Module 1

25 August 2021 20:56

Symbols - basic building block of any lang.

Let us have a language with symbols a & b

possible strings - a, b, aaa, bbb

Represented by Σ

$\{a, b, c\} = \{ab, cb, aab, \ldots\}$ ← infinite language

for c variables alphabets possible :-

$\{A-Z, a-z, 0-9, -\}$

Constraint - should not start with number

Strings - sequence of alphabets

Language - set of all valid strings

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

$L_1 \rightarrow$ set of strings of length 2

$L_1 = \{aa, ab, ba, bb\}$ finite language

$L_2 \rightarrow$ set of strings of length 3

$L_2 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$ finite

$L_3 \rightarrow$ set of all strings that start with 'a'

$L_3 = \{aaa, aab, aba \ldots\ldots aaaaa\}$ ← infinite

Length of string :- $W = abc$, $|W| = 3$

Null String = ε (epsilon) (empty)

Q $\{0, 1, \#\}$

$L_1 \rightarrow$ set of strings of length 2

$L_2 \rightarrow$ set of strings of length 2 ending with #

$L_3 \rightarrow$ set of strings that start with 0 & ends with #

$L_1 = \{00, 01, 0\#, 10, 11, 1\#, \#0, \#1, \#\#\}$

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

$L_3 = \{000\#, 01\#, 0\#\# \ldots\ldots\}$ (infinite)

Union & Concatenation

$A = \{a, b, c\}$ $B = \{0, 1, 2\}$

$A \cup B = \{a, b, c, 0, 1, 2\}$ $\cup \equiv |, +$

$AB = \{a0, a1, a2, b0, b1, b2, c0, c1, c2\}$

& Concatenation

union $\rightarrow A \cup B = A|B = A+B$

$BA = \{0a, 1a, 2a \ldots\ldots\}$

$(A \cup B)(A \cup B) = \{aa, ab, ac, a0, a1, a2, ba, bb, bc, b0, b1, b2, ca, cb, cc, c0, c1, c2, 0a, 0b, 0c, 00, 01, 02, 1a, 1b, 1c, 10, 11, 12,$

2a, 2b, 2c, 20, 21, 22 } length of str = 2

Kleen Closure (*)

concatenated 0 or more times

$A^1 = \{a, b, c\}$ strings of length 1

$AA = A^2$ strings of length 2

$AAA = A^3$ strings of length 3

A^{10}

A^0 – A is concatenated 0 times.

$A^0 = \epsilon$ (empty string)

$A^* = \{\epsilon, a, b, c, aa, ab, ac, \ldots\}$

Positive Closure (+)

concatenated 1 or more time

$A^+ = \{a, b, c, aa, ab, ac, \ldots\}$

$(A \cup B)^* = \{\epsilon, a, b, 0, 1, aa, ab, a0, a1, ba, bb, b0, b1, \ldots aaa, aab \ldots\}$

Grammar

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

Grammar has 4 tuples:-

general convention ↓

1. Non Terminals - represented in CAPS
2. Terminals - smallcase, numbers, operators, etc.
3. Set of productions
4. Start Production - Sentence should start from this.

$V = \{S\}$

$T = \{a, b\}$

$P = \{S \rightarrow aSbS,\ S \rightarrow bSaS,\ S \rightarrow \epsilon\}$

$S = \{S\}$

From S we can proceed to derive other terms hence it is a non terminal. But we cannot derive further expressions from a & b. Hence they are terminals.

Start with Non Terminal

$S \to aSbS \to abSaSbS \to abaSbS \to ababaSbS \to ababab$

scan the sentence

Q. $V = \{S, A, B, C\}$

$T = \{a, b, c\}$

$P = \{ S \to ABC, S \to BC, A \to a, B \to b, C \to c \}$

$S = \{S\}$

$S \to ABC \to aBC \to abC \to abc$

$S \to BC \to bC \to bc$

Q. $A = \{a, b\}$ $B = \{0, 1\}$

$A * B$

$A^* = \{ \epsilon, a, b, aa, ab, ba, bb, \ldots \}$

$A^*B = \{ 0, 1, a0, a1, b0, b1, aa0, aa1, ab0, ab1 \ldots \}$

Valid C Identifiers

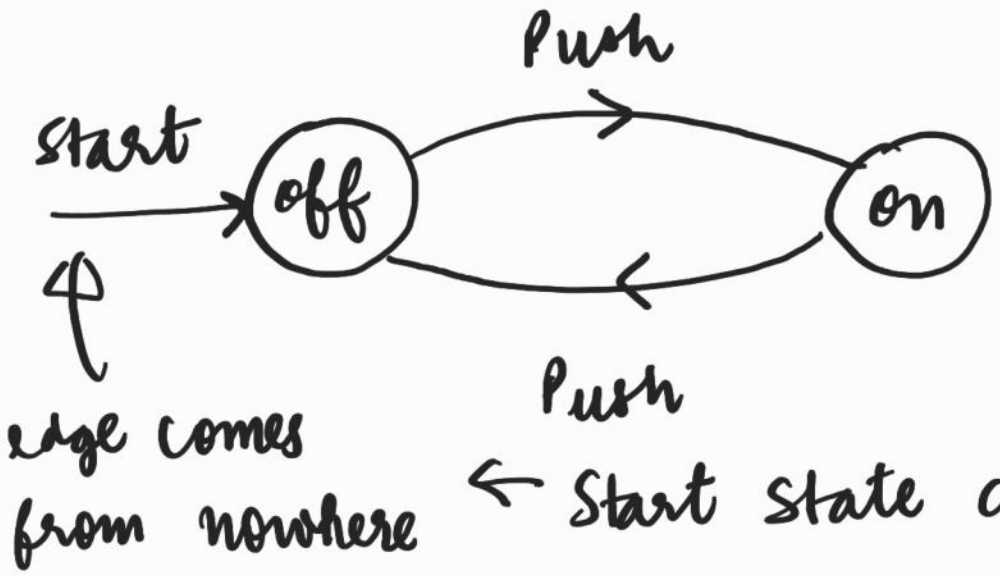
letter = {a, b, c, d..... A, B, C, ... Z, _ }

digit = {0, 1, ... 9}

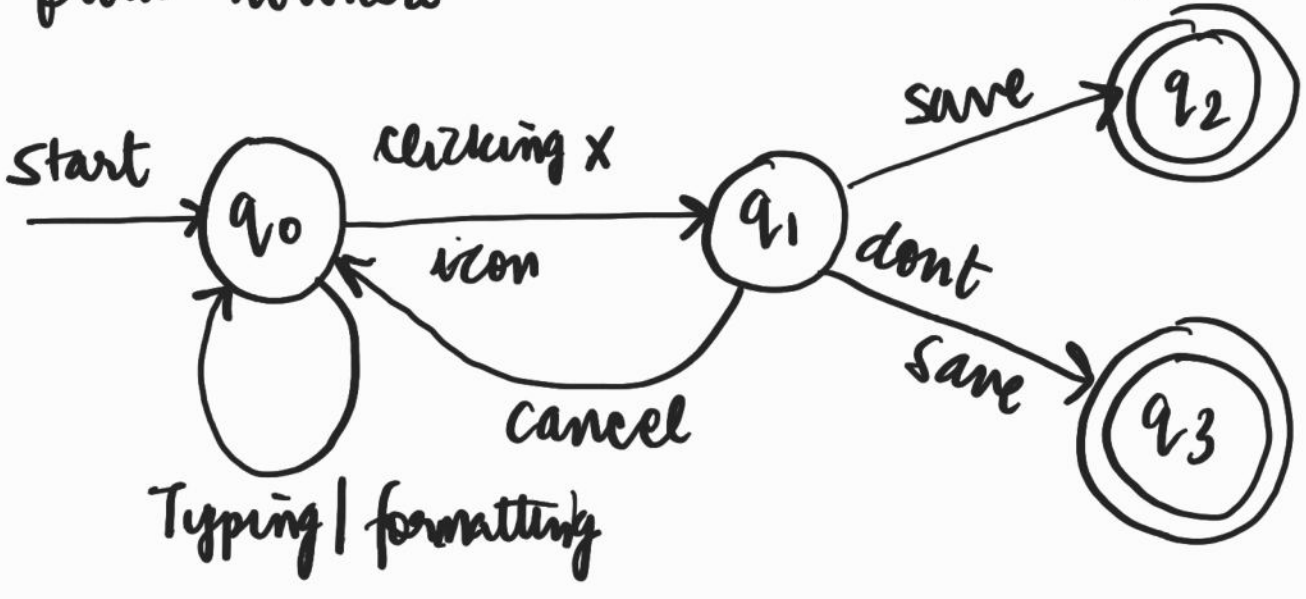
constraint :- first char letter

expression → letter (letter | digit)*

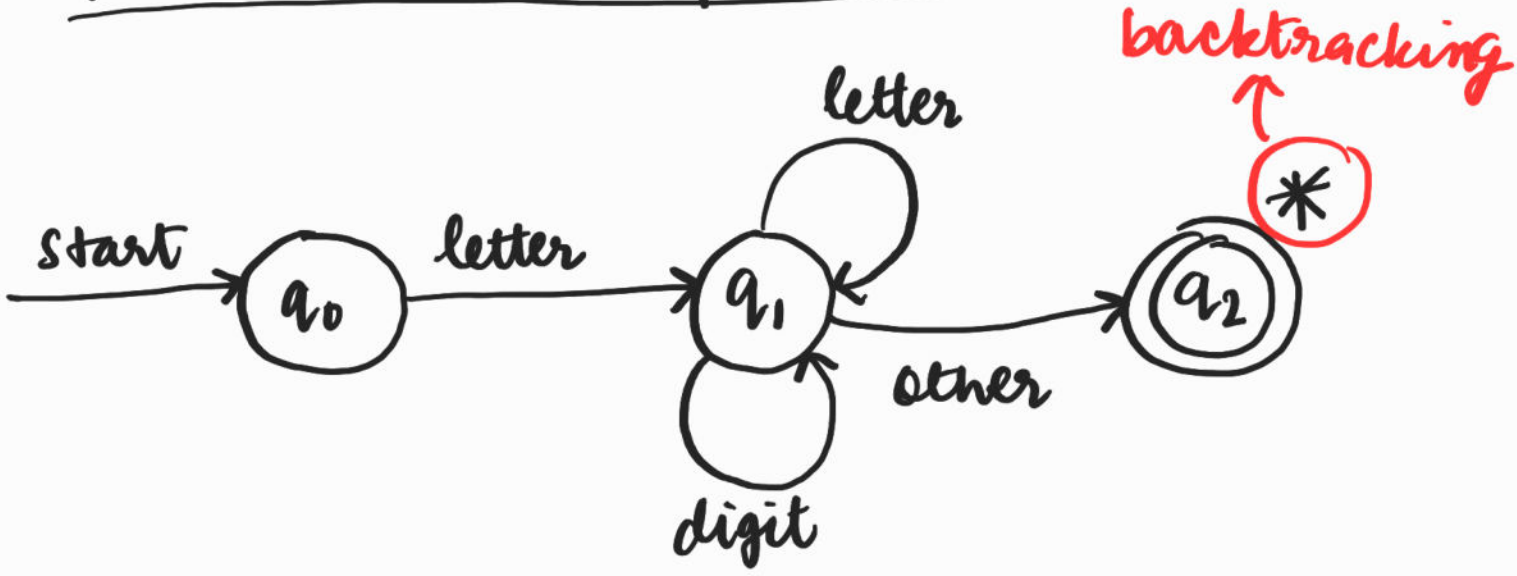
Finite Automata



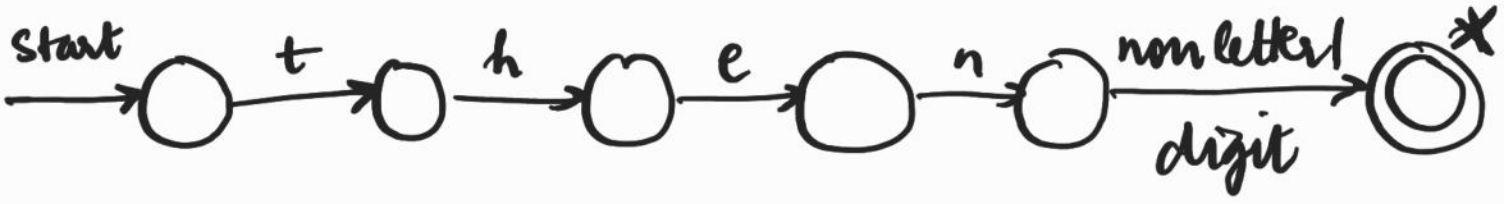
edge comes from nowhere ← Start state designation



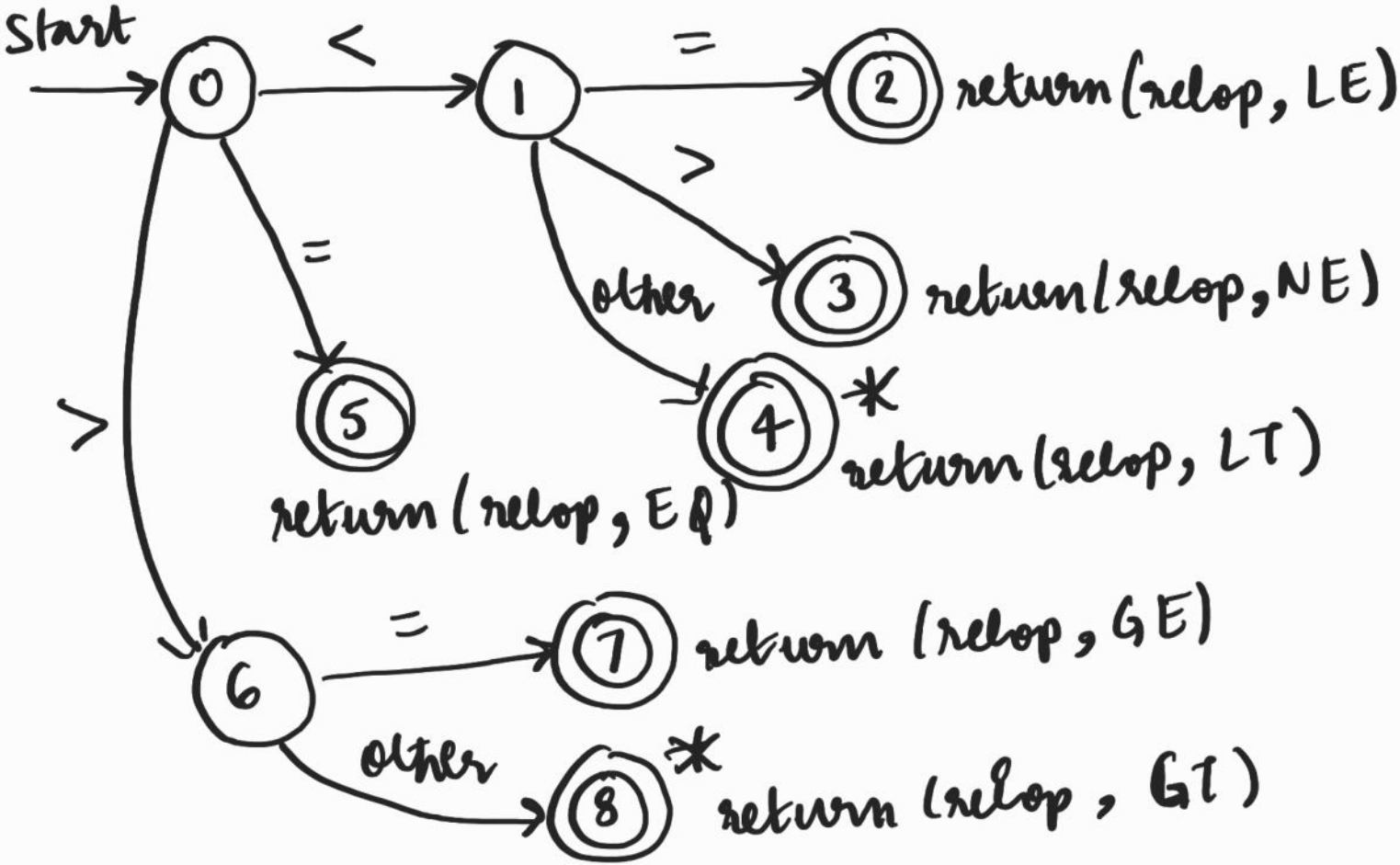
Finite Automata for valid C identifiers



= for keyword 'then'

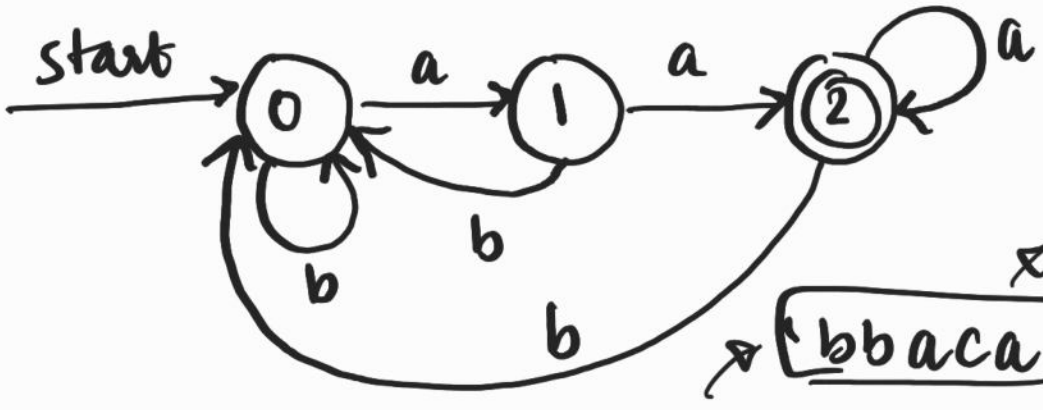


= relational operator



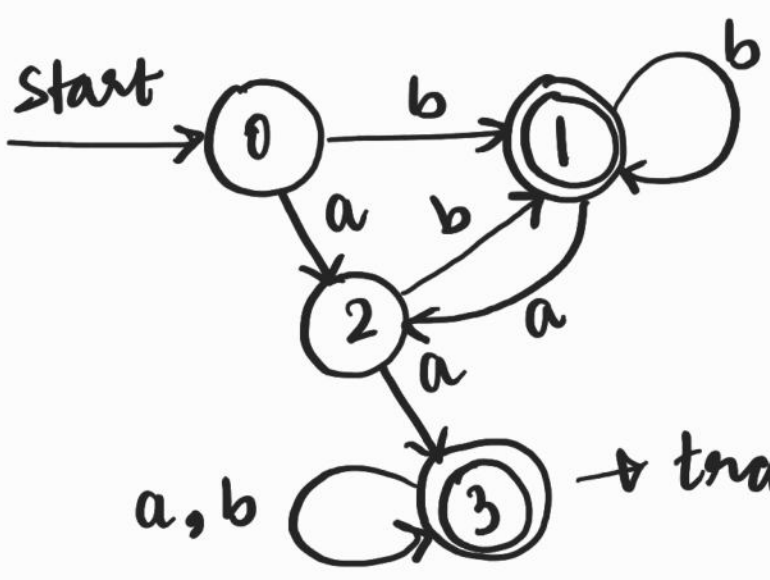
* Finite Automata accepting the language of strings ending in 'aa'. $(a|b)^*aa$

aa, abaa, ababaa, aaaaaba



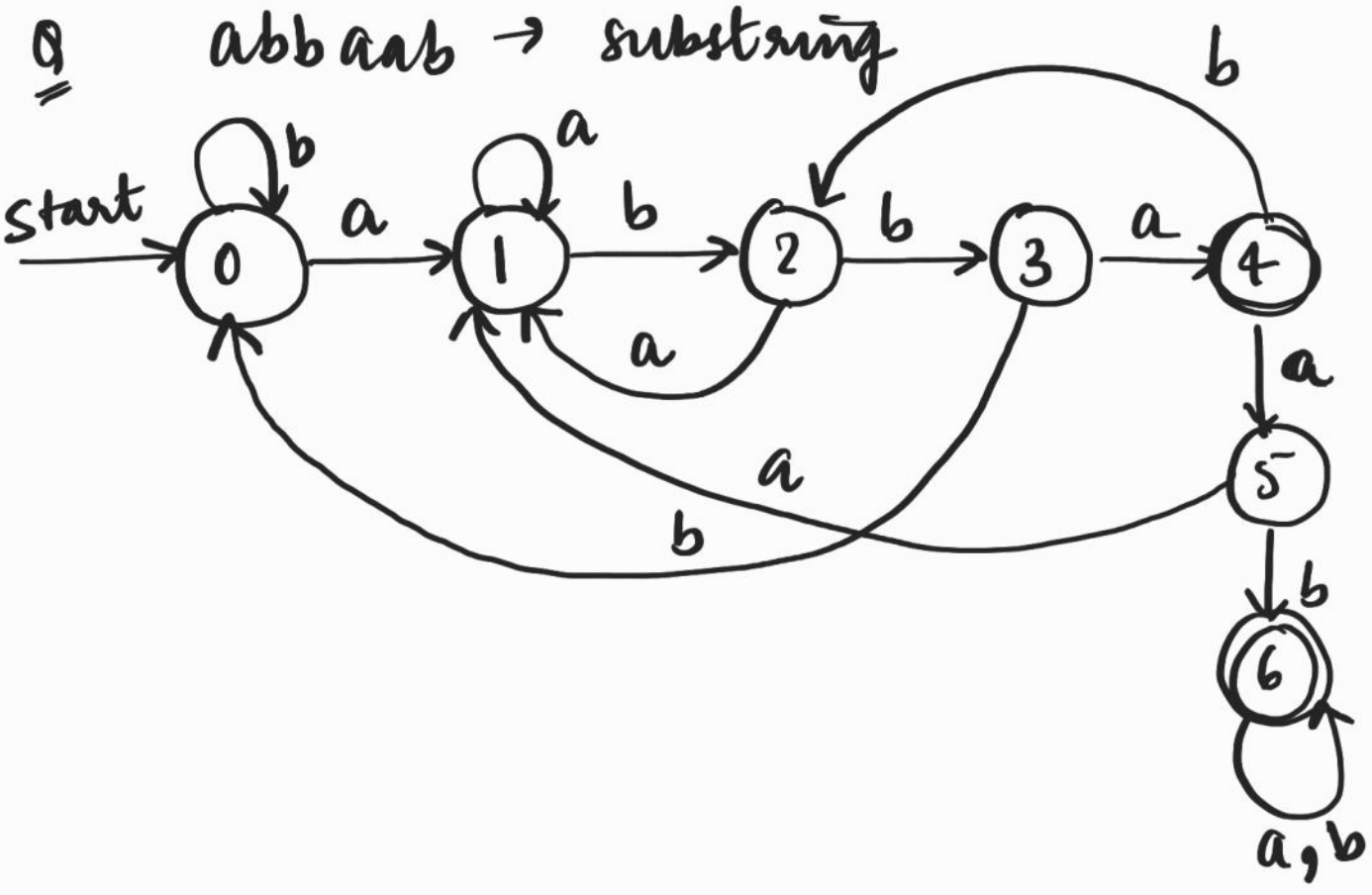
Q Finite Automata accepting the language of strings ending in b but not containing 'aa'

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



trapped state :- once it is reached you cannot reach the final state.

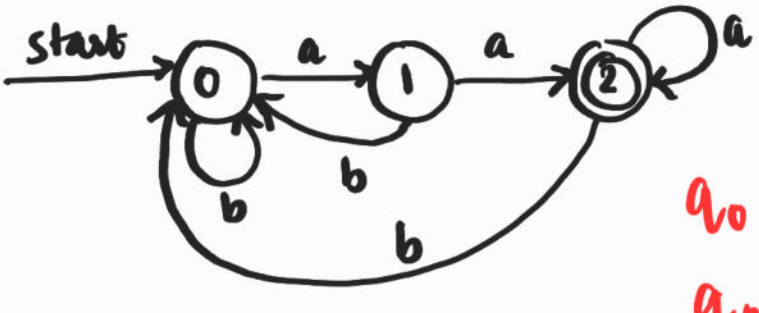
Q abbaab → substring



Deterministic Finite Automata (DFA)

Finite set of states Q
Finite non-empty set ∑
Transitions function δ: Q x ∑ -> Q → transition to only single state
Start state q_0 in Q
Set of final states F contained in Q

Deterministic Finite Automata is represented as 5 tuples $(Q,\sum,\delta,q_0,F)$



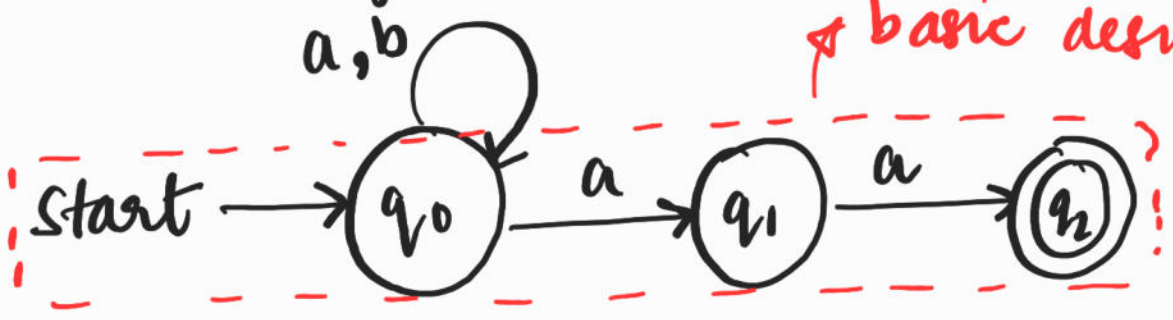
Q = 3 states
$\sum = \{a, b\}$
$q_0 \times a \rightarrow Q_1$ which $\in Q$
$q_0 \in Q$
F = { set of final states }

Non Deterministic Finite Automata (NFA)

Finite set of states Q
Finite non-empty set Σ
Transitions function δ: Q x Σ -> 2^Q ↗ multiple states transitions possible.
Start state q_0 in Q
Set of final states F contained in Q

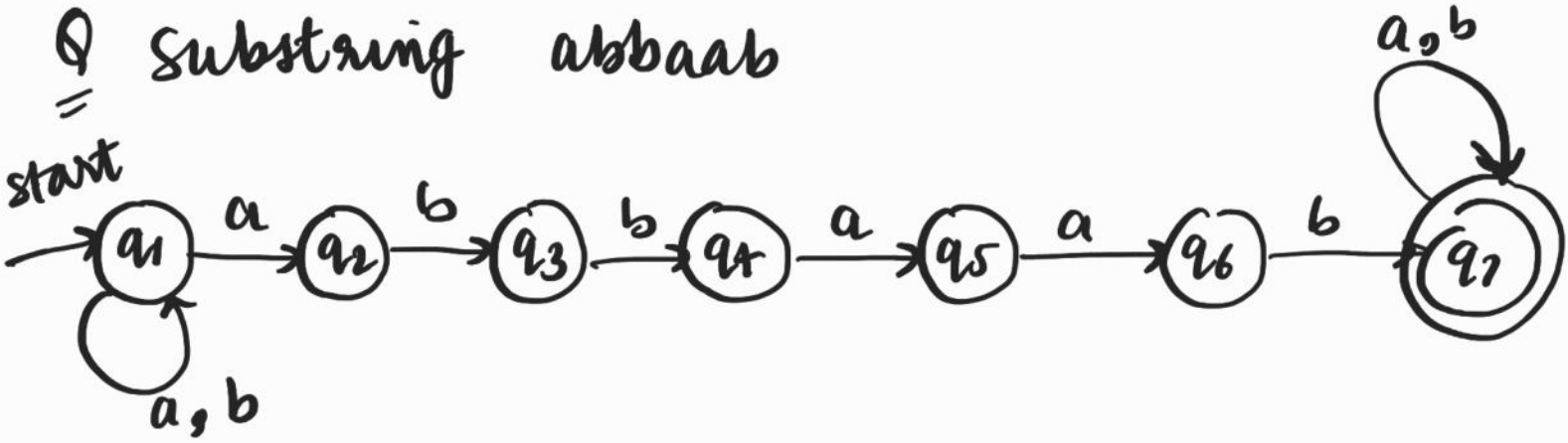
Non-Deterministic Finite Automata is represented as 5 tuples $(Q, \Sigma, \delta, q_0, F)$

NFA accepting the language of strings ending in 'aa' (Regular exp :- (a|b)*aa)



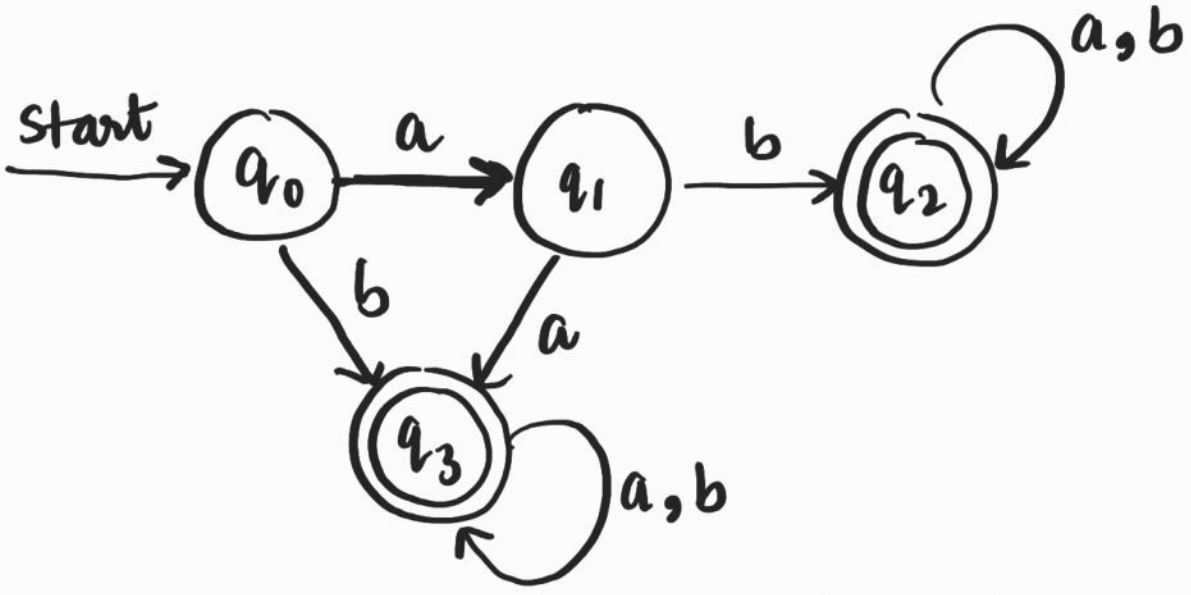
Non Deterministic we cannot design a machine. It is a state to determine all the possible cases. For research purposes only.

Q Substring abbaab

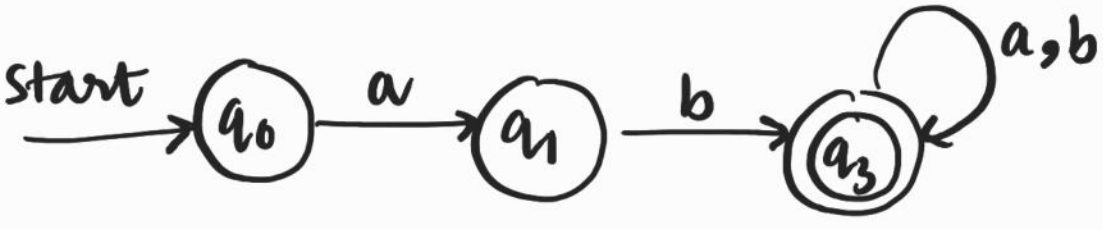


Q. $ab(a|b)^*$

Should start with ab



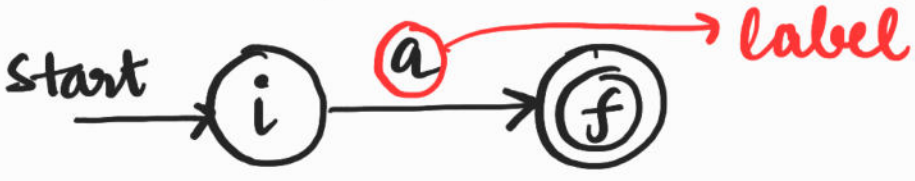
*NFA without epsilon transition



Construction of NFA from a Regular Expression - McNaughton Yamada Thompson Algorithm

1. Basis Rule:- (for input symbols)
for the expression ε, NFA is :-

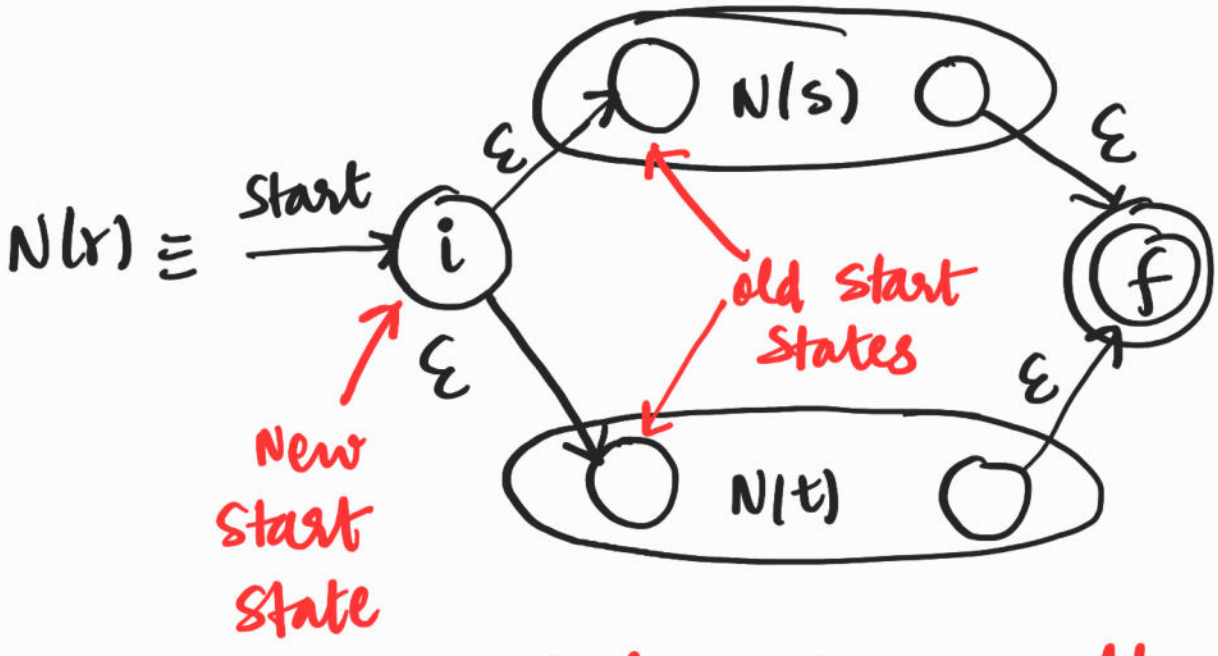
for the expression a, NFA is :-



2 Induction Rule:- (for operators)

(a) let N(s) and N(t) are NFA's for regular expressions 's' and 't' respectively. For union of 2 regular expressions $r = s|t$

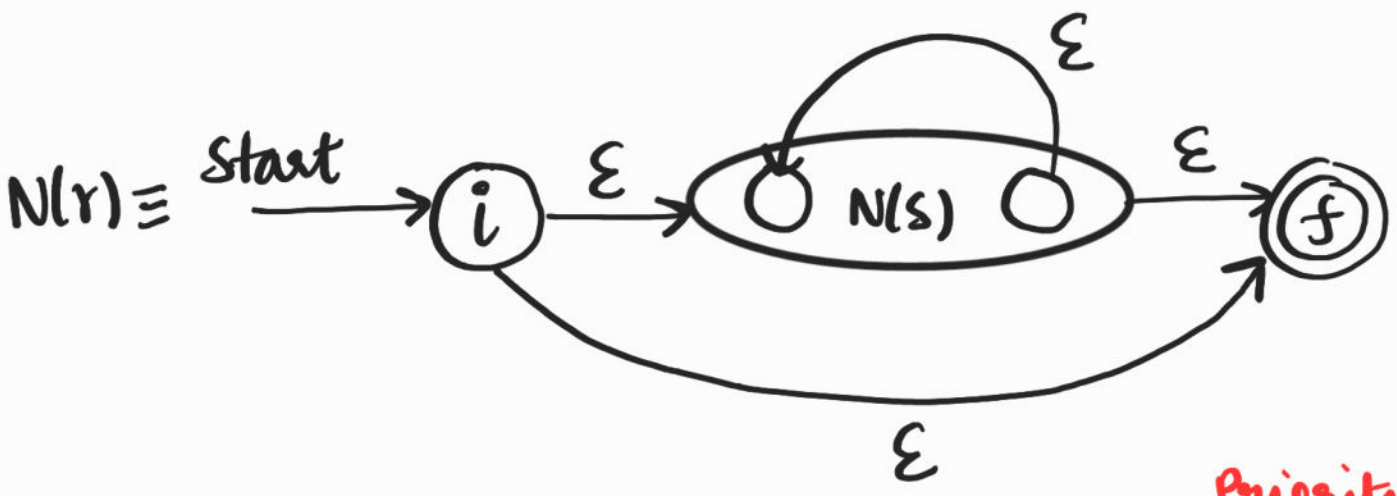
NFA



epsilon transition from New to old start states.

(b) For concatenation of 2 regular expressions, $r = st$. NFA

(c) Kleen closure, $r = s*$



Priority
( )
*
concat
∪

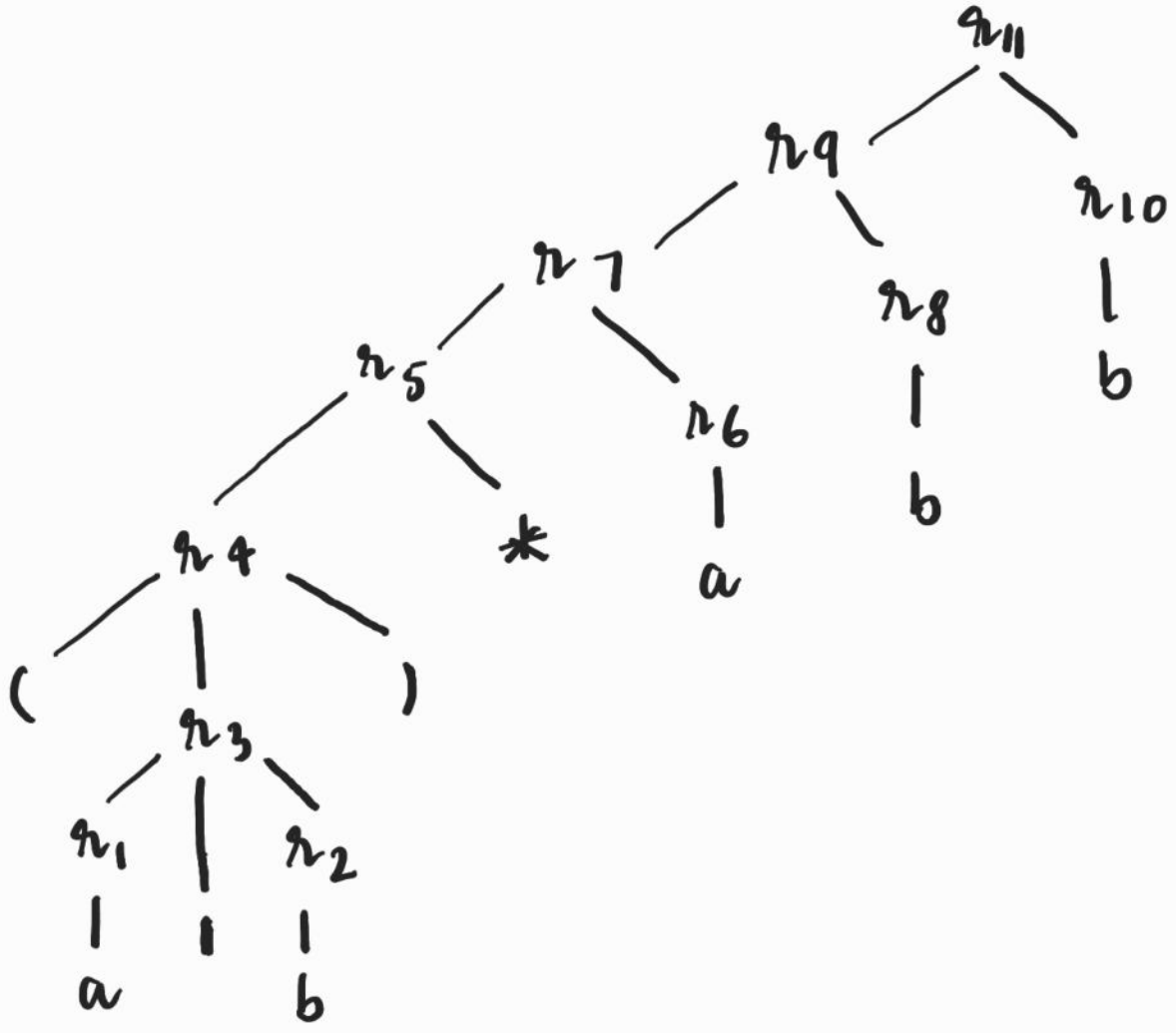
(d) Brackets

For $r = (s)$, then $N(r) = N(s)$

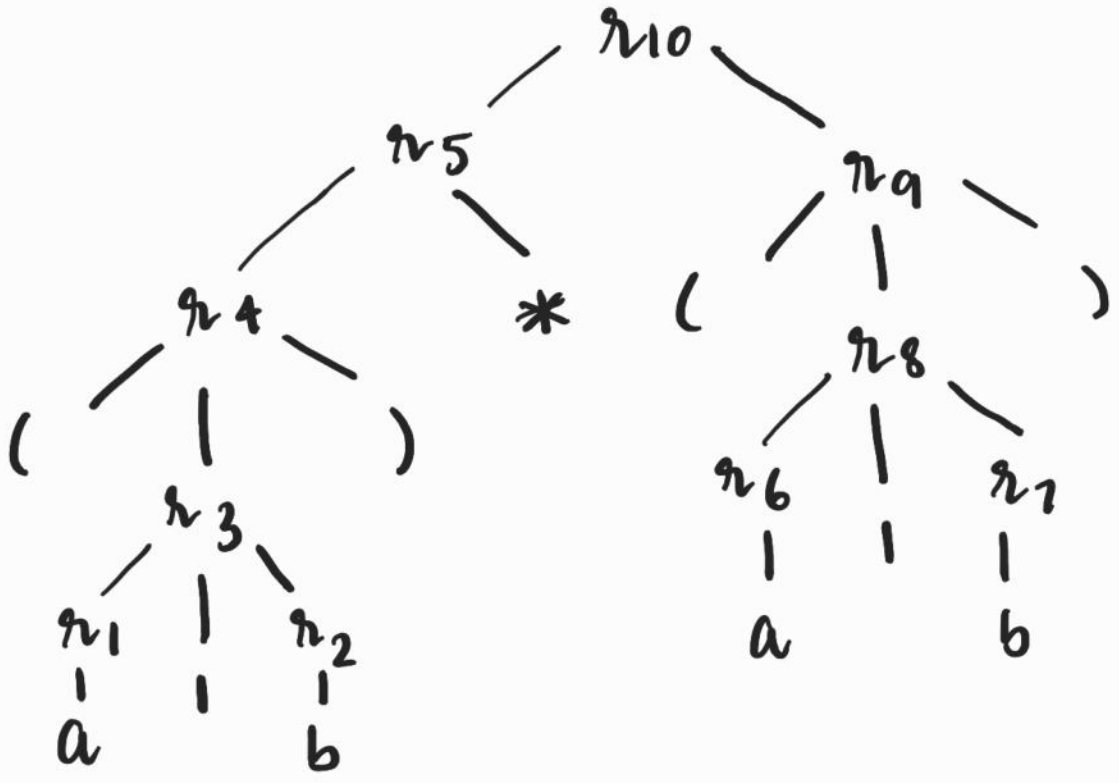
$\therefore$ NFA N(s) is N(r)

Q $\equiv$ $(a|b)^*abb$

Parse Tree of $(a|b)^*abb$



Q (a|b) * (a|b)

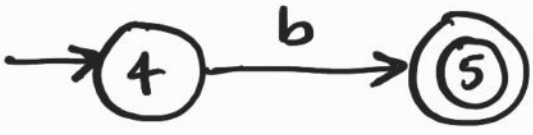


Q (a|b)* abb

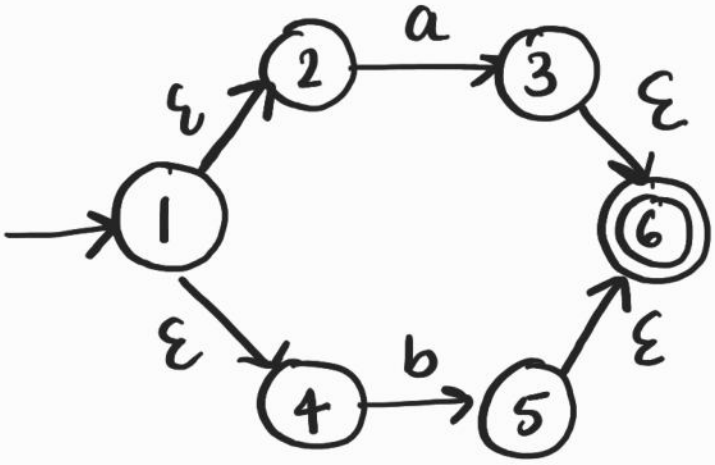
r_1 ≡

(Basis Role)

r_2 =



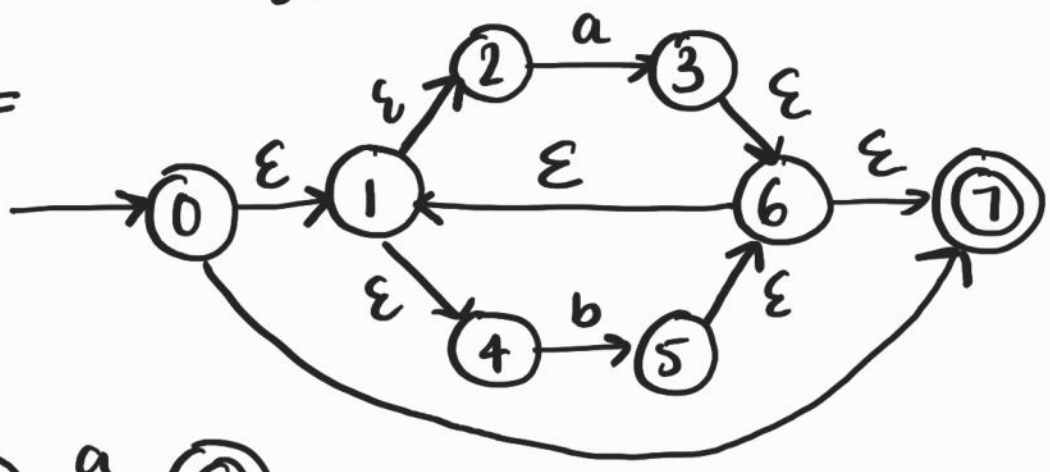
$r_3 = r_1 | r_2$ ≡



$r_4 = (r_3)$ this from Induction Rule

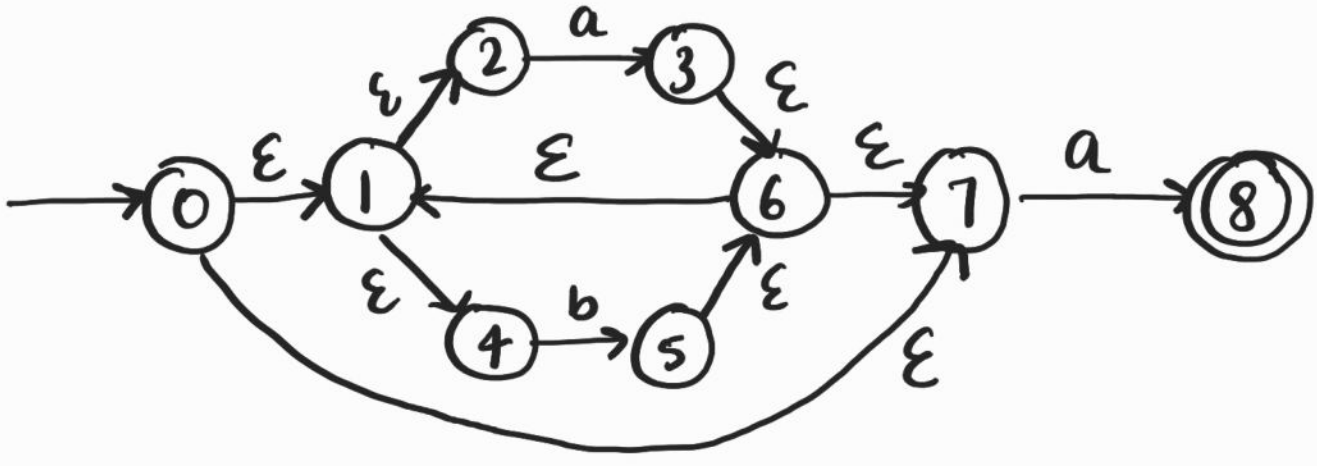
NFA(r_4) = NFA(r_3)

$r_5 = r_4^* =$



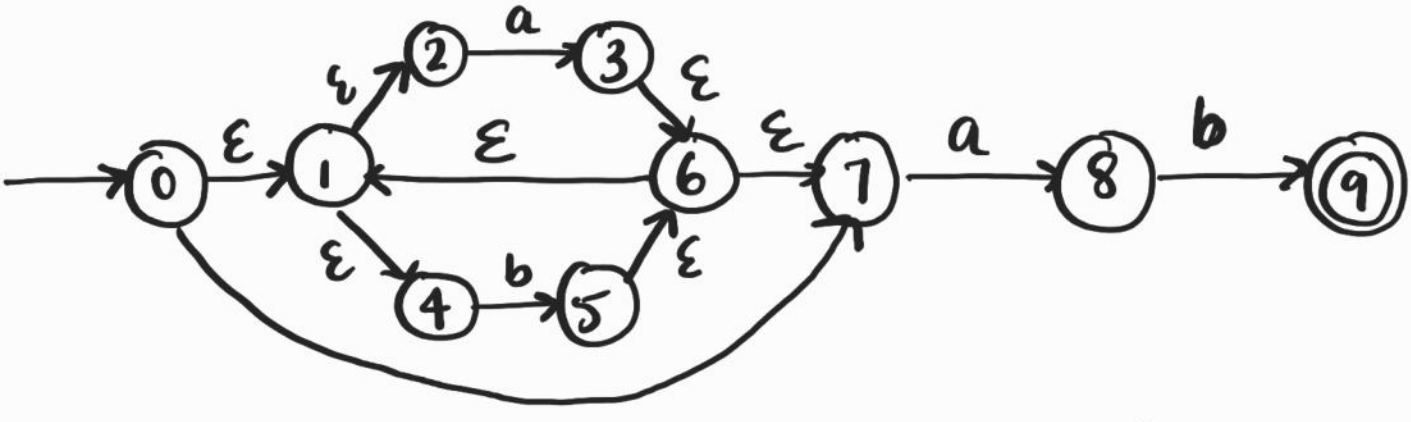
$r_6 = a$

$r_7 = r_5 r_6$

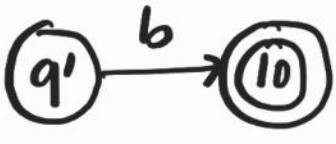


$r_8 = b$

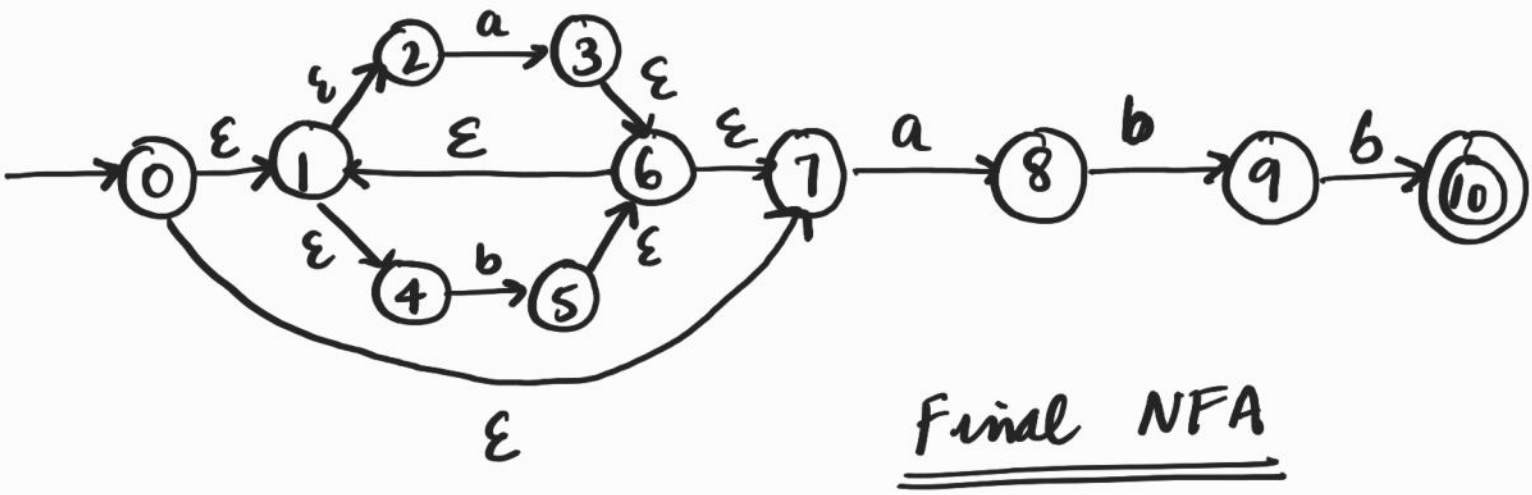
$r_9 = r_7 r_8$



$r_{10} =$



$r_{10} = b$ $r_{11} = r_9 r_{10}$

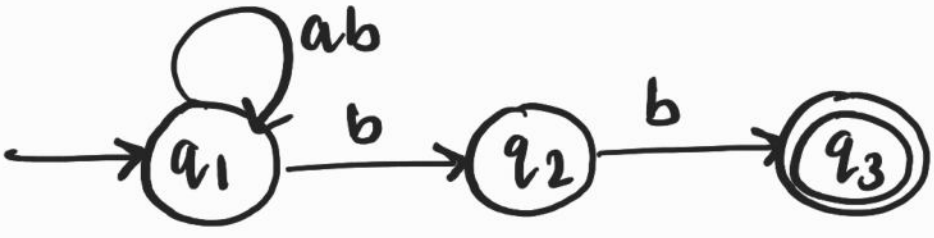


Final NFA

Since NFA & DFA are equal in strength hence we should be able to obtain DFA from NFA.

NFA with ε transition is called epsilon NFA.

conversion of Non ε NFA to its DFA

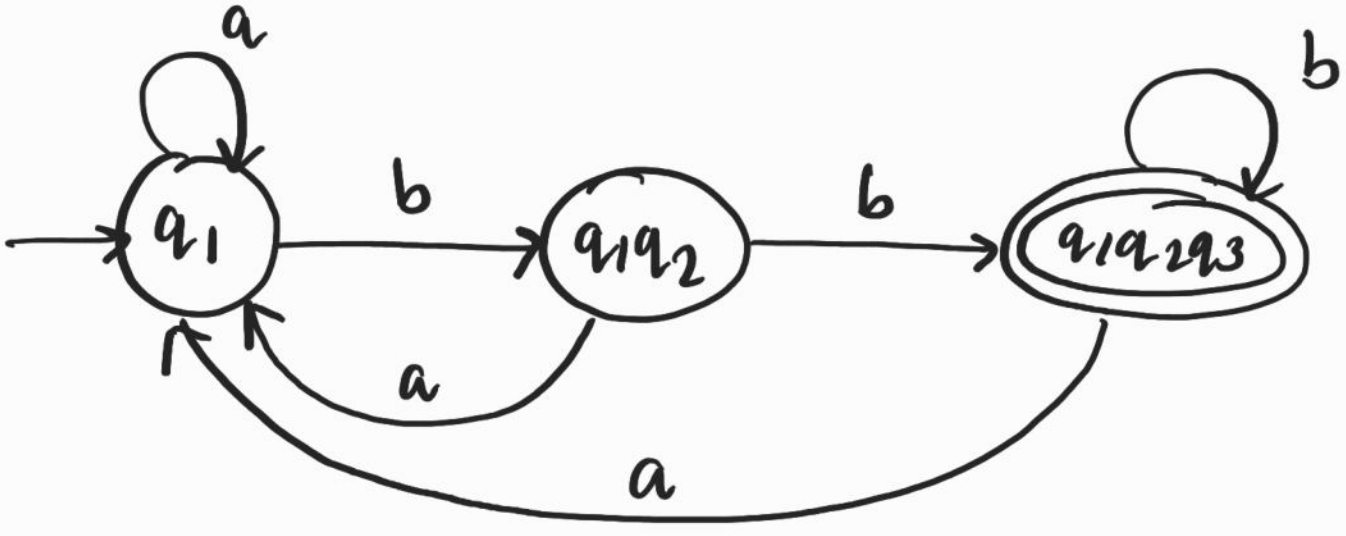


→ Construction of NFA table

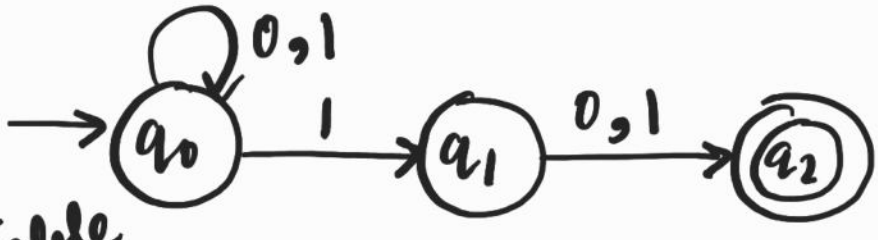
δ	a	b
→ q_1	q_1	$\{q_1, q_2\}$
q_2	{ }	q_3
final state * q_3	{ }	{ }

→ Construction of DFA table

δ	a	b
→ q_1	q_1	$[q_1q_2]$ → single state
$[q_1q_2]$	$q_1 \cup \{\} = q_1$	$q_1q_2 \cup q_3 = [q_1q_2q_3]$
* $[q_1q_2q_3]$	$q_1 \cup \{\} \cup \{\} = q_1$	$q_1q_2q_3$



Q



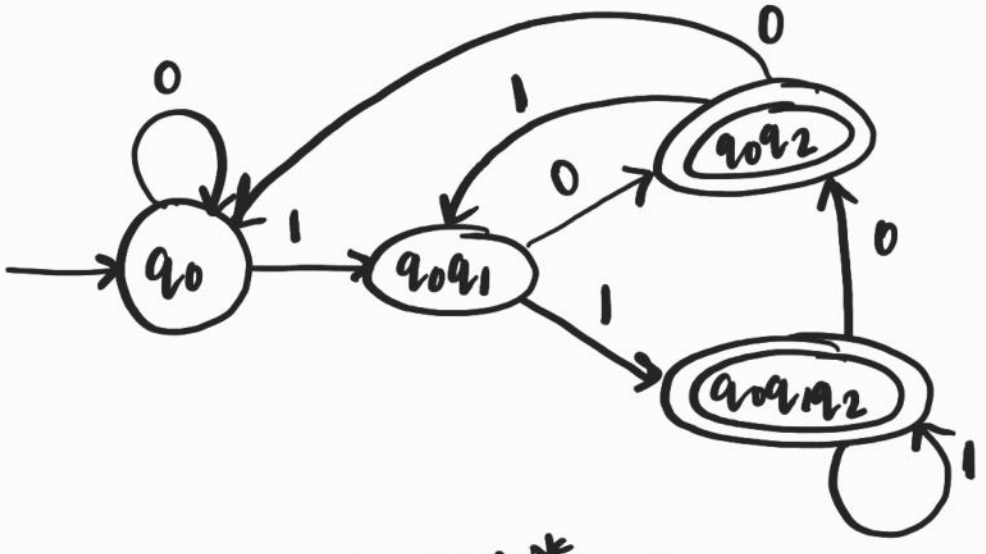
NFA Table

δ	0	1
→ q_0	q_0	$\{q_0, q_1\}$
q_1	q_2	q_2
* q_2	$\{\}$	$\{\}$

DFA Table

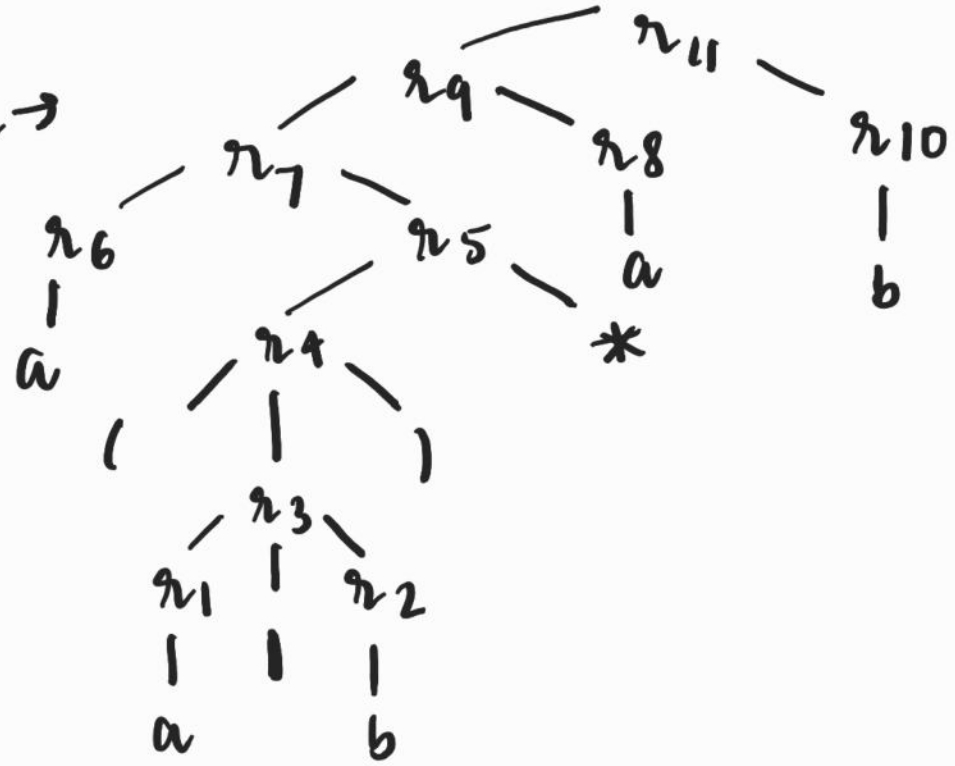
δ	0	1
$\rightarrow q_0$	q_0	$[q_0q_1]$
$[q_0q_1]$	$[q_0q_2]$	$[q_0q_1q_2]$
$*\ [q_0q_2]$	q_0	$[q_0q_1]$
$*\ [q_0q_1q_2]$	$[q_0q_2]$	$[q_0q_1q_2]$

→ Doubt exact same state ✱



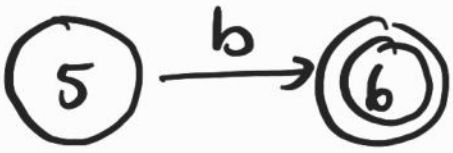
Q NFA for $a(a|b)^* ab$

Parse Tree →

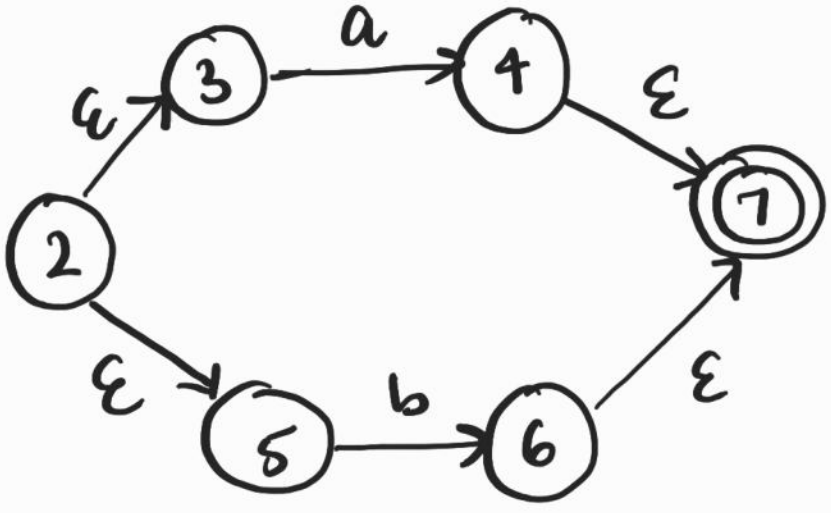


NFA for r_1 (Basis Rule)

NFA for r_2 (Basis Rule)

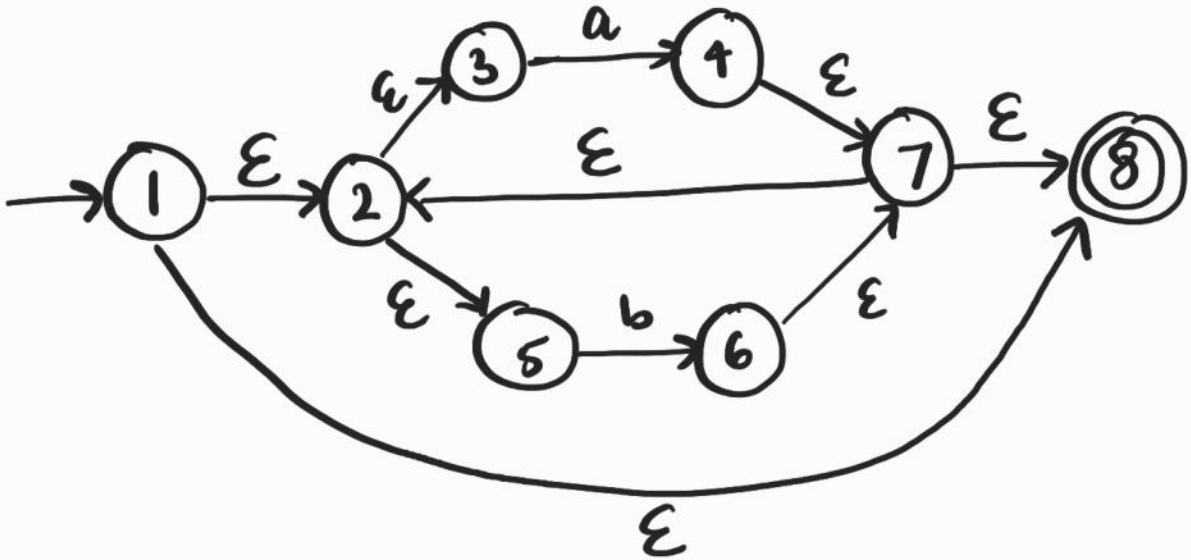


NFA for r_3 (Induction Rule - Union)

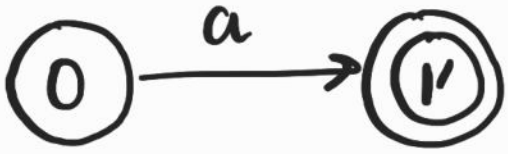


NFA for r_4 = NFA for r_3

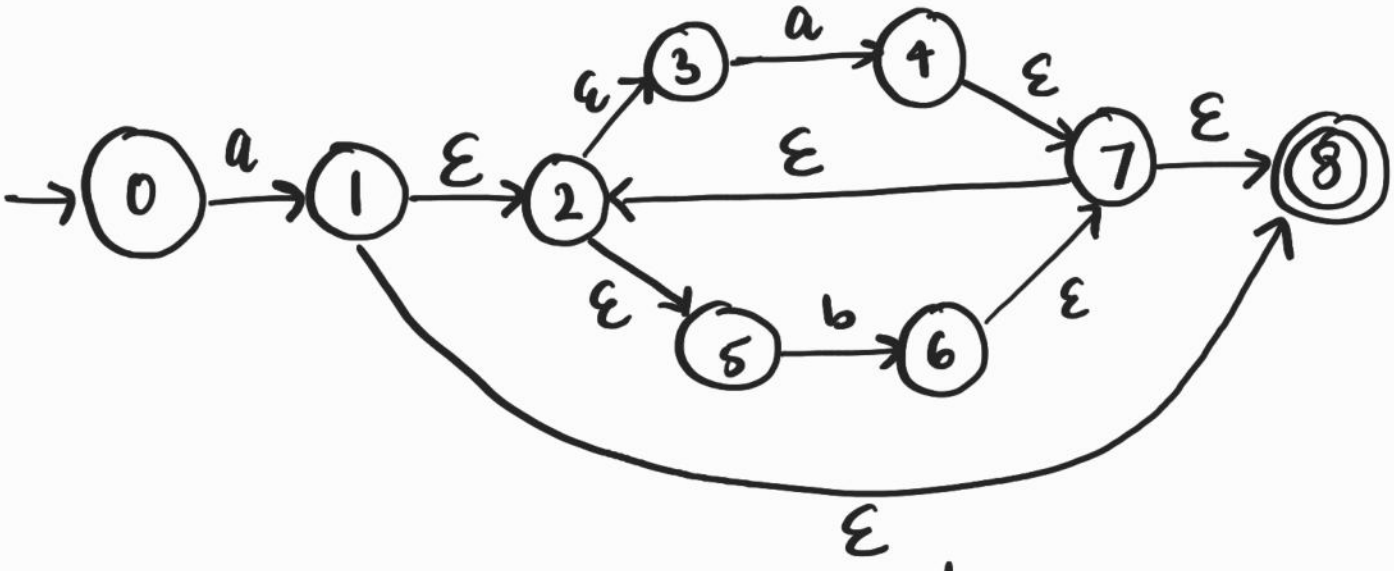
NFA for r_5 (Induction Rule - Kleen Closure)



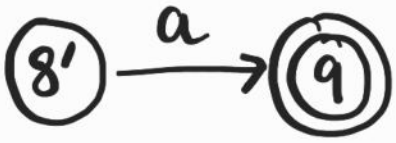
NFA for r_6 (Basis Rule)



NFA for r_7 (Induction Rule)

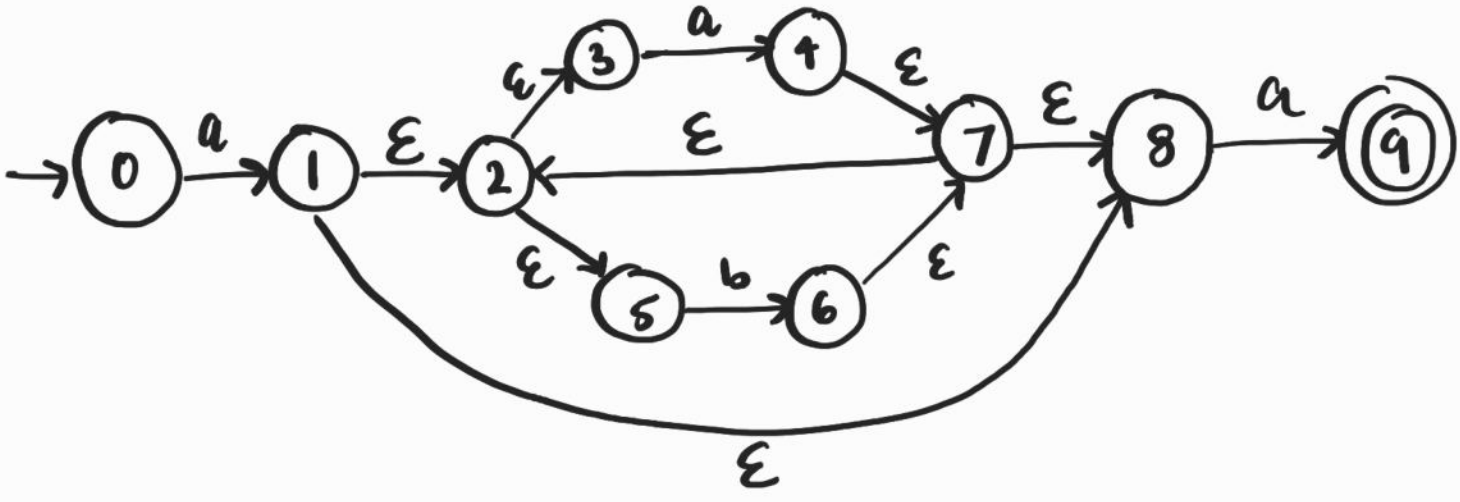


NFA for r_8 (Basis Rule)

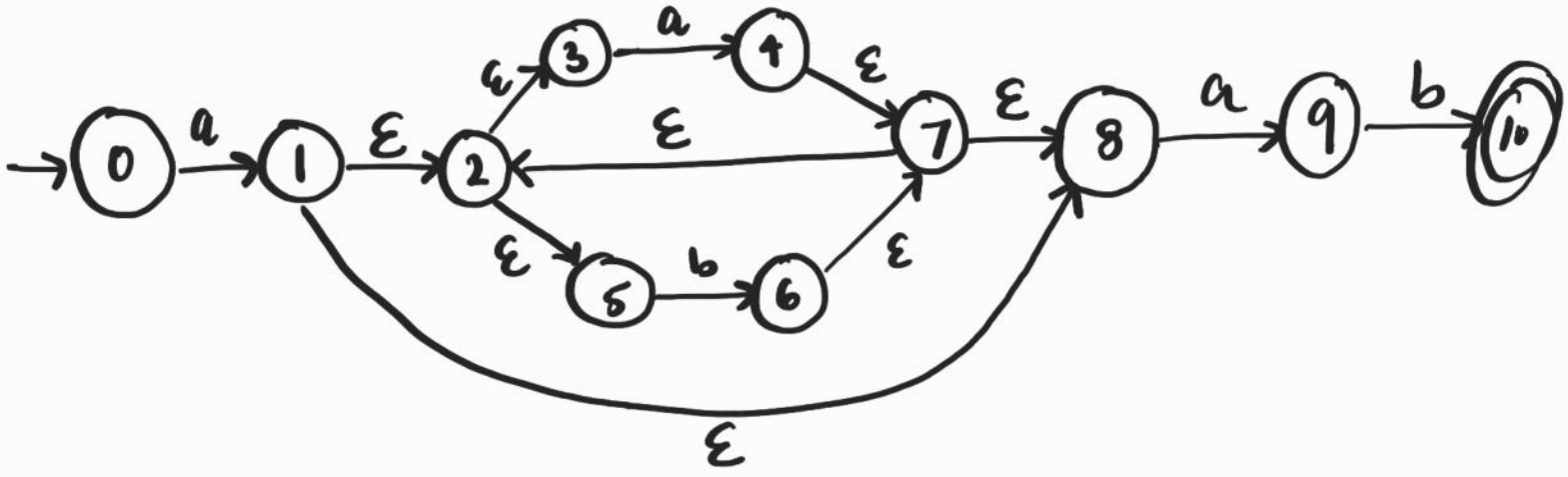


r_{10}

NFA for r_9



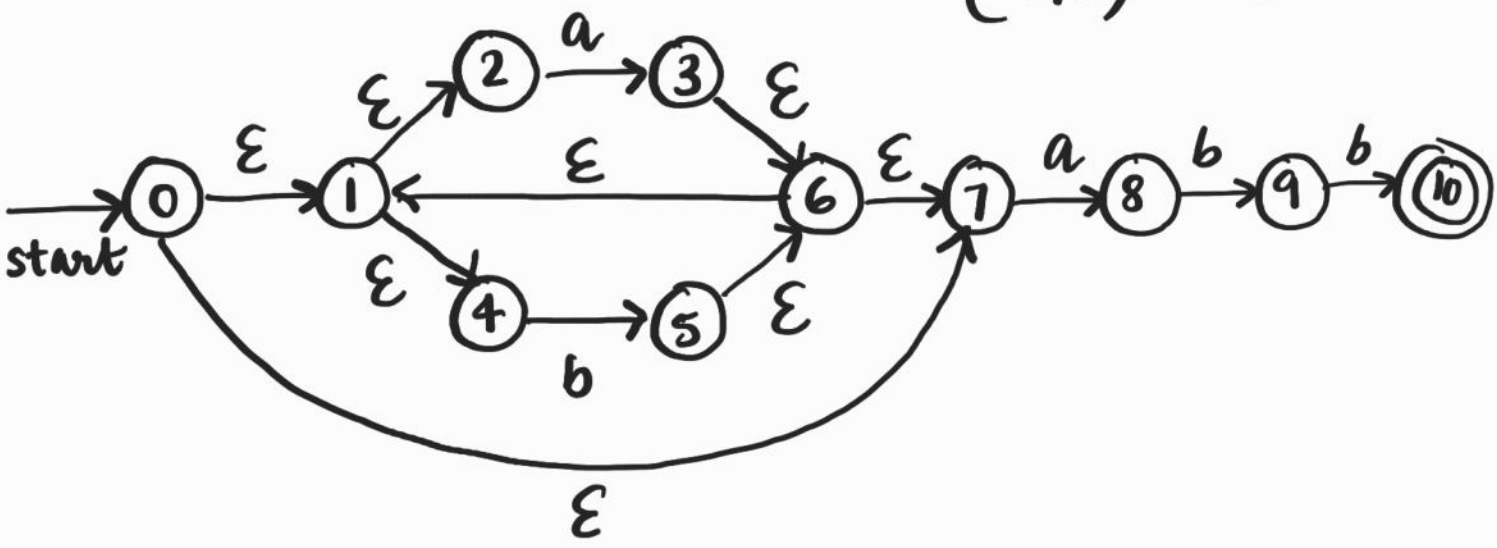
NFA for $a^*|b$



Conversion of ε-NFA to DFA

tent presented inside the header <h1> tag on the webpage. Assume that the h1 tag

$(a|b)^*abb$



* Subset construction algorithm

a) ε closure (s) - set of NFA states reachable from NFA state s on Σ transition alone.

b) ε closure (T) - set of NFA states reachable from some NFA state s in set T on Σ transition alone.

c) Move (I, a) - set of NFA states to which there is a transition on input symbol 'a' from some state S in I.

ε-closure(0) = {0, 1, 2, 4, 7} ← start state of DFA ≡ A

every state is reachable to itself on empty transition (DFA transition)

Dtran [A, a] = ε-closure (move (A, a))

move (A, a) = {3, 8}

ε-closure ({3, 8}) = {3, 6, 1, 2, 4, 7, 8}

= {1, 2, 3, 4, 6, 7, 8}

New State (B)

Dtran [A, b] = ε-closure (move (A, b))

move (A, b) = {5}

ε-closure ({5}) = {6, 7, 1, 2, 4, 5}

= {1, 2, 4, 5, 6, 7} ≡ C

Dtran [B, a] = ε-closure (move (B, a))

move(B,a) = {3, 8}

ε-closure({3,8}) = {1, 2, 3, 4, 6, 7, 8} (B)

Dtran [B,b] = ε-closure [move (B,b)]

move (B,b) = {5, 9, 10} → Doubt why not 10?

ε({5,9,10}) = {5,6,1,7,2,4,9,10}

= {1,2,4,5,6,7,9,10} ≡ D

Dtran [C,a]

move (C,a) = {3,8}

ε-closure ({3,8}) ≡ B

Dtran [C,b]

move(C,b) = {5}

ε-closure ({5}) ≡ C

Dtran [D,a]

move (D,a) = {3,8}

ε-closure ({3,8}) ≡ B

Dtran [D,b]

move (D,b) = {5,10}

ε-closure({5,10})

= {5,6,7,1,2,4,10}

= {1,2,4,5,6,7,10}

≡ E

Dtran [E,a]

move (E,a) = {3,8}

ε-closure ({3,8}) ≡ B

Dtran [E,b]

Move (E,b) = {5}

ε-closure ({5}) ≡ C

STATES

A ≡ {0,1,2,4,7}

B = {1,2,3,4,6,7,8}

C ≡ {1,2,4,5,6,7}

D ≡ {1,2,4,5,6,7,9}

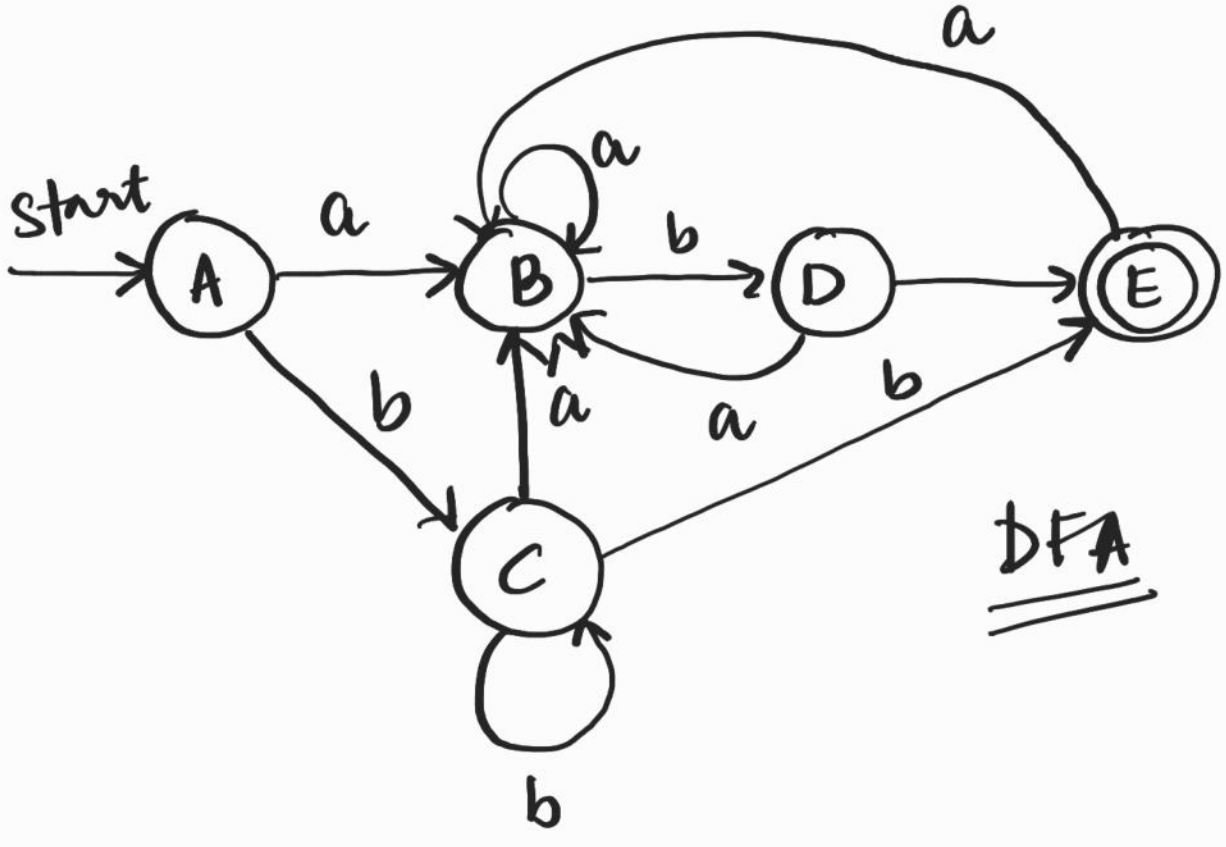
E ≡ {1,2,4,5,6,7,10}

DFA State	a	b
→ A	B	C
B	B	D
C	B	C
D	B	E
* E	B	C

only state has '0'; start state of NFA

only state which has 10 in it.

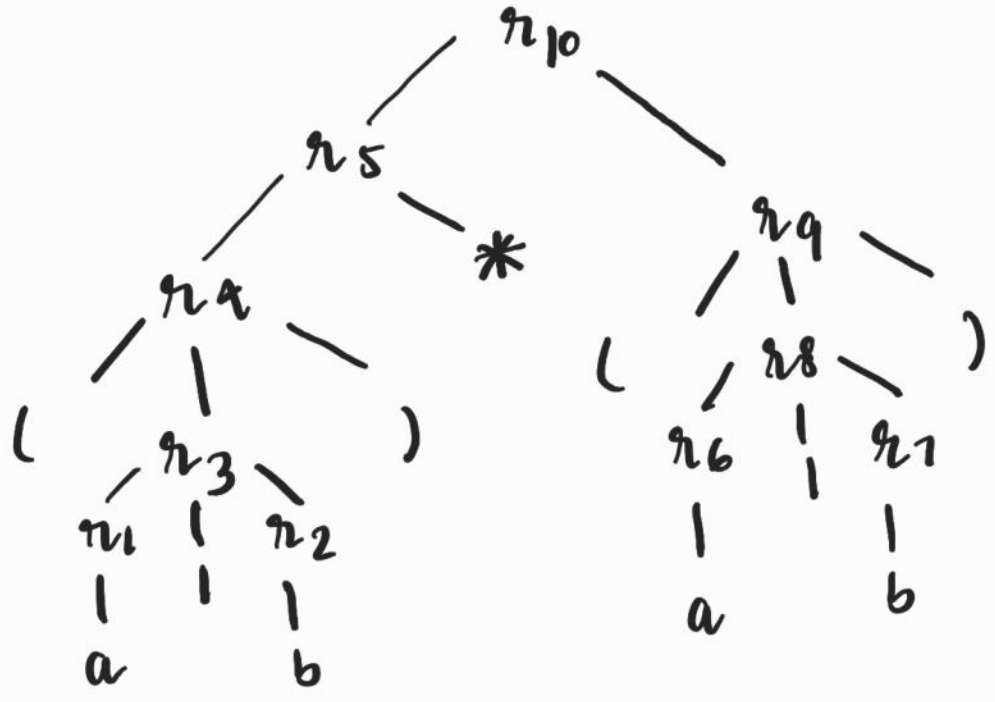
thus Final State.



DFA

8. $(a|b)^*(a|b)$ (ε-NFA to DFA)

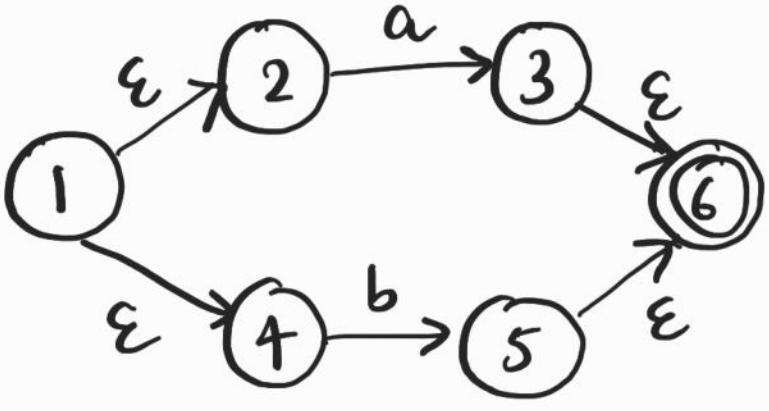
Parse Tree :-



NFA for r_1 :-

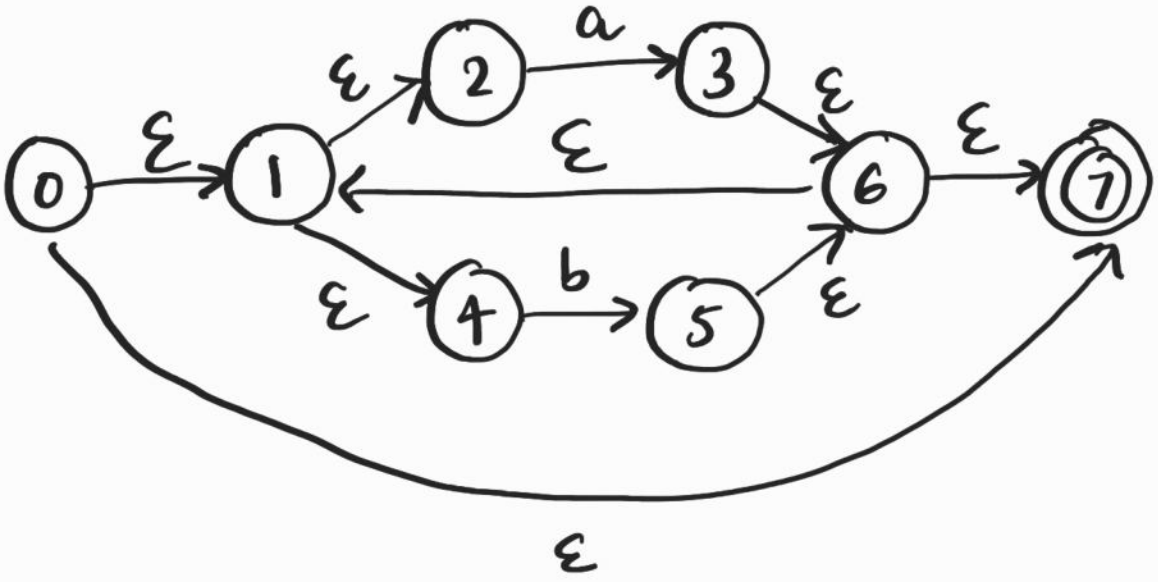
NFA for r_2 :-

NFA for r_3 :-



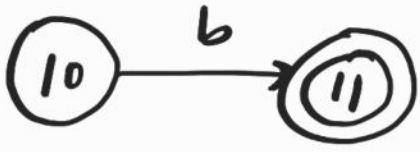
NFA for $r_4 \equiv$ NFA for r_3

NFA for r_5

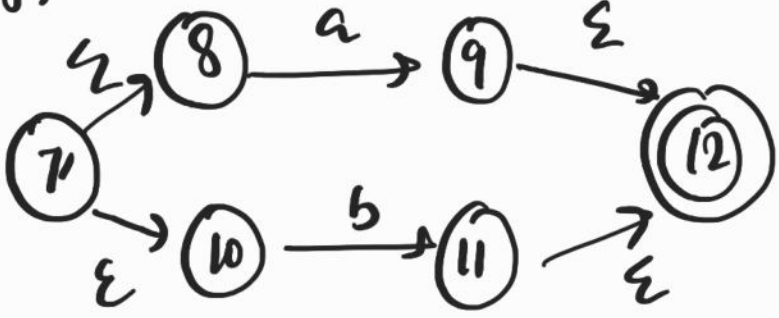


NFA for r_6 :-

NFA for r_7 :-

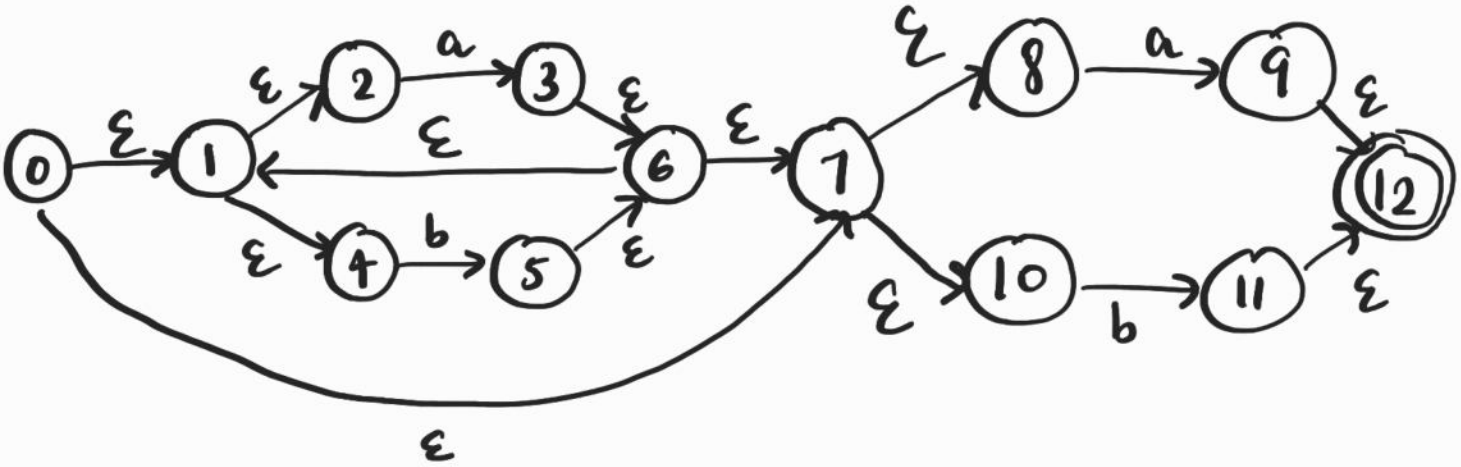


NFA for r_8 :-



NFA for $r_9 \equiv$ NFA for r_8

NFA for $r_{10} \equiv$



Conversion to DFA

ε-closure $(\{0\}) = \{0, 1, 2, 4, 7, 8, 10\} \equiv A$

$DTran[A, a] = \varepsilon\text{closure}(mov(A, a))$

$mov(A, a) = \{3, 9\}$

$\varepsilon\text{closure}(\{3, 9\}) = \{3, 6, 7, 1, 2, 4, 12, 9, 8, 10\}$

$= \{1, 2, 3, 4, 6, 7, 8, 9, 10, 12\} \equiv B$

$DTran[A, b] =$

$mov(A, b) = \{5, 11\}$

$\varepsilon\text{closure}(\{5, 11\}) = \{5, 6, 7, 1, 2, 4, 8, 10, 12, 11\}$

$= \{1, 2, 4, 5, 6, 7, 8, 10, 11, 12\} \equiv C$

$DTran[B, a]$

$mov(B, a) = \{3, 9\}$

$\varepsilon\text{closure}(\{3, 9\}) \equiv B$

$DTran[C, a] \equiv B$

$mov(C, a) = \{3, 9\}$

$DTran[B, b]$

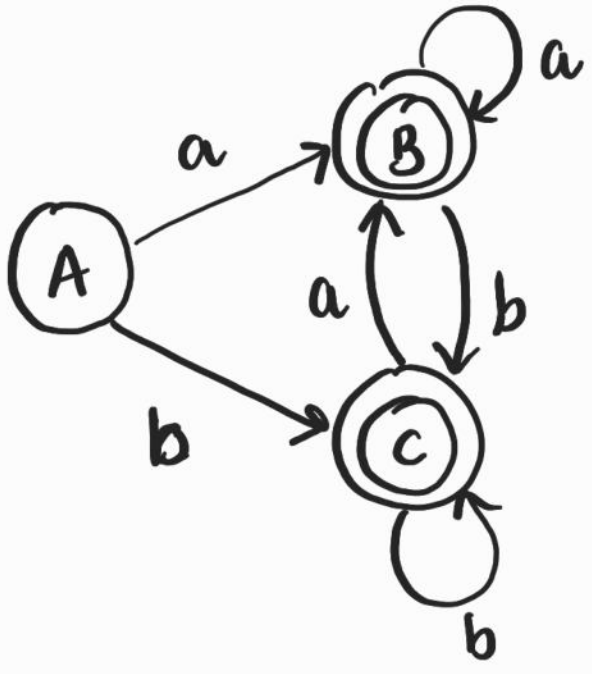
$mov(B, b) = \{5, 11\}$

$\varepsilon\text{closure}(\{5, 11\}) \equiv C$

$DTran[C, b] \equiv C$

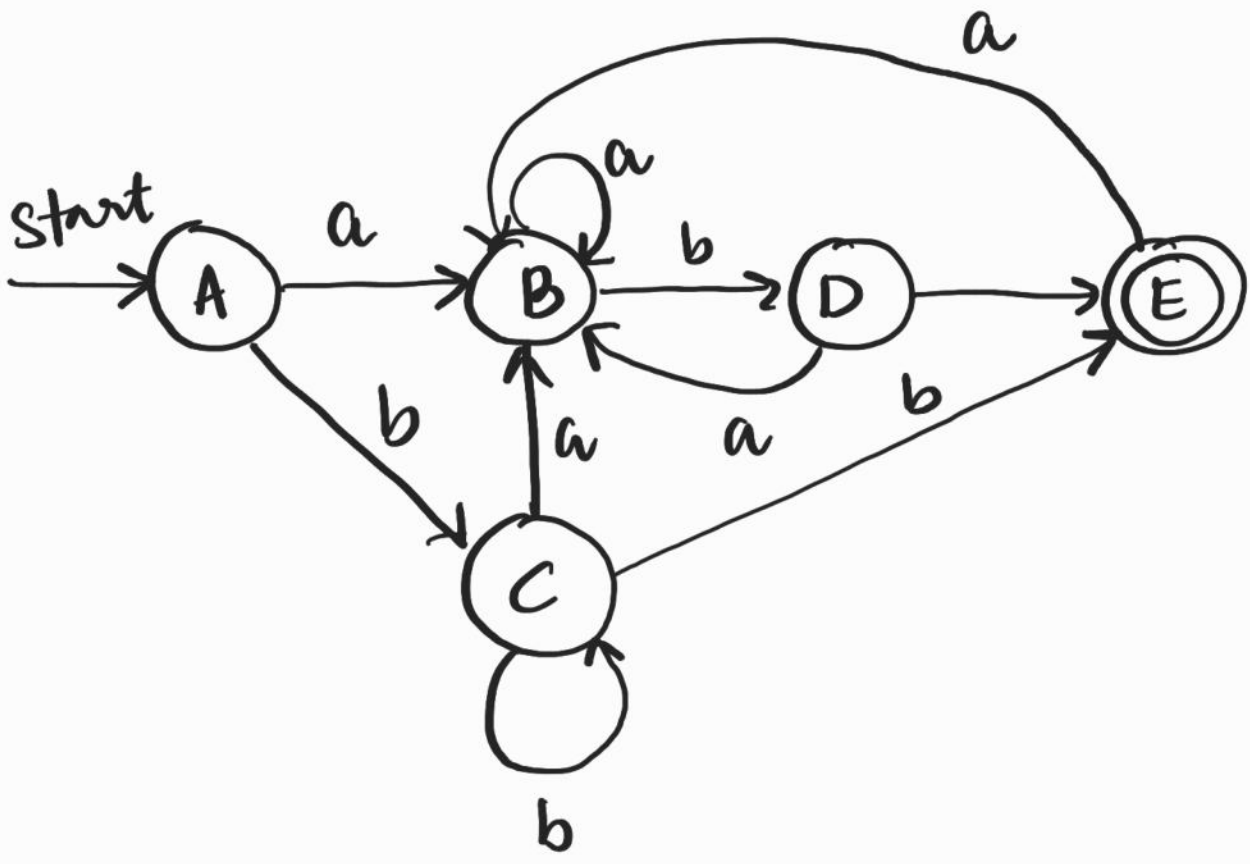
$mov(C, b) = \{5, 11\}$

DFA State	a	b
→ A	B	C
* B	B	C
* C	B	C



Minimization

Due to transition from NFA to DFA by means of subset construction algorithm there maybe addition of certain unnecessary transitions that add to the complexity of the machine. So we optimize.



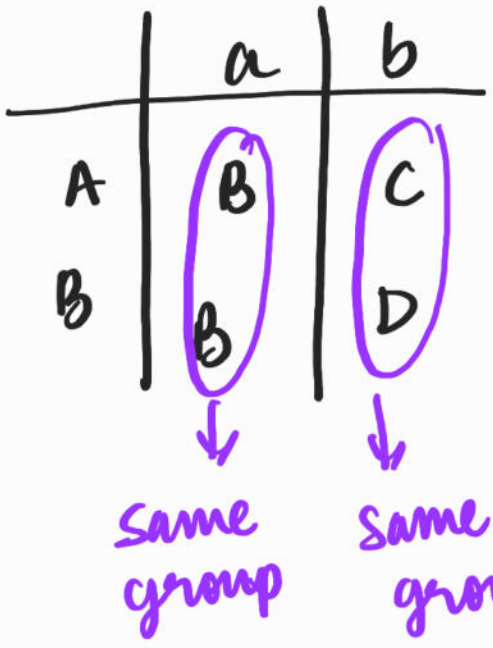
State \ ip symbol	a	b
→ A	B	C
B	B	D
C	B	C
D	B	E
* E	B	C

Final State - E

Non Final States - {A, B, C, D}

$\pi = \{A, B, C, D\}, \{E\}$

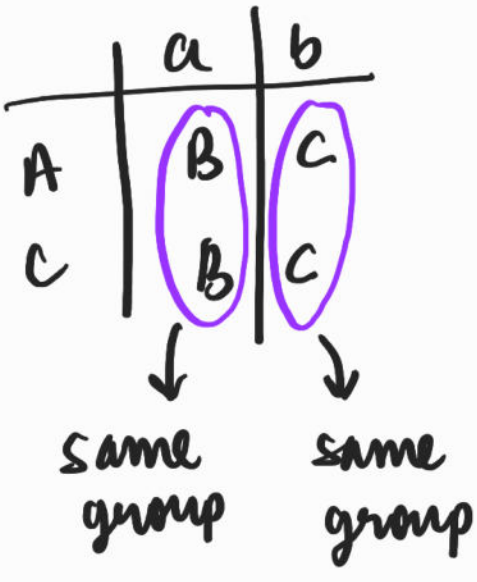
Compare (A, B)



* check if the highlighted are present in same group

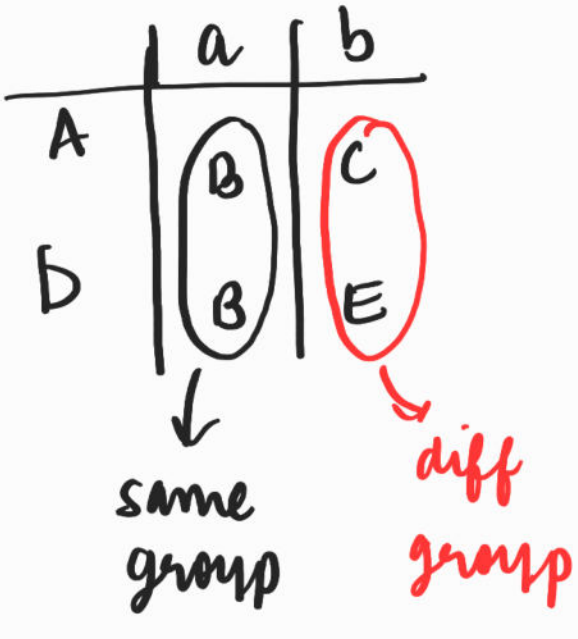
(A, B) can be present in same set.

Compare (A, C)



A & C can be in the same set.

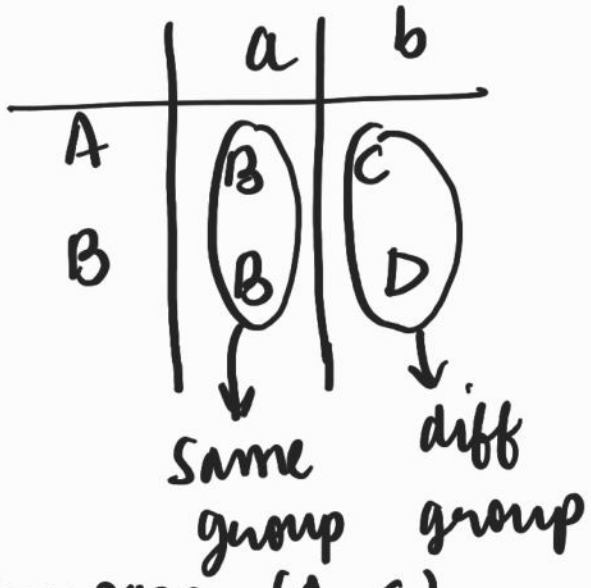
Compare (A, D)



* A & D cannot be in the same group.

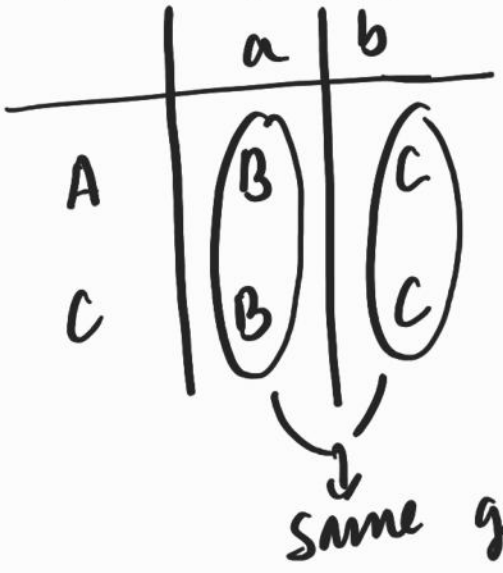
$\pi_1 = \{A, B, C\}, \{D\}, \{E\}$

Compare (A, B)



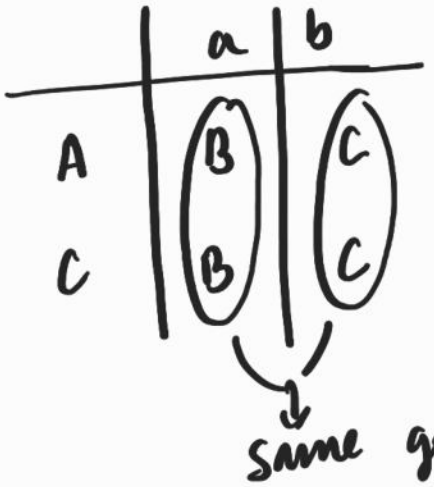
A & B cannot be in same group

Compare (A, C)



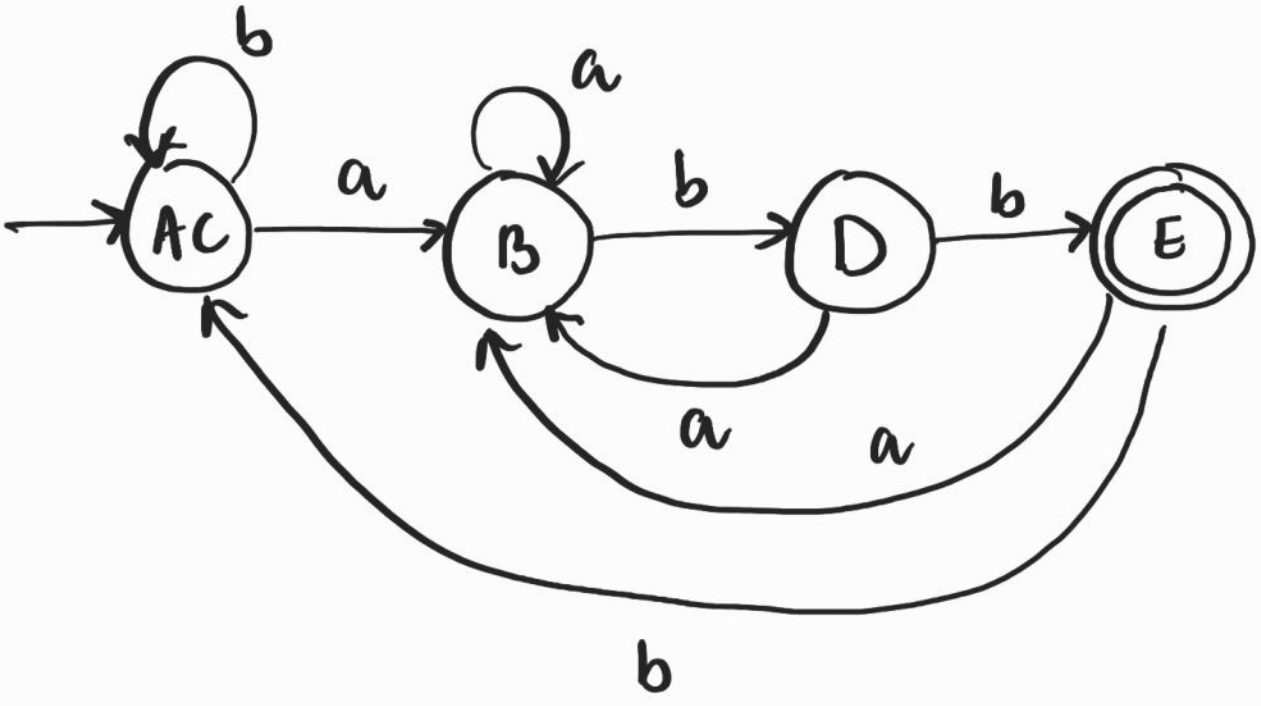
$\pi_2 = \{A, C\}, \{B\}, \{D\}, \{E\}$

Compare (A, C)

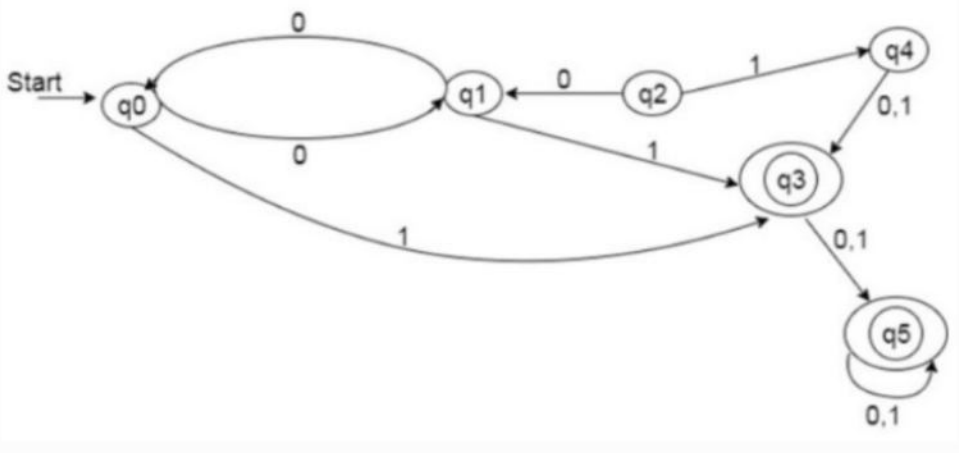


$\pi_2 = \{A, C\}, \{B\}, \{D\}, \{E\}$

state \ ip symbol	a	b
→ {A, C}	B	{A, C}
B	B	D
D	B	E
* E	B	{A, C}



Q. Minimise the DFA



State / IP symbol	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
q_2	q_1	q_4
* q_3	q_5	q_5
q_4	q_3	q_3
* q_5	q_5	q_5

→ not to include .Y?

(cannot reach from q_0)

$\pi = \{q_0, q_1, q_2, q_4\}, \{q_3, q_5\}$

compare (q_0, q_1)

	0	1
q_0	q_1 ✓	q_3 ✓
q_1	q_0	q_3

same set

compare (q_0, q_2)

	0	1
q_0	q_1 ✓	q_3
q_2	q_1	q_4 X

not same set

Compare (q_0, q_4)

	0	1
q_0	q_1	q_3
q_4	q_3	q_3
	X	✓

not same set

Compare (q_3, q_5)

	0	1
q_3	q_5	q_5
q_5	q_5	q_5
	✓	✓

same set

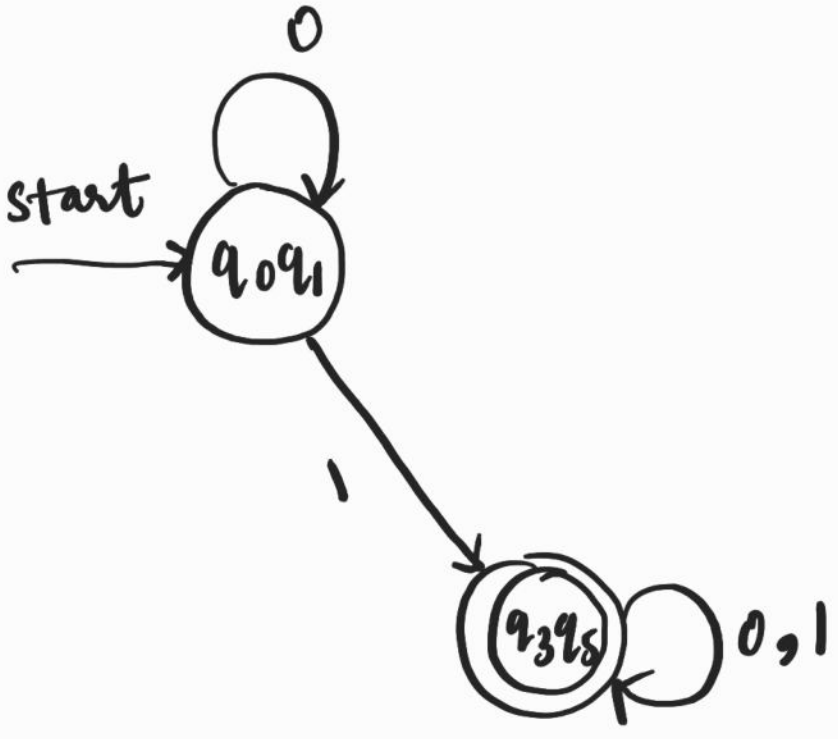
$\pi_1 = \{q_0, q_1\}, \{q_2\}, \{q_4\}, \{q_3, q_5\}$

Compare (q_0, q_1)

	0	1
q_0	q_1	q_3
q_1	q_0	q_3
	✓	✓

same set

State / IP symbol	0	1
$\rightarrow \{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_3, q_5\}$
$* \{q_3, q_5\}$	$\{q_3, q_5\}$	$\{q_3, q_5\}$



— x — x — x — Correct ans — x — x —

	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
$* q_3$	q_5	q_5
$* q_5$	q_5	q_5

$\pi = \{q_0, q_1\}, \{q_3, q_5\}$

compare (q_0, q_1)

	0	1
q_0	q_1	q_3
q_1	q_0	q_3

✓ ✓

same set

Compare (q_3, q_5)

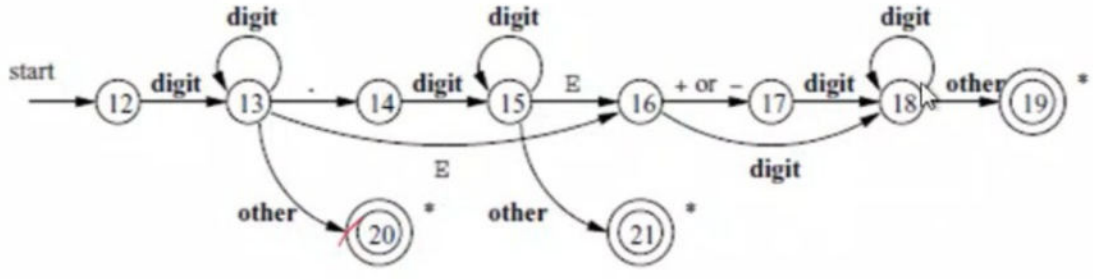
	0	1
q_3	q_5	q_5
q_5	q_5	q_5

✓ ✓

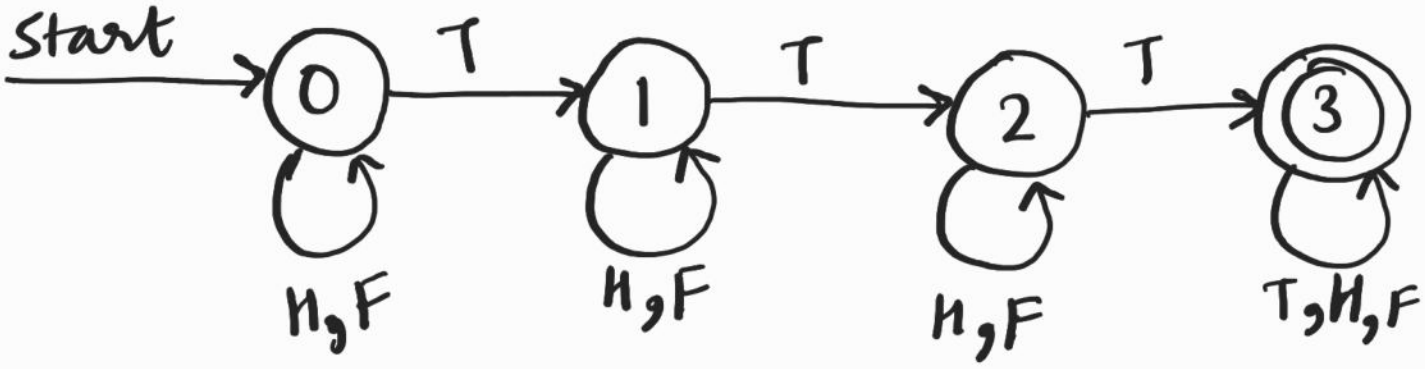
same set

$\pi = \{q_0, q_1\}, \{q_3, q_5\}$

* Unsigned numbers (integers or floating point) are strings such as 5280, 0.01234, 6.336E4, 1.89E-4



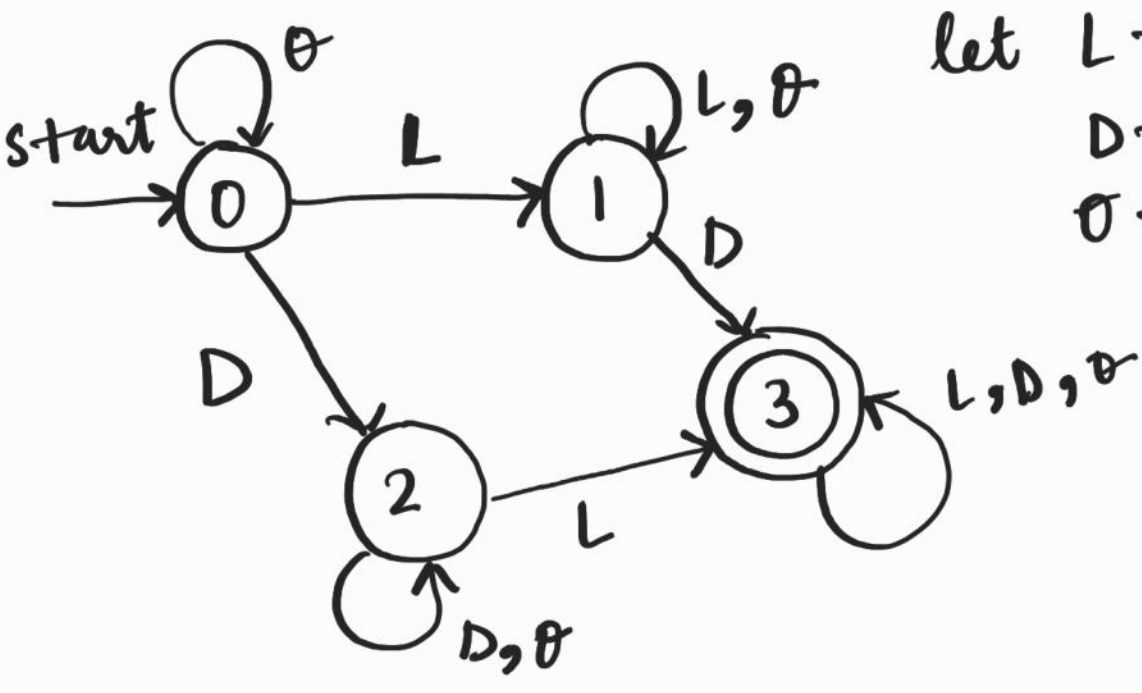
Q Design a DFA that accepts only the sequence of rupees consists of at least 3 10(T) rupee notes. The rupee notes are 10(T), 100(H), 500(F) $\Sigma = (T, H, F)$



Q Consider the given HTML script, which displays the content presented inside the header <h1> tag on the webpage. Assume that the h1 tag should contain atleast one digit or one alphabet.

```
<html>
<h1>Welcome to CSE2002 CAT Exam </h1>
</html>
```

Design a finite-state automata to validate the HTML script.

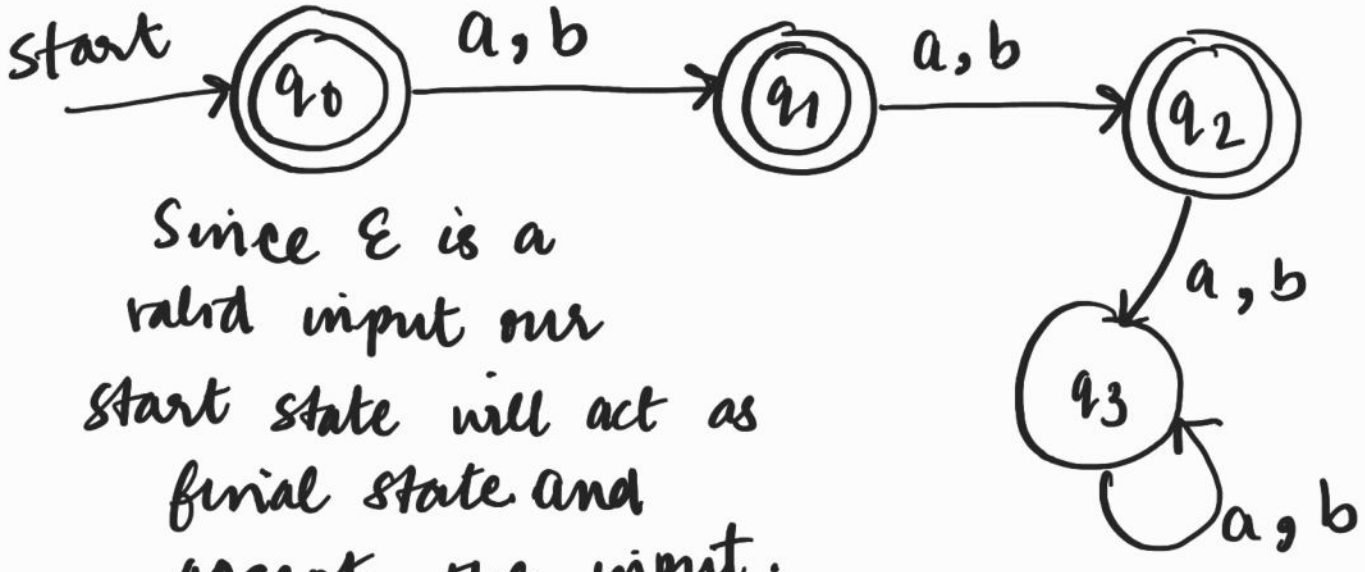


Q $\Sigma = \{a, b\}$

Design a DFA for length of string atmost 2.

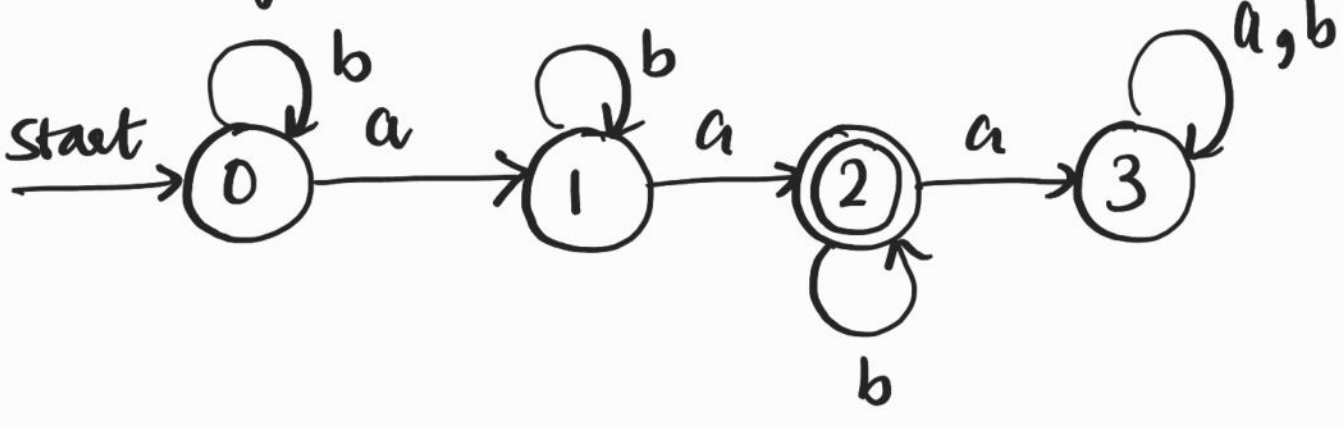
$\varepsilon, a, b, aa, ab, ba, bb$

* DFA cannot have ε transitions.

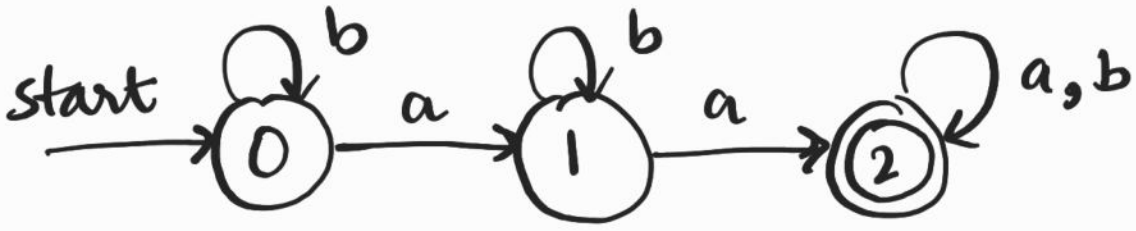


Since ε is a valid input our start state will act as finial state and accept the input.

Q exactly 2 'a's

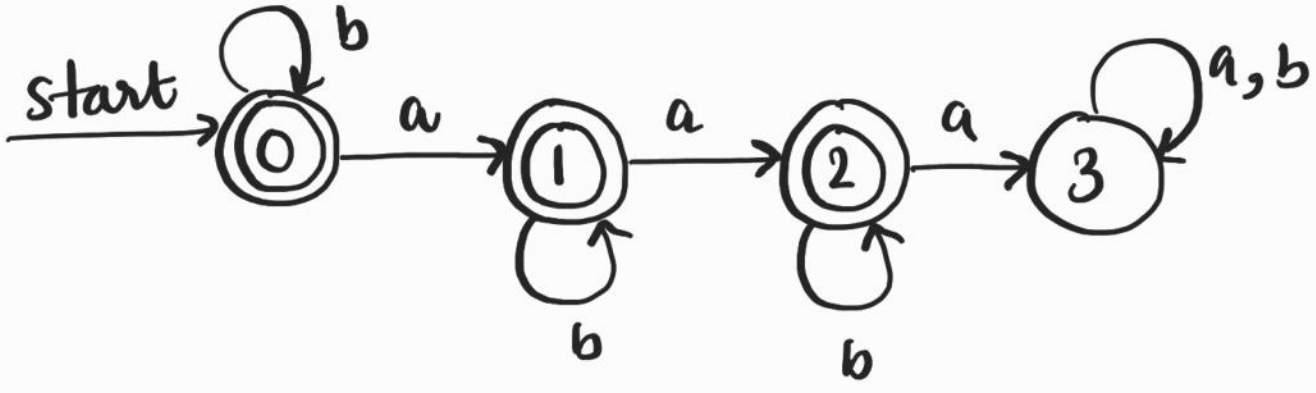


Q atleast 2 'a's



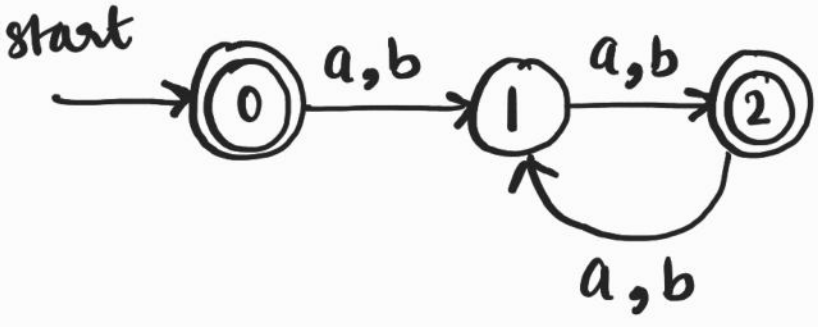
Q atmost 2 'a's

Σ, a, aa, ba, baa

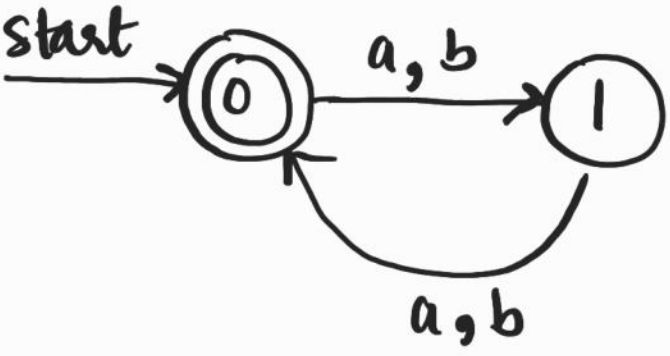


Q $|w| \bmod 2 = 0$ (even length strings)

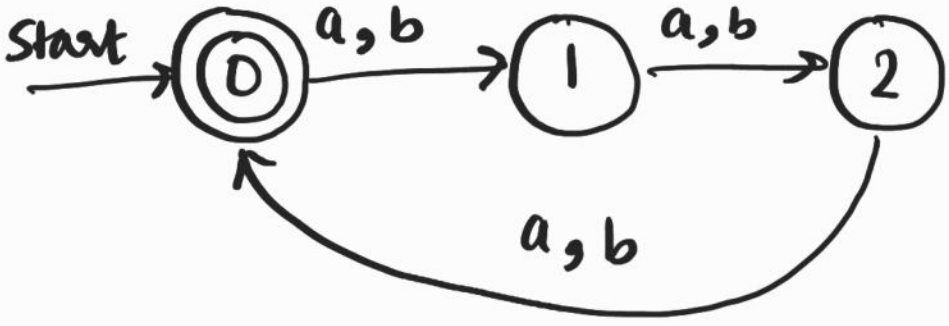
0, 2, 4, 6....



$\equiv$

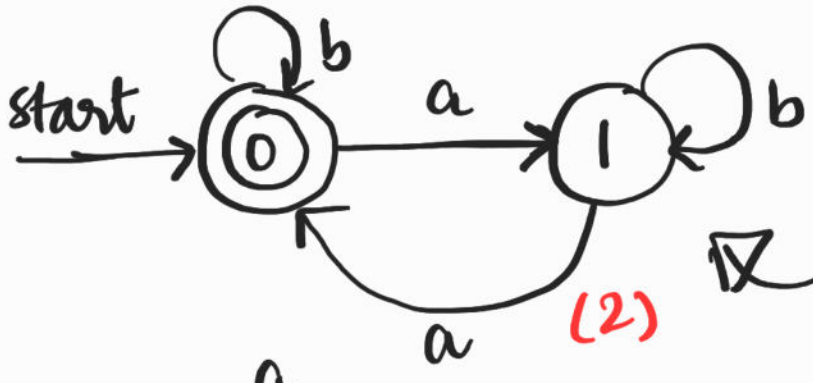


Q $|w| \bmod 3 = 0$

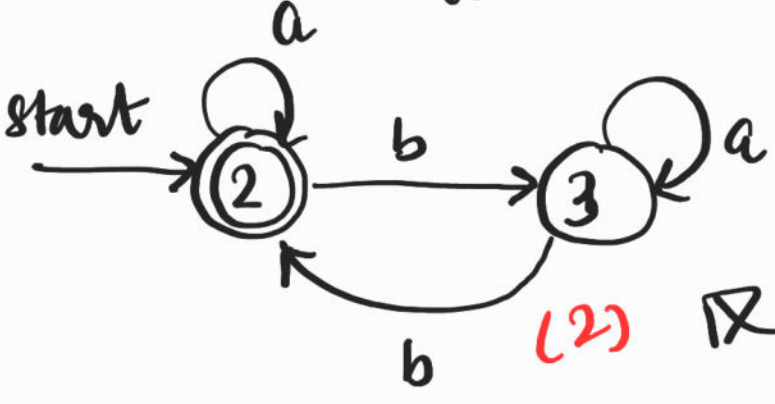


Q 'a's divisible by 2 and 'b's divisible by 2. $n_a(w) \bmod 2 = 0$

$n_b(w) \bmod 2 = 0$



$n_a(w) \bmod 2 = 0$



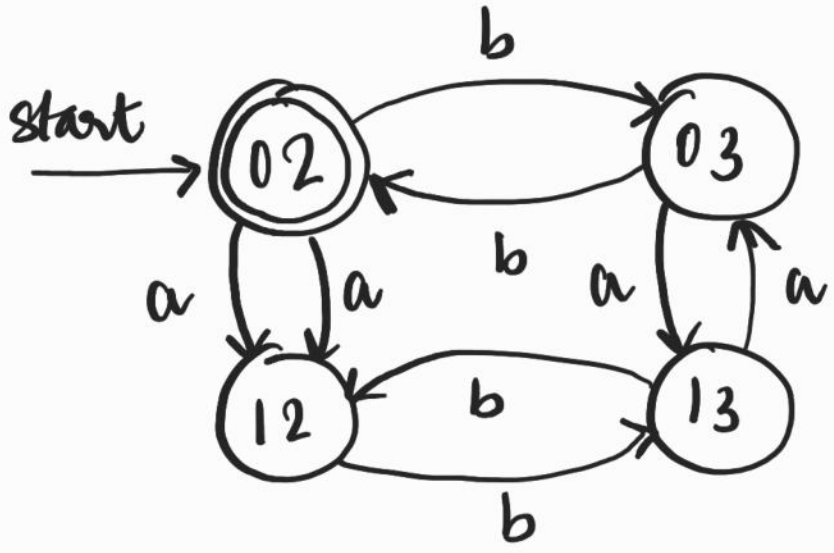
$n_b(w) \bmod 2 = 0$

Take cross product of the 2 automatas

Final DFA will have 4 states

If DFA_1 has m states, DFA_2 has n states

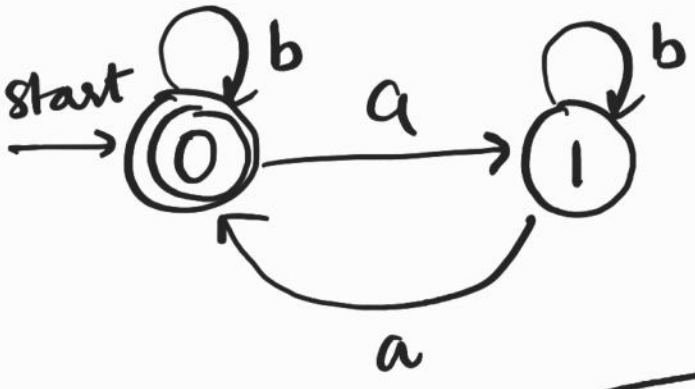
Final DFA will have m x n states.



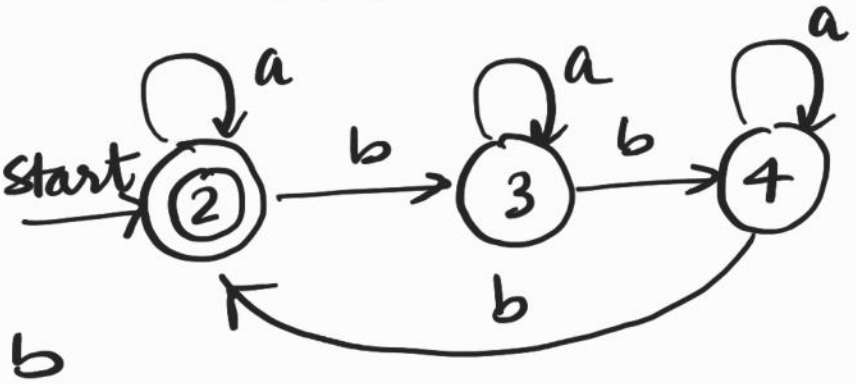
$\underline{\underline{\text{Q}}}$ $n_a(w) \bmod 2 = 0$

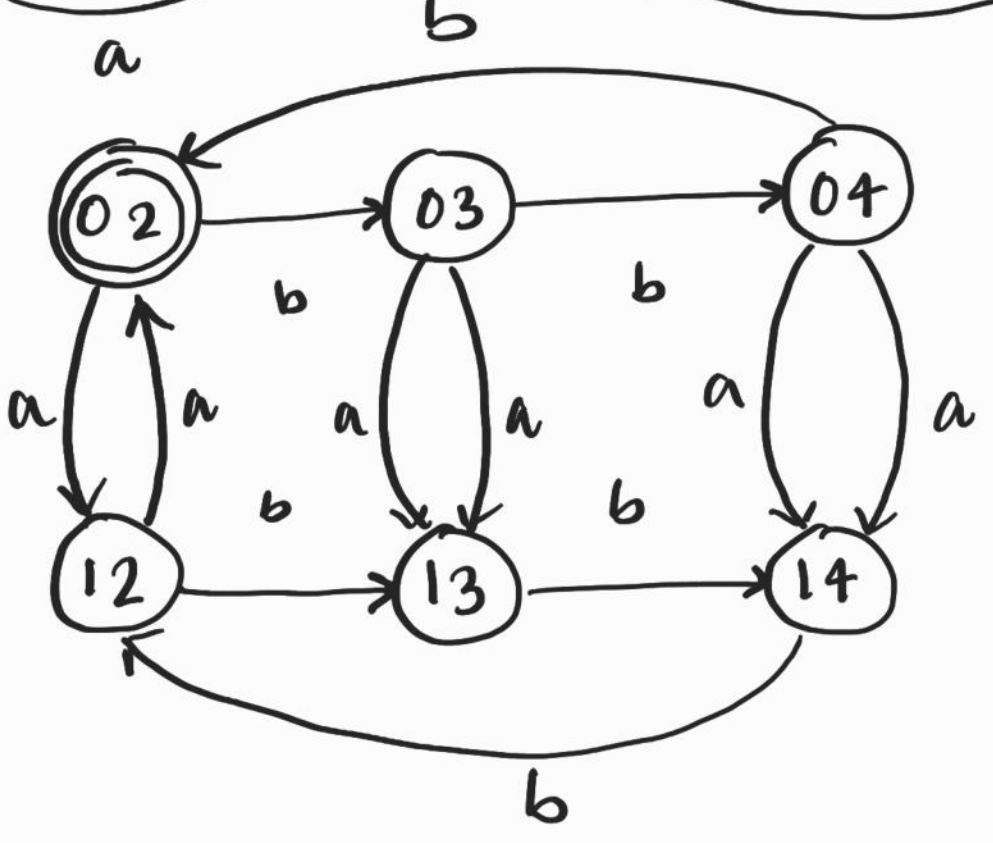
$n_b(w) \bmod 3 = 0$

$n_a(w) \bmod 2 = 0$



$n_b(w) \bmod 3 = 0$

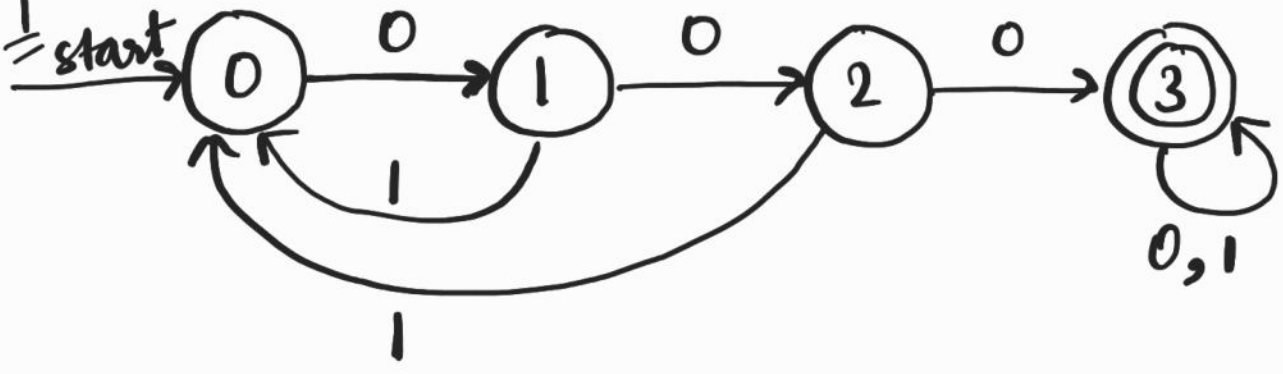




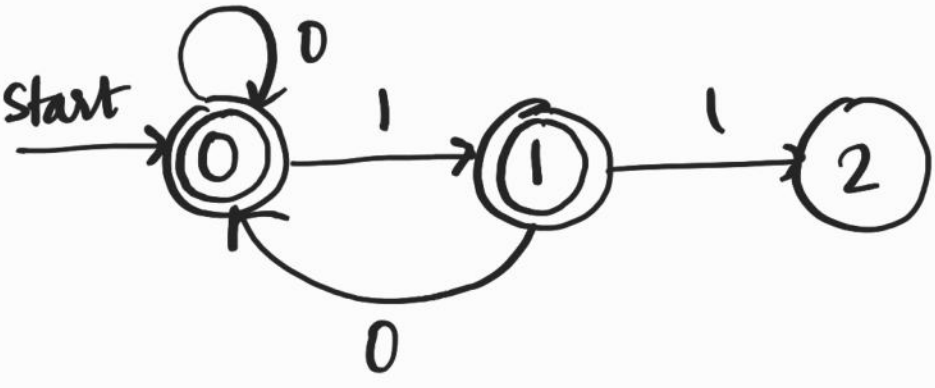
Q $\Sigma = \{0, 1\}$

1. should have 3 consecutive '0's
2. does not contain consecutive '1's
3. starting with '0' and ending with '1'.

1

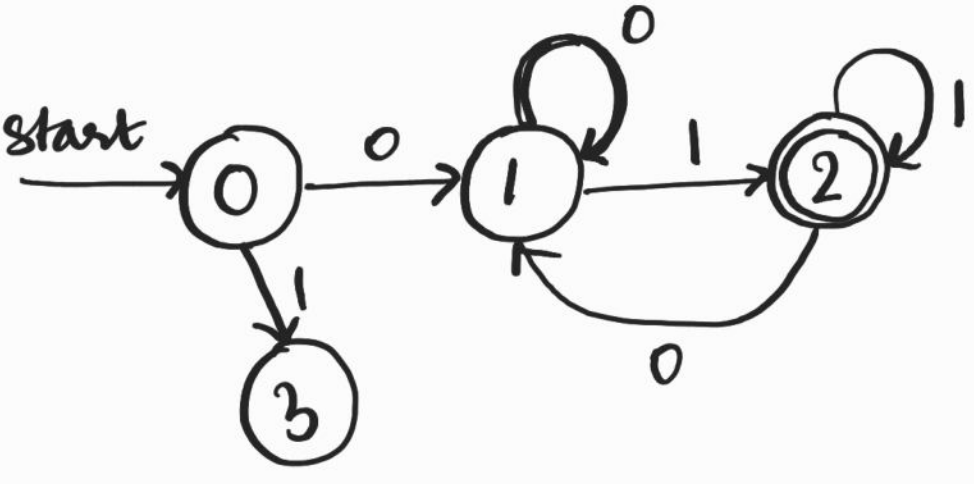


2



3 $0(0|1)^*1$

0 1 1 1 0 1



Arden's Theorem

DFA to Regular expression

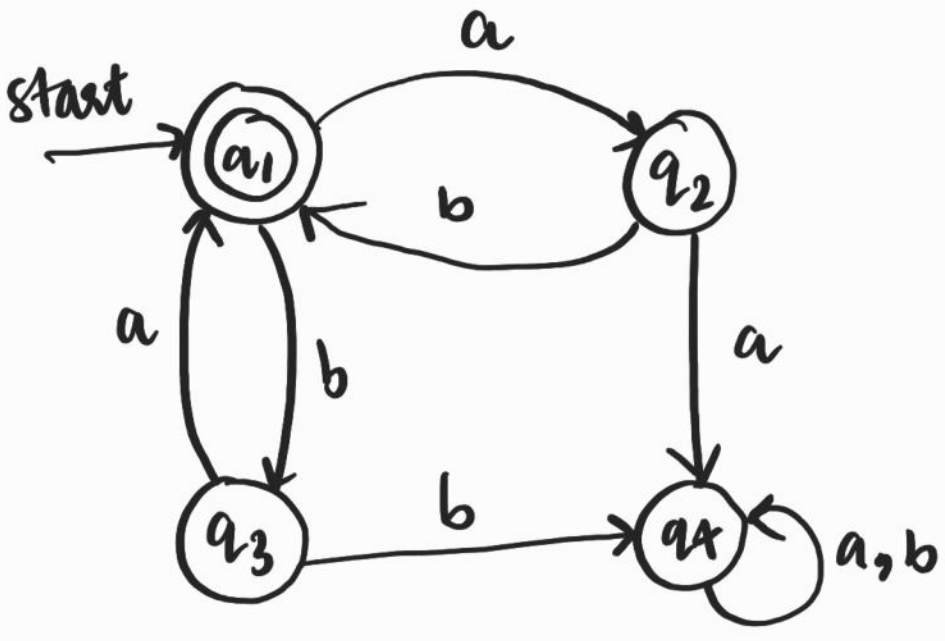
let P and Q be 2 regular expressions.

If P does not contain null string,

then $R = Q + RP$ has a unique solution

$$R = QP^*$$

(Make for final states)



$$q_1 = q_2 b + q_3 a + \in \quad -(1)$$

incoming transition

$$q_2 = q_1 a \quad -(2) \; ; \; q_3 = q_1 b \quad -(3)$$

$$q_4 = q_2 a + q_3 b + q_4 a + q_4 b \quad -(4)$$

① $q_1 = q_1 ab + q_1 ba + \in$

$q_1 = q_1(ab+ba) + \in$ $\quad P = q_1 \quad Q = \in$

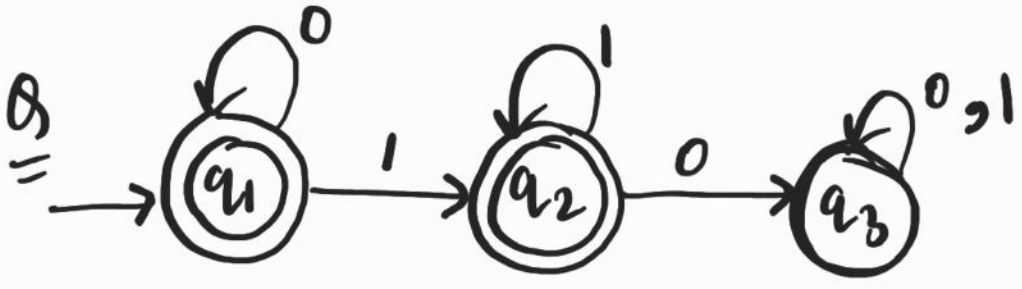
$R = ab + ba$

$q_1 = \in (ab+ba)^*$

union

Regular Expression :- $(ab + ba)^*$

$\equiv (ab|ba)^*$



$q_1 = \Sigma + q_1(0)$ —①

$q_1 = q_1(0) + \Sigma$

R R(P) Q

$q_2 = q_1 1 + q_2 1$ —②

$q_3 = q_2 0 + q_3 0 + q_3 1$ —③

$q_1 = 0^*$

$q_2 = q_1 1 + q_2 1$

$q_2 = 0^* 1 + q_2 1$

R Q R P

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

Take $q_1 + q_2$

$0^* + (0^*1)(1^*)$

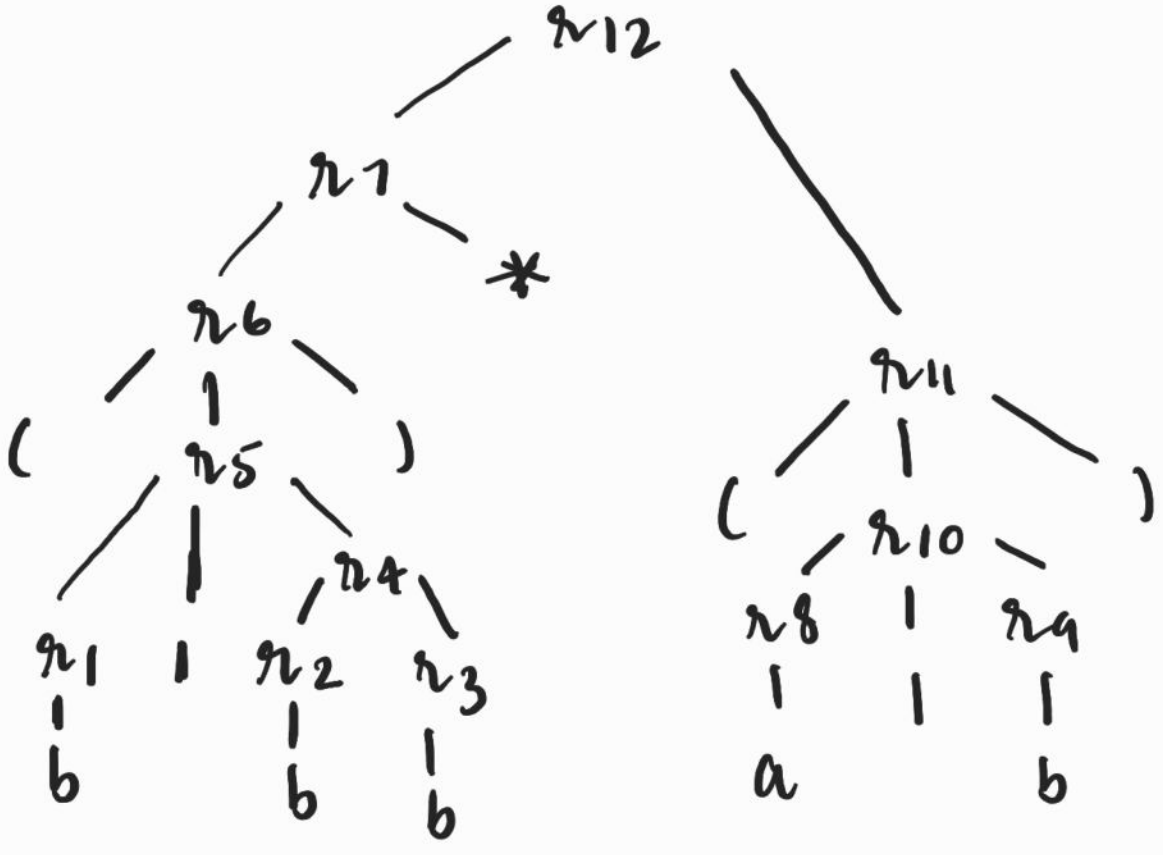
$\equiv 0^* (\in + 11^*)$

$\equiv 0^* (11^*)$

* if for some Dtran $[A, a] = \phi$, then ($move(A, a) = \{\}$) include a trap state.

DFA State	a	b
A	$\phi \equiv D$	
B		
C		
D	D	D

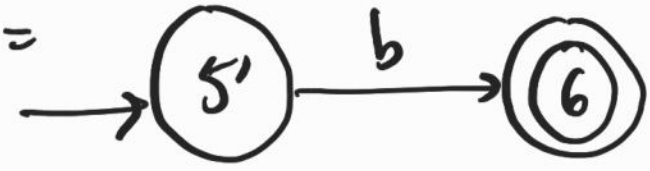
Q $(b|bb)^*(a|b)$ find ϵ-NFA to DFA.



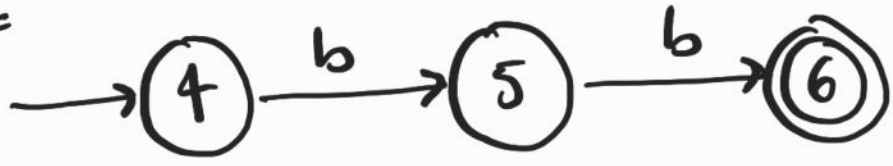
$r_1 =$

$r_2 =$

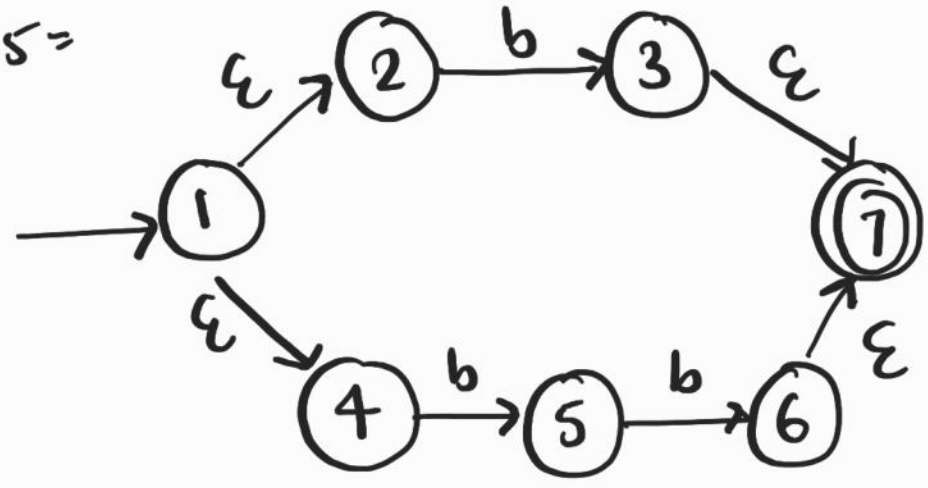
$r_3 =$



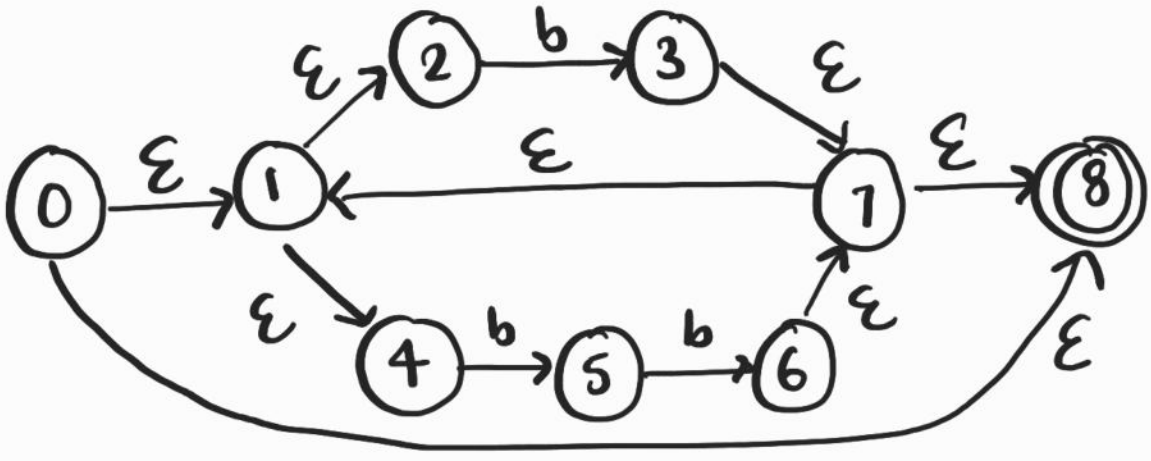
$r_4 =$



$r_5 =$

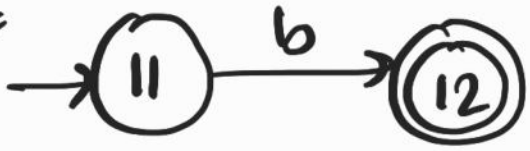


$r_6 = r_5$, $r_7 =$

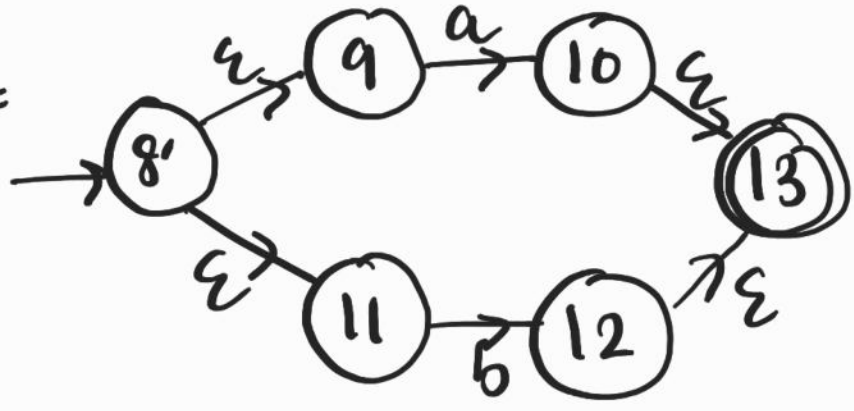


$r_9 =$

$r_{9} =$

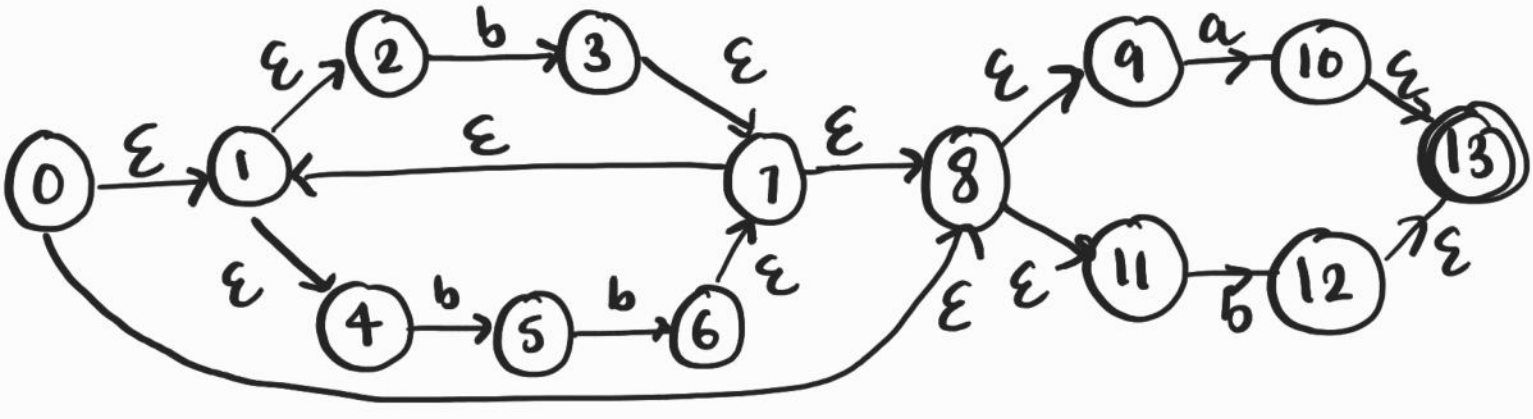


$r_{10} =$



$r_{11} = r_{10}$

$r_{12} =$



ϵ closure($\{0\}$) = $\{0, 1, 2, 4, 8, 9, 11\} \equiv A$

DTran [A, a]

move(A, a) = $\{10\}$

εclosure($\{10\}$) = $\{13, 10\} \equiv B$

DTran [A, b]

mov (A, b) = {3, 5, 12} C

∈({3, 5, 12}) = {3, 7, 1, 2, 4, 8, 9, 11, 13, 12}

DTran [B, a]

mov (B, a) = { } ϕ Trap state → D

DTran [B, b] =

mov (B, b) = { } ϕ Trap state

DTran [C, a] =

mov [C, a] = {10}

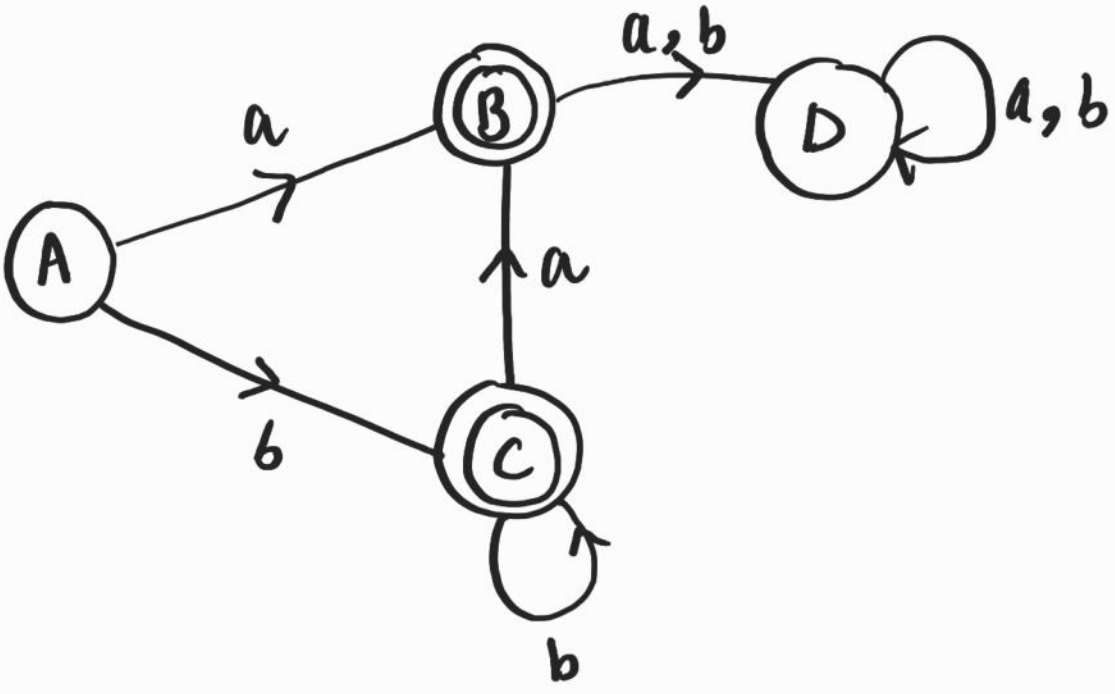
ε closure ({10}) ≡ B

DTran [C, b]

mov [C, b] = {5, 12, 3}

Edonure {5, 12, 3} = C

	a	b
→ A	B	C
* B	D	D
* C	B	C
D	D	D



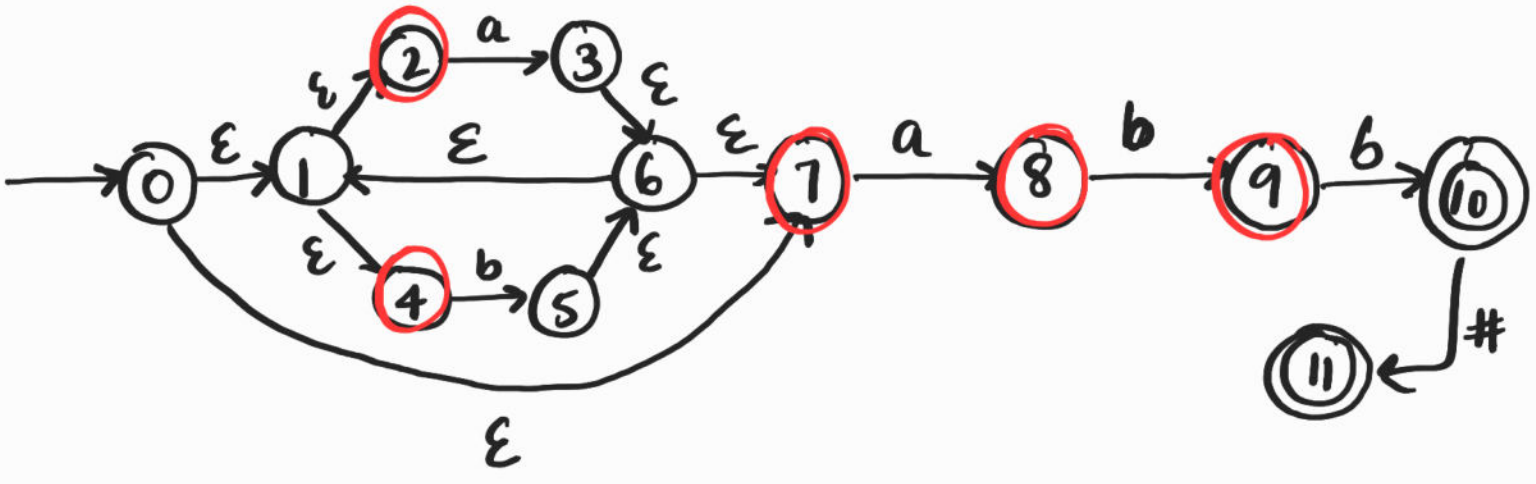
Construction of DFA from Reg Ex

OPTIMISATION

Q $(a|b)^*abb$

1. Identify the important states in the NFA.

The NFA states without Non-ε out transition are called important states.



Final State also imp. We include it

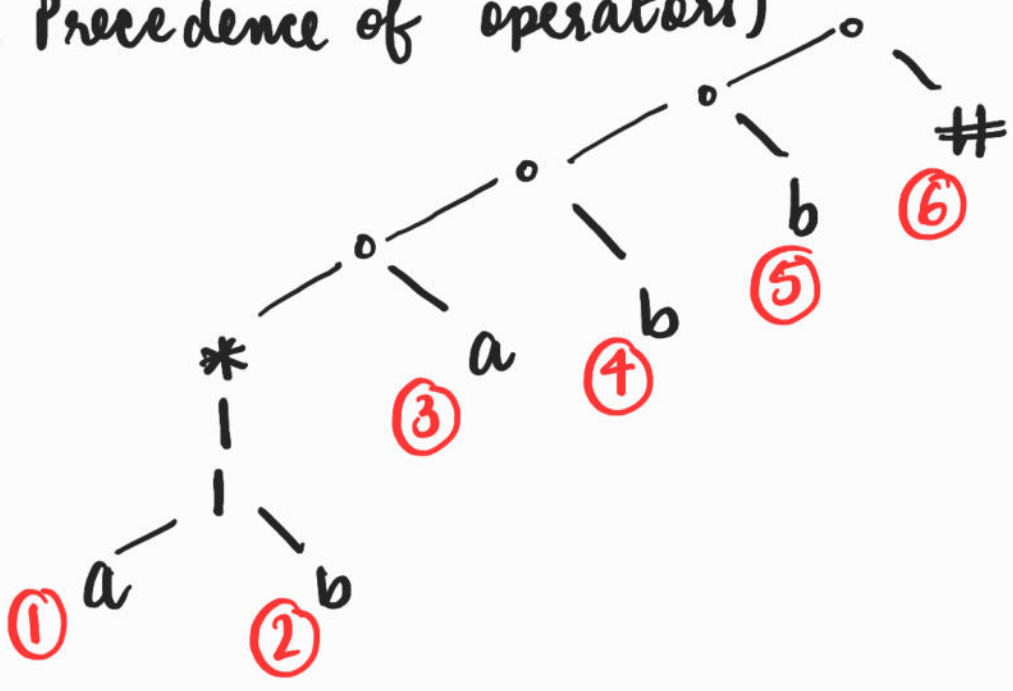
by ① converting Reg Ex to augmented Reg Ex (concatenate a # output from 10 to 11)

→ $(a|b)^*abb\#$ → Augmented Reg Ex

② Construct Parse Tree

• → concat

(Precedence of operators)



③ Number all Non ε leaf nodes.

④ Work on 4 functions :-

Nullable :- Check whether it is empty or not

First posn

Last posn

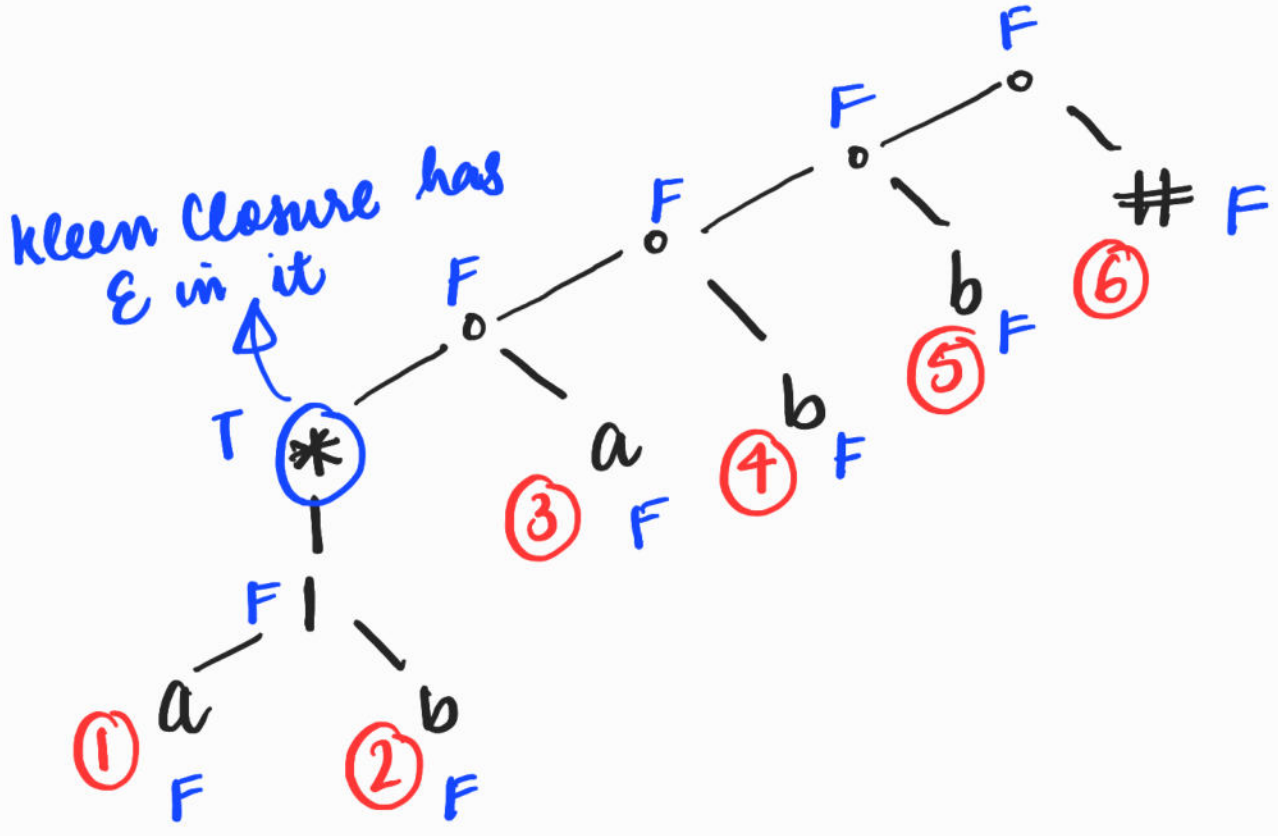
follow posn

Node n	nullable(n)	firstpos(n)	lastpos(n)
A leaf labelled ϵ	true	ϕ	ϕ
A leaf with position i	false	$\{i\}$	$\{i\}$
or-node ie $n = c_1 \mid c_2$	nullable(c_1) or nullable(c_2)	firstpos(c_1) $\cup$ firstpos(c_2)	lastpos(c_1) $\cup$ lastpos(c_2)

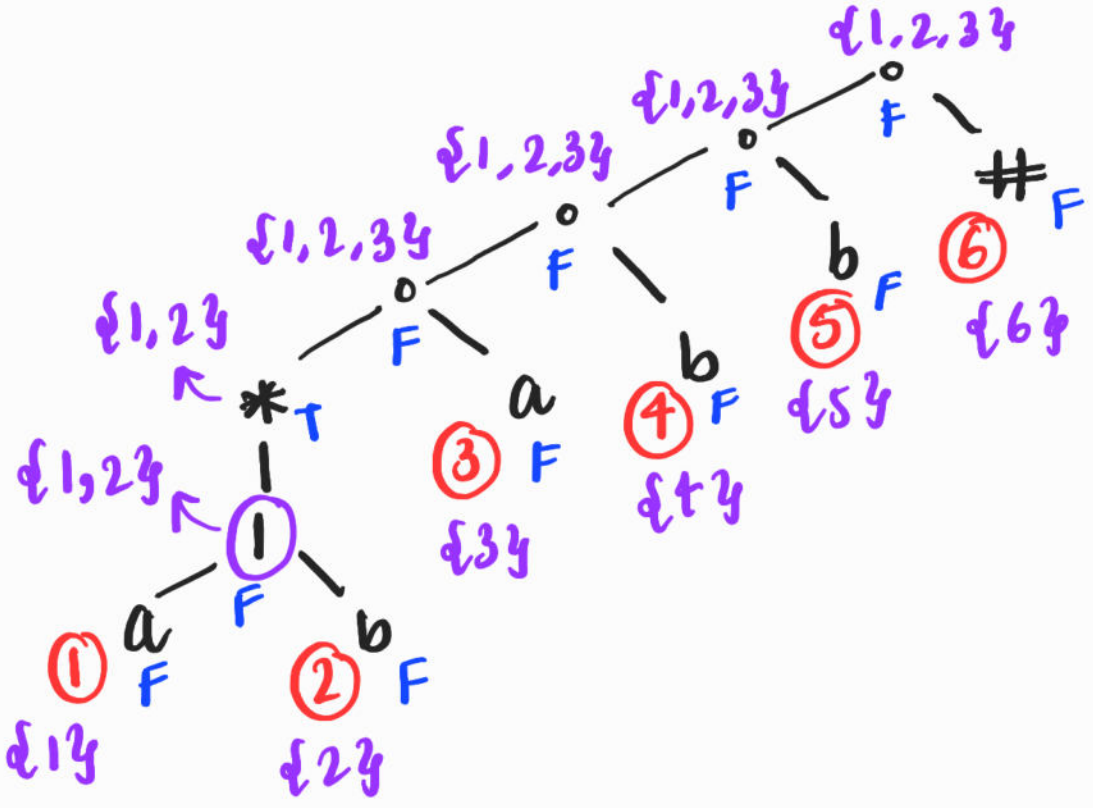
Cat-node ie $n = c_1 c_2$	nullable(c_1) and nullable(c_2)	IMP
		if (nullable(c_1))
		firstpos(c_1) $\cup$ firstpos(c_2)
		else
		firstpos(c_1)
			if (nullable(c_2))
			lastpos(c_1) $\cup$ lastpos(c_2)
			else
			lastpos(c_2)

Star-node ie $n = c_1^*$	true	firstpos(c_1)	lastpos(c_1)

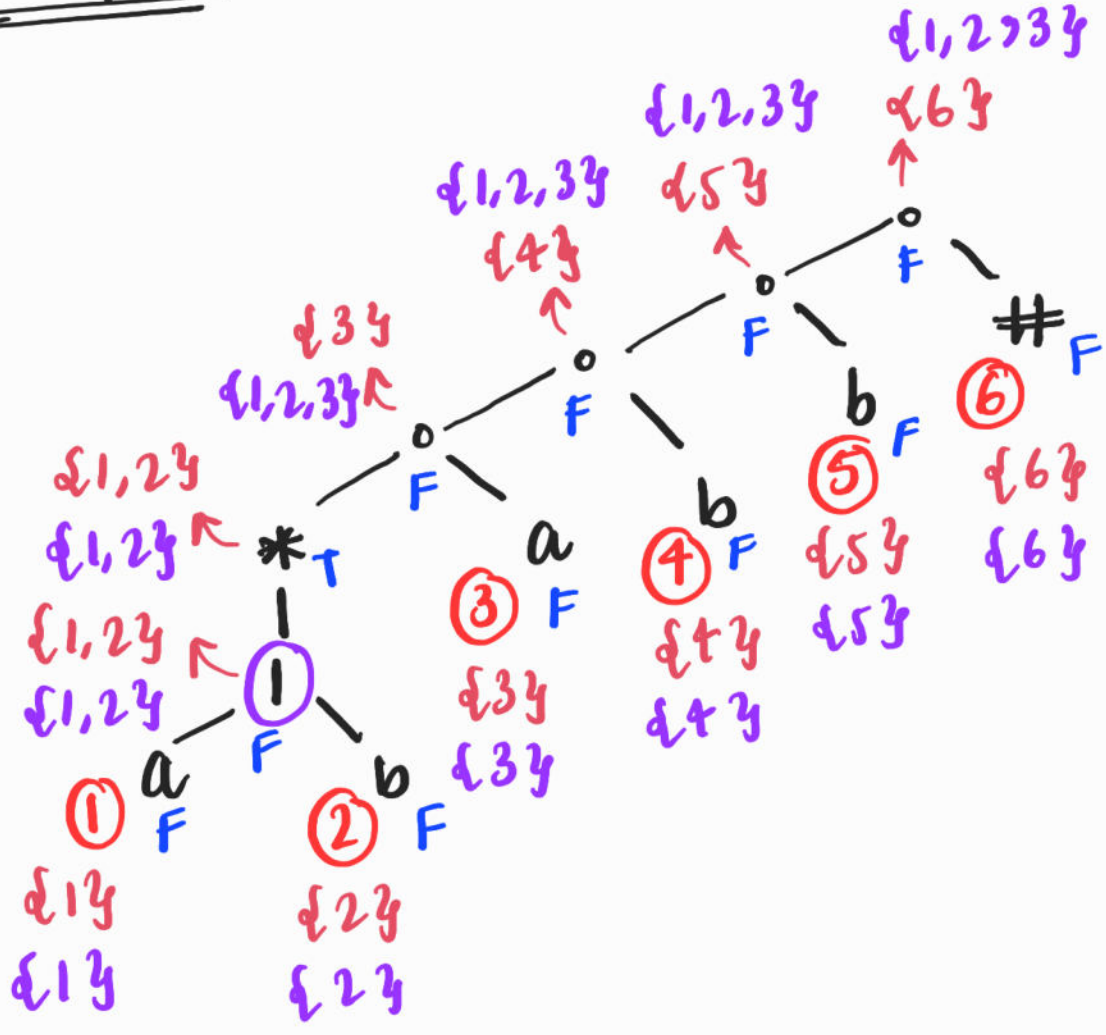
Nullable



2 Firstpos



3 Lastpos



4 Followpos

only for the numbered leaf nodes *

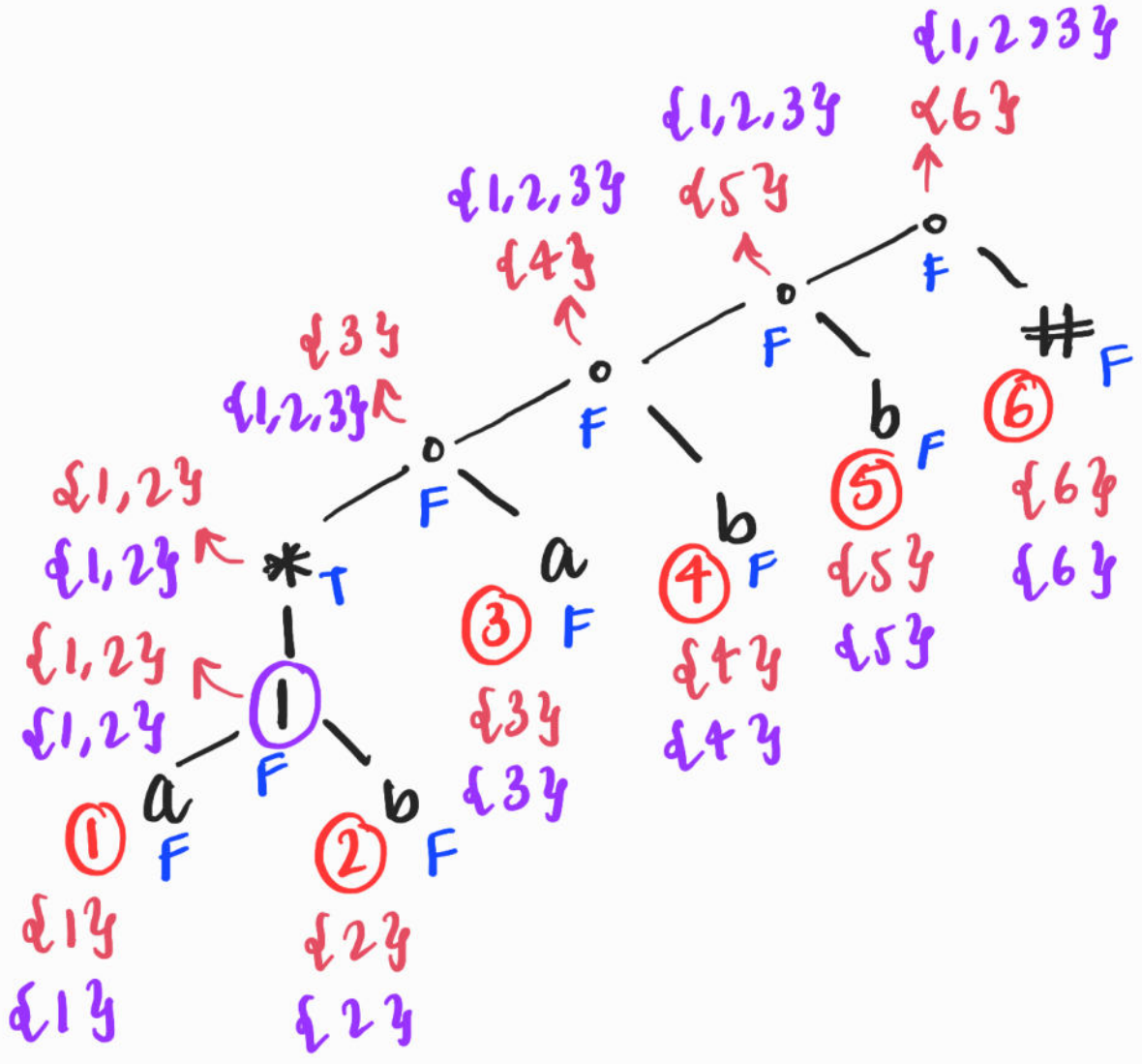
followpos(i) — Check all cat nodes and star nodes

cat node → if i is present in lastpos(C_1)

⇒ include all firstpos(C_2) in followpos(i)

star node → if i is present in laspos(*)

⇒ include firstpos(*) in followpos(i)



followpos(1) :- {1, 2, 3}

followpos(2) :- {1, 2, 3}

followpos(3) :- {4}

followpos(4) :- {5}

followpos(5) :- {6}

followpos(6) :- ϕ

(5) Construct DFA table.

* Start state will be firstpos(root node)

{1, 2, 3} → A

Dtran [A, a] & Dtran [A, b]

Dtran({1, 2, 3}) = followpos(1) ∪ followpos(3)

include the posns which have input symbol 'a'.

Dtran [A, a] = {1, 2, 3, 4} ≡ B

Dtran [A, b] = followpos(2) = {1, 2, 3} ≡ A

Dtran [B, a] = followpos(1) ∪ followpos(3)

= {1, 2, 3, 4} ≡ B

Dtran [B, b] = followpos (2) ∪ followpos (4)

= {1, 2, 3, 5} ≡ C

Dtran [C, a] = followpos (1) ∪ followpos (3)

= {1, 2, 3, 4} ≡ B

Dtran [C, b] = followpos (1) ∪ followpos (5)

= {1, 2, 3, 6} ≡ D

Dtran [D, a] = followpos (1) ∪ followpos (3)

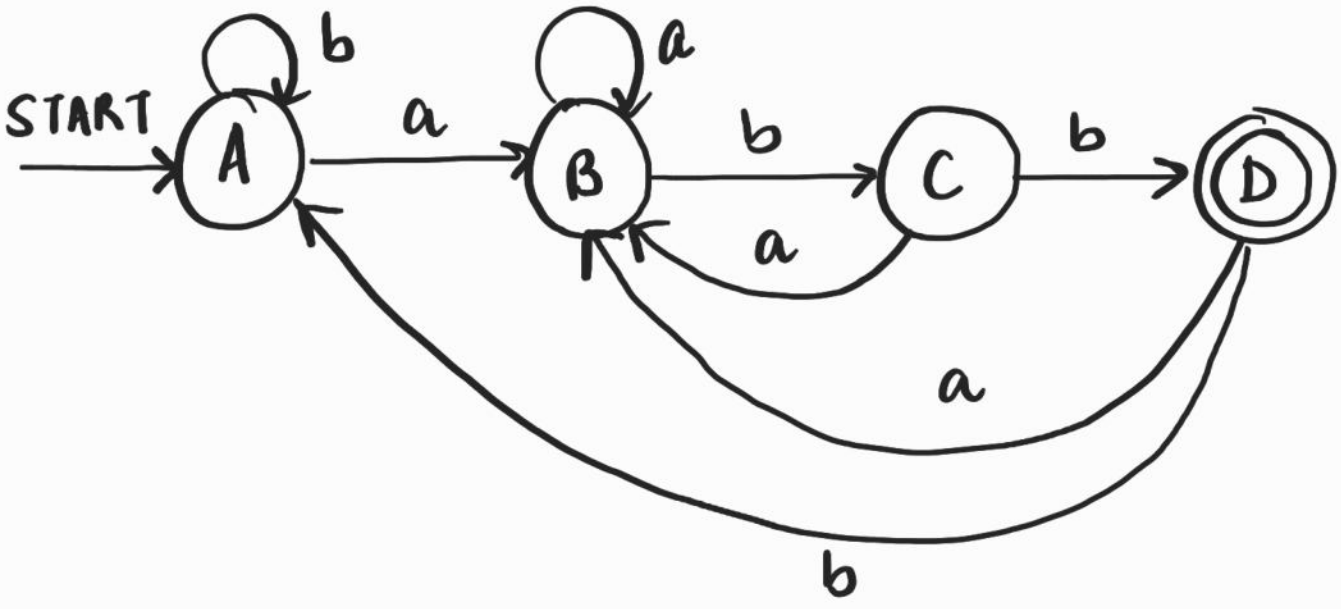
= {1, 2, 3, 4} ≡ B

Dtran [D, b] = followpos (2)

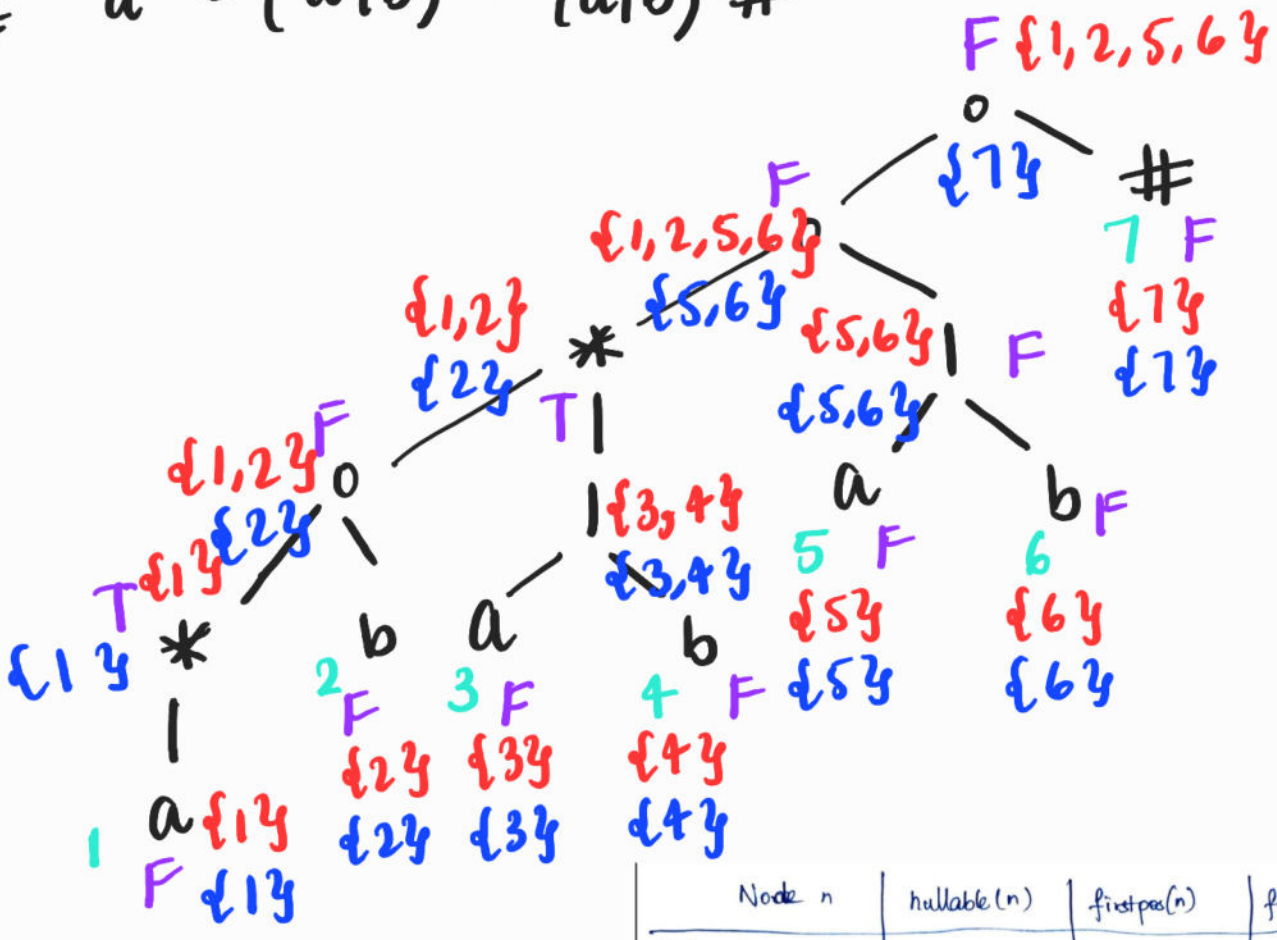
= {1, 2, 3} ≡ A

Transition Table

	a	b
→ A	B	A
B	B	C
C	B	D
* D	B	A



Q $\equiv$ $a^* b (a|b)^* (a|b) \#$



Node n	nullable(n)	firstpos(n)	followpos(n)
A leaf labelled ϵ	true	$\emptyset$	$\emptyset$
A leaf with position i	false	$\{i\}$	$\{i\}$
Or-node ie $n = c_1 \mid c_2$	nullable(c_1) or nullable(c_2)	firstpos(c_1) $\cup$ firstpos(c_2)	
Cat-node ie $n = c_1 c_2$	nullable(c_1) and nullable(c_2)	if (nullable(c_1)) firstpos(c_1) $\cup$ firstpos(c_2) else firstpos(c_1)	if (nullable(c_2)) lastpos(c_1) $\cup$ lastpos(c_2) else lastpos(c_2)
Star-node ie $n = c_1^*$	true	firstpos(c_1)	lastpos(c_1)

followpos (1) = {1, 2}

followpos (2) = {1, 2,

followpos (3) =

followpos (4) =

followpos (5) =

followpos (6) =

followpos (7) =

cat node → if i is present in lastpos(C_1)

⇒ include all firstpos(C_2) in followpos(i)

star node → if i is present in laspos(*)

⇒ include firstpos(*) in followpos(i)

Pumping Lemma

* to prove that a language is not regular we cannot prove that it regular.

For any reg lang L, there exists a constant n, such that for all $w \in L$ with $|w| \geq n$. May be divided into 3 parts $w = xyz$, such that the foll condn must be true

1. $|xy| \leq n$
2. $|y| > 0$
3. $xy^i z \in L, \forall i \geq 0$

Q Prove that $0^n 1^n$ is not regular.

* Proof by contradiction.

let $n = 3 \Rightarrow 000111$

$x = 0 \quad y = 0 \quad z = 0111$

$|xy| \leq n$

$|y| > 0$

$xy^i z \Rightarrow i = 0 \quad 0111 \notin L$

$i=1$ $000111 \in L$

$i=2$ $0000111 \notin L$

Q $L = \{ a^n \mid n$ is a perfect square $\}$

$L = \{ a^1, a^4, a^9, a^{16}, \ldots \}$

$w = aaaa$, $n = 4$ $\Rightarrow |w| \geq n$

$x = a$ $y = a$ $z = aa$

$\rightarrow |xy| \leq n$

$\rightarrow |y| > 0$

$\rightarrow xy^i z$ $\Rightarrow i=0$ $aaa \notin L$

$\Rightarrow i=1$ $aaaa \in L$

$\Rightarrow i=2$ $aaaaa \notin L$

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

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

$L = \{ a^2, a^3, a^5, a^7, \ldots \}$

let $w = aaaaa$ $\Rightarrow |w| \geq n$

$x = a$ $y = a$ $z = aaa$

$|xy| \leq n$, $|y| > 0$

$xy^i z$ $\quad i=0$ $\quad a\,aaa \notin L$

$\quad i=1$ $\quad aa\,aaa \in L$

$\quad i=2$ $\quad aaa\,aaa \notin L$

$\quad i=3$ $\quad a\,aaa\,aaa \in L$

hence not a reg language.

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

$L = \{ 0^2, 0^4, 0^6, 0^8 \ldots \}$

$L = 000000$

$x = 0$ $\quad y = 0$ $\quad z = 000$

$|xy| \leq n$, $|y| > 0$

$xy^i z \Rightarrow$ $\quad i=0$ $\quad 0\,0000 \notin L$

$\quad i=1$ $\quad 0\,00000 \in L$

$\quad i=2$ $\quad 0\,000000 \notin L$

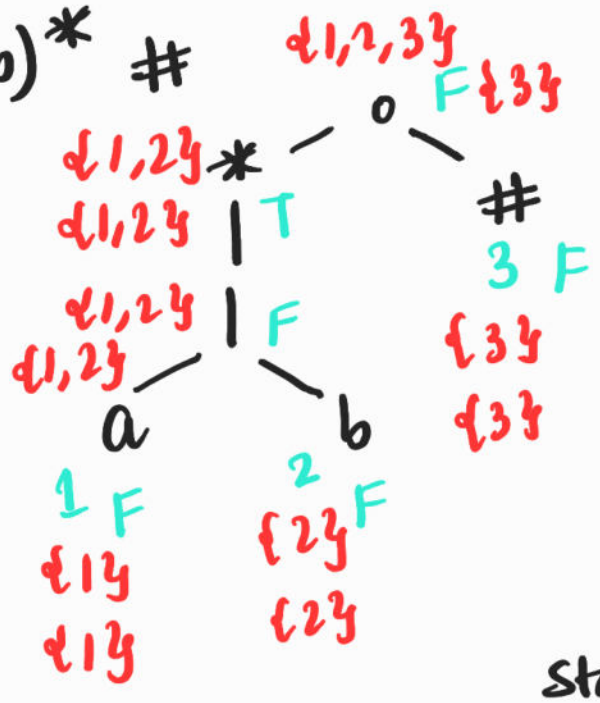
$\quad i=3$ $\quad 0\,0000000 \in L$

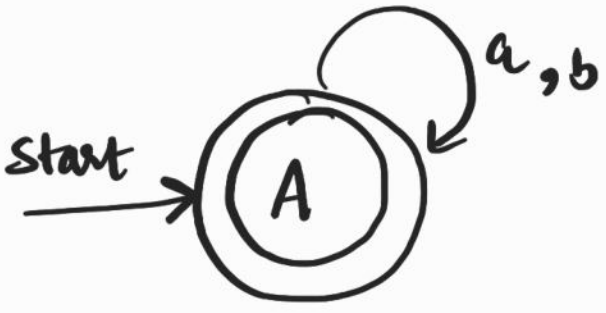
Hence not a reg language.

Q (a|b)* and ((a|b)*)*

Check if these 2 languages represent the same language or not.

(a|b)* #





followpos(1) = {1, 2, 3}

followpos(2) = {1, 2, 3}

followpos(3) = ϕ

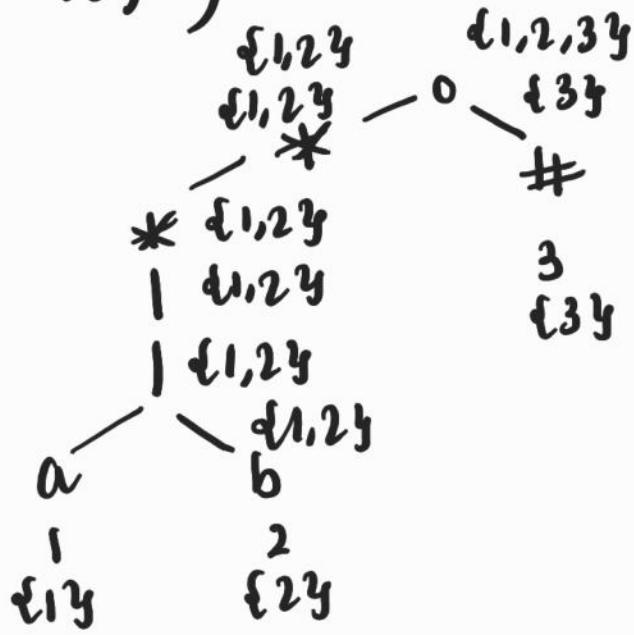
start state = {1, 2, 3} ≡ A

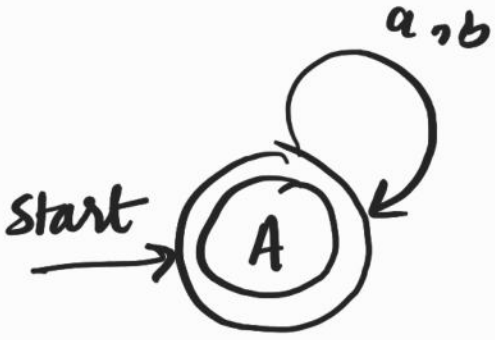
Dtran [A, a] = followpos(1) = {1, 2, 3} = A

Dtran [A, b] = followpos(2) = {1, 2, 3} = A

	a	b
A	A	A

2 $((a|b)^*)^*$





Followpos (1) = {1,2,3}

Followpos (2) = {1,2,3}

Followpos (3) = ϕ

Start state = A = {1,2,3}

Dtran [A, a] = followpos (1) = {1,2,3} = A

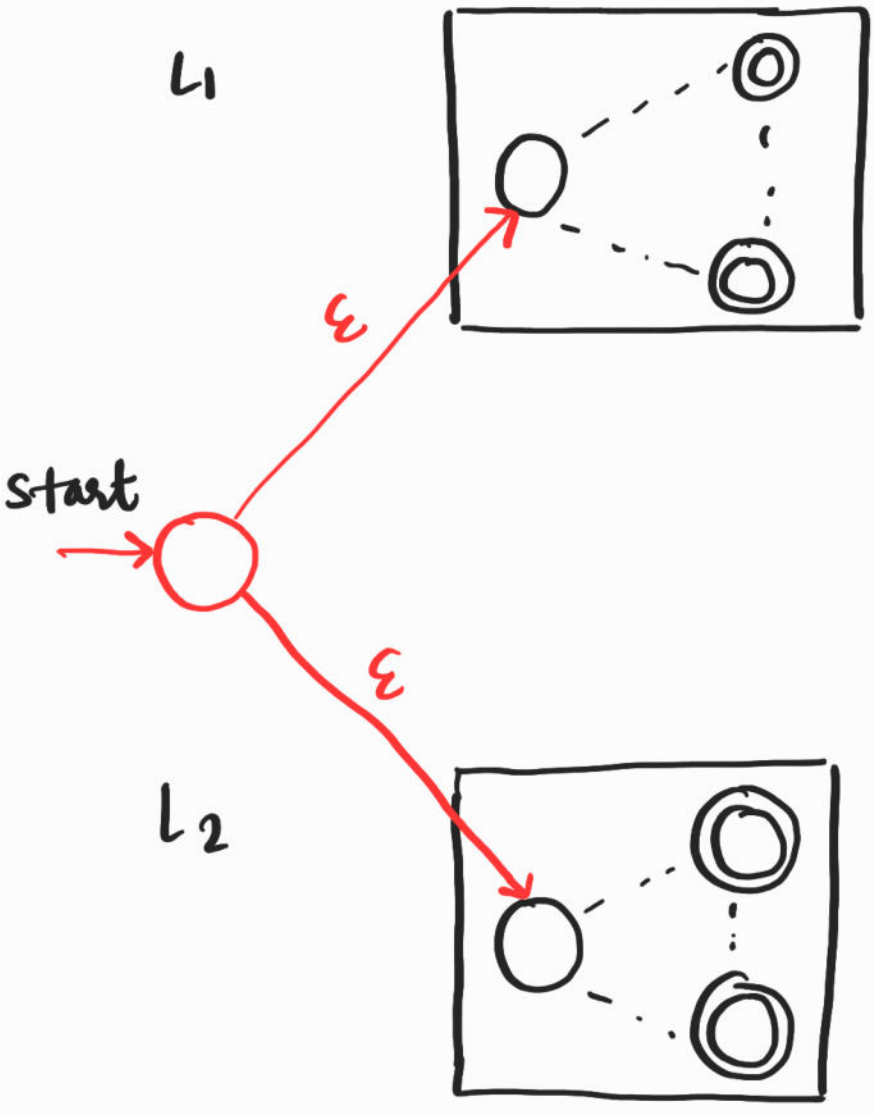
Dtran [A, b] = followpos (2) = {1, 2, 3} = A

	a	b
A	A	A

Closure Properties of Reg Languages

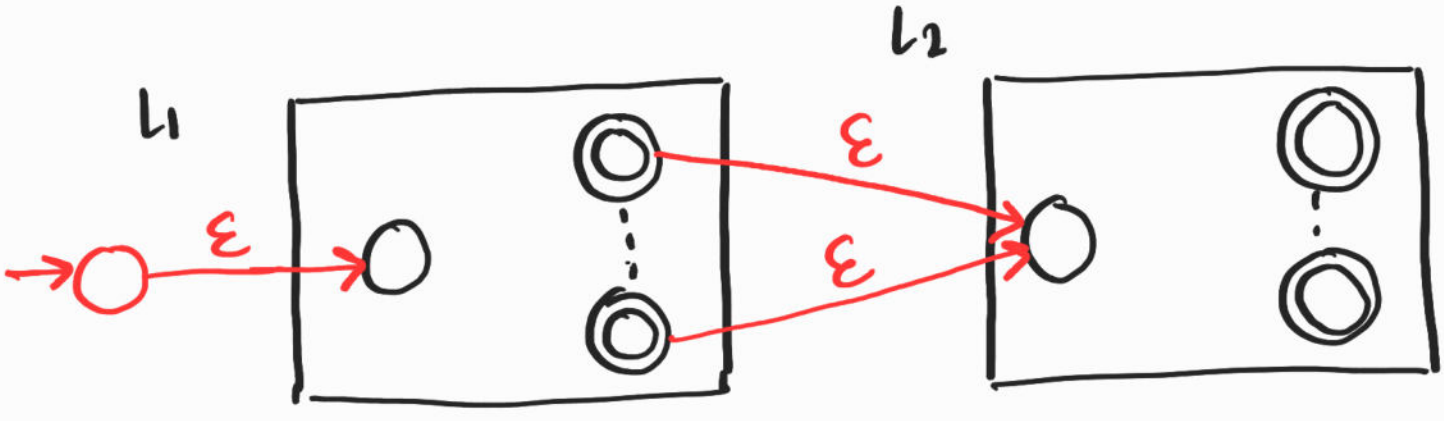
(a) Union :- let L_1, L_2 be 2 regular language. To prove $L_1 \cup L_2$ is also a regular lang.

$L_1 = \{a, b\}$ $L_2 = \{0, 1\}$ $L_3 = \{a, b, 0, 1\}$



$L_1 \cup L_2$

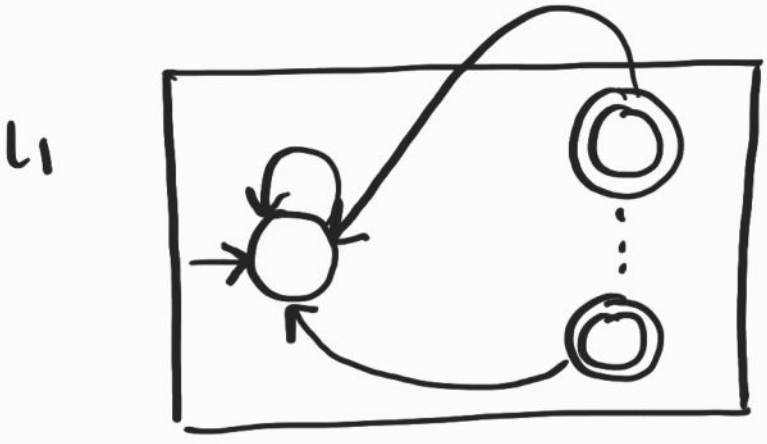
(b) Concantenation :-



$L_1 = \{aa, bb\}$ $L_2 = \{cc, dd\}$

$L_1 L_2 = \{aacc, aadd, bbcc, bbdd\}$

(c) kleen closure :-



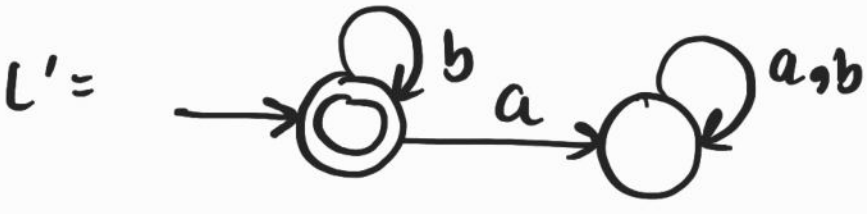
(d) Intersection

$n_a(w) \bmod 2 = 0$, $n_b(w) \bmod 2 = 0$ eg.

(e) Complement

$L = \{a, aa, ba, bba, bab\}$

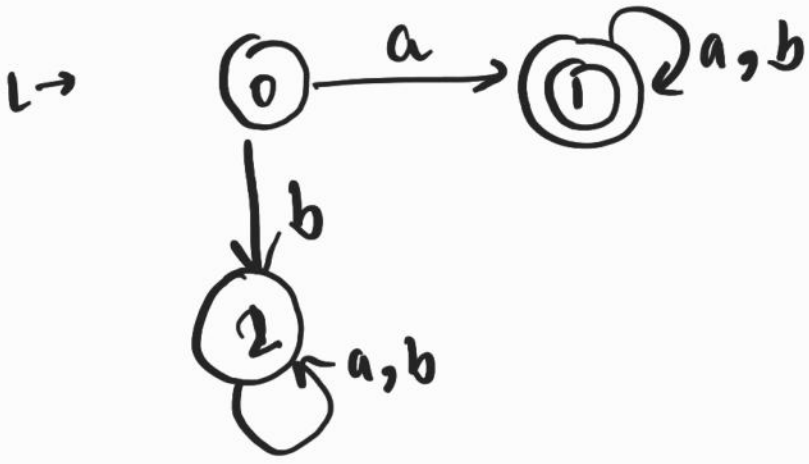
L' = does not contain 'a'



(f) Reversal

$L = \{a, abb, aab, aba\}$ Starts with A.

$L^R \rightarrow$ ends with 'a'



start state becomes final & final becomes start state.