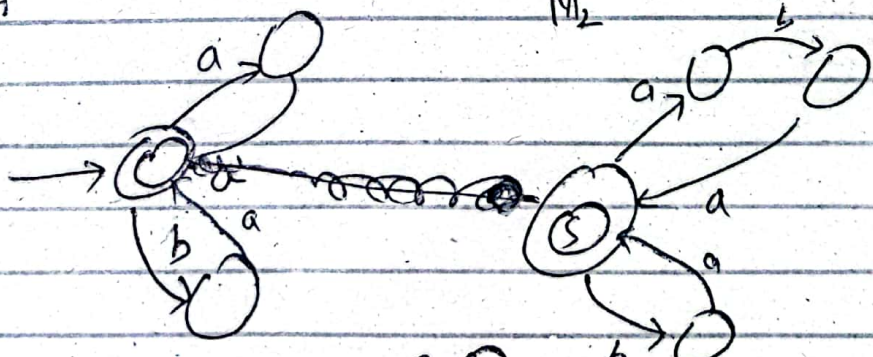


A

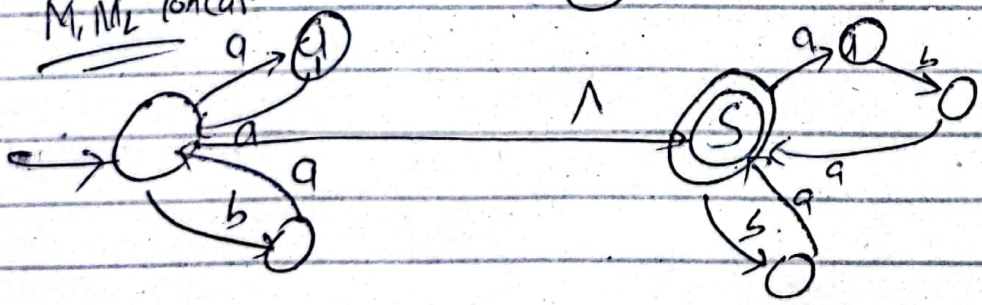
M_1 M_2

$\{aa, ba\}^* . \{aba, bq\}$

M_1 NFA



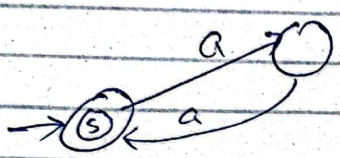
$M_1 M_2$ concat



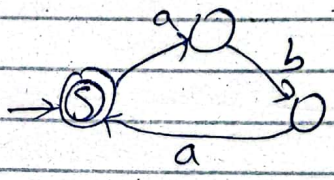
a, (a+b)* a

contains aa or aba

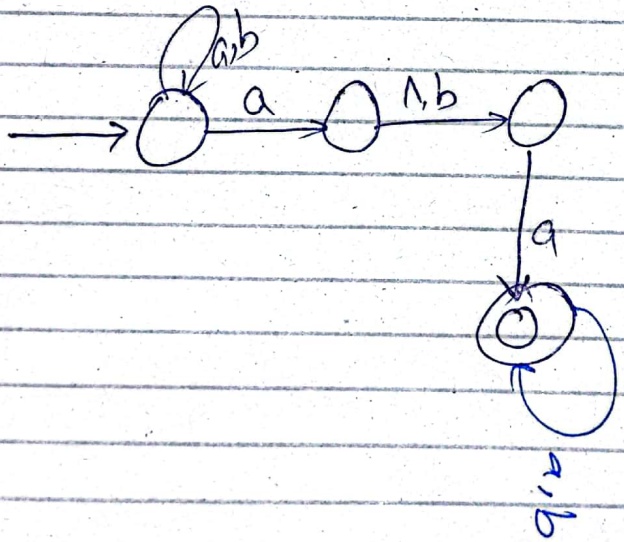
NFA aa



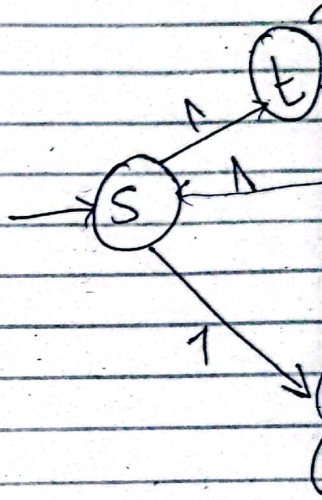
NFA aba



NFA-Λ contains aa or aba

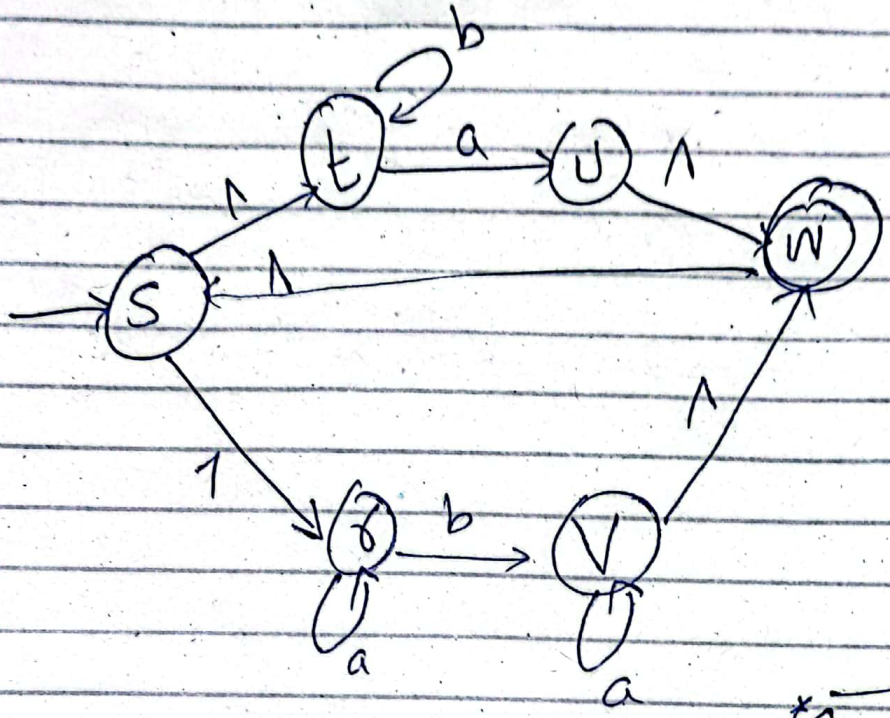


NFA



	a
→S	ϕ
t	{u}
u	ϕ
*w	ϕ
x	{x}
y	{y}

NFA - Λ to NFA



	a	b	Λ
→S	ϕ	ϕ	{t, γ}
t	{u}	{t}	ϕ
u	ϕ	ϕ	{w}
*w	ϕ	ϕ	{s}
γ	{γ}	{v}	ϕ
v	{v}	ϕ	{w}

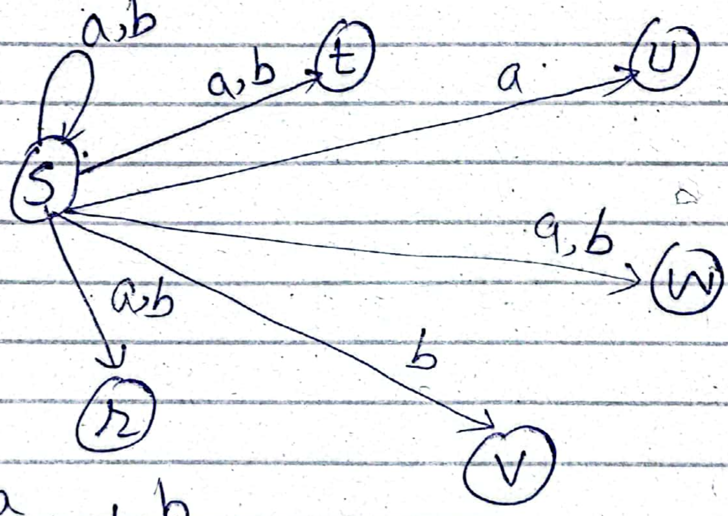
* جتنی NFA-Λ کی States ہیں اتنی ہی NFA میں ہوں گی

NFA-Λ

اگر starting state ← final میں ہے

پھر اگر Λ کی base پہ ہم final state تک پہنچ
جا سکیں starting state سے تو NFA میں ہم
starting state کو بھی final کر دیں گے جو already final
ہیں وہ بھی final ہی رہیں گی۔

اگر starting state ← already ← final ہے تو اس کی ضرورت نہیں



	a	b
s	{u,r,w,s,t}	{t,v,w,s,r}
t	{u,w,t,s,r}	{t}
u	{u,r,w,s,t}	{t,v,w,s,r}
w	{u,r,w,s,t}	{t,v,w,s,r}
r	{r}	{v,w,s,r,t}
v	{u,r,w,s,t}	{t,v,w,s,r}

جب null transition ختم کر دی ہے تو ایک state سے a اور b کے edges بڑھ جائیں گے نکلتے ہیں (ترتیب)

null closure of S

$\Lambda(\{s\}) = \{s, t, r\}$

اگر آگے r پر بھی null کے ذریعے کسی state پر پہنچ رہے ہیں تو وہ بھی آ جائے گی۔ آگے چلتے جائیں گے جب تک Λ ختم نہیں ہوتا۔ Λ empty ہو مطلب آنا بند ہو گئے۔

$\Lambda(\{t\}) = \{t\}$

$\Lambda(\{v\}) = \{v, w, s, t, r\}$

$\Lambda(\{w\}) = \{w, s, t, r\}$

$\Lambda(\{r\}) = \{r\}$

$\Lambda(\{u\}) = \{u, w, s, r, t\}$

Null closure (→ reflexive property)

ایک state سے ہم null کی base پر لے کر جا سکتے ہیں۔ جب بھی کسی state کا null closure لیں تو وہ state اس میں خود include ہو گی۔

$\Lambda(\{s\}) = \{s, t, r\}$ — سب سے a read

$= \{\ \} \cup \{u\} \cup \{r\}$

$= \{u, r\}$

اب دوبارہ {U, γ} کا null closure لیں تو

$\Lambda(\{U, \gamma\}) = \{U, \gamma, W, S, t\}$

U اور γ سے seed کریں گے تو S means سب پہ جائیں گے۔

$\Lambda(\{S\}) = \{S, t, \gamma\} \xrightarrow{b} \{t, V\}$

null closure

↓

$\{t, V, W, S, \gamma\}$

باقی خود کریں۔