Date: ____________ Day: M T W T F S

TOA - 1

pre-req of compiler construction.

Automata → model / machine

Language ⇒ Medium of communication.

Formal Language	**Informal Language**
⇒ set of rules defined	• verbal + gesture meanings.
⇒ well defined protocol e.g C++.	• d/f meanings on same verbal if d/f gestures are used.
⇒ only verbal meanings, same meaning each time	

Building blocks of language

(Informal) **Eng** ⇒ Alphabets (26 char) ⇒ formation of letters (dictionary)

⇒ sentence (grammar rules).

(formal) **C++** ⇒ ASCII (256 char).

⇒ words (rules) (Infinite possible).

⇒ Instructions.

• for checking valid variable names ⇒ construct rules (automata)

Date: ______ Day: M T W T F S

TOA-2 (16/01/23)

- valid & invalid string.
- D/f Language.
- Σ/set of alphabets for language.

Represent a Language

1) Descriptive.

$$\Sigma = \{0,1,\ldots,9\}$$

Language of odd no :- $\{1,3,5,\ldots\}$

can be written in set builder form.

2) Regular Expression..

- Some language are regular/some not.

language that can be represented in regular form.

3) Content free Grammar.

Descriptive form

$\Sigma = \{0,1\}$ → can form infinite strings.

$= \{0,1,00,01,000,\ldots\}$

Date: ____________ Day: M T W T F S

- Language start with 0 → L1 → {0,01,001,011,000,----}

  can be subset of other lang.

  word ⇒ valid string belonging to a language.

  every word is string. but not every string is word.

- Language start & end with d/f char → L2 ⇒ {01,10,001,--}

- Language which contains equal no of 0's & 1's → L3

  ⇒ {01,10,1100,---}.

- Language with equal 0's & 1's & 0's are formed before 1s → L4 ⇒ {01,0011,000111,---}.

  ⇒ $\{0^n 1^n, n>=1\}$.

- Palindrome Language ⇒ L5 ⇒ {0,1,00,010,---}.
- Palindrome language with length <5 ⇒ finit set.
- **Regular Expression.**
- Regular exp can be difficult.

→ Password Validation

$$\Sigma = \{0,1,\$,\#,A,\ldots,F \text{ / } a,\ldots,f\}$$

- atleast 1 special char.
- atleast 1 digit.
- atleast 1 capital char
- min length 5

⇒ {A\$001} valid.

{a\$001} invalid.

- Regular exp have some limitations.

Operators:

1) Kleen Star. $*$

Repeat 0 or more/infinite time.

2) Kleen Plus. $+$

Repeat 1 or more time.

Examples:

$\Sigma_1 = \{a\}$

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

$\Sigma_1^* = a^0 = \lambda$

$= a^2 = aa$.

$\Sigma_1^* = \{\lambda, a, aa, \ldots\}$.

$\Sigma_2^* \Rightarrow (0,1)^* \Rightarrow$ Kleen star on whole set.

$\Rightarrow (0,1)^0, (0,1)^1, (0,1)^2, \ldots$

each possible combination. ← λ, 0,1 00,01,10,11, ...

Kleen Plus $= \Sigma_1^+ = (a)^+ \Rightarrow a^1, a^2, \ldots$

(null/empty not included). a, aa, aaa, ...

Date:

Day: M T W T F S

String Operations:

- length of string. $\Rightarrow |aaa| = 3$ (cardinality operator)
- reverse string.
- concatenate string.

↳ $s_1 = a$, $s_2 = ab$.

$s_1 . s_2 = aab$.

$s_1 = \lambda$, $s_2 = ab$.

$s_1 . s_2 = \lambda . ab = ab$.

Recursive Def

↳ self repeating until base condition.

$\Sigma = \{0, \ldots, 9\}$.

Language of integers $= \{0, \pm 1, \pm 2, \ldots\}$.

1) Some strings are the part of the language.
2) Define some rule that generate the new strings of the language.
3) Any string which produce from rule 2 belongs to language other than that no string would be part of language.

Date: ____________ Day: M T W T F S

Example:

Rules for creating int language:

base string/condition

$$\Sigma = \{0, \pm 1, \pm 2$$

$$x \in \Sigma \Rightarrow x+1 \text{ or } x-1 \in \Sigma$$

Rules for creating even no language

$$\Sigma = \{2, 4, 6, \ldots\}.$$

$$x \in \Sigma \Rightarrow x+2 \text{ or } x-2 \in \Sigma$$

Polynomial

$$5x^0, 6x^3, 2x + 3x^2.$$

$$\Sigma = \{\text{Any number}, x\}.$$

If p & q are polynomial.

Then $p+q$, $p-q$, $p \times q$ are also polynomial.

$$\underbrace{2x}_{p} + \underbrace{\underbrace{3}_{p}\,\cancel{x}\,\underbrace{x}_{q}}_{q}$$

TOA-3 (22/01/23)

- 2nd thing Representation - Regular Expression.

Operators:-

1) Kleen Star (*) 0 or more rep

2) Kleen Plus (+) 1 or more rep.

Date: ______ Day: M T W T F S

3) Concatenation (.) $(a)(b) = ab$ $a.b \neq ab$.

4) Option / Choice (| , +) $(a+b|c)$

either a or b or c.

- **Represent a Language:**

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

Multiple RE are possible

1) Language which produce strings that only contain as.

$\Rightarrow \{a, aa, aaa, ----\}$ Descriptive form.

R.E $\Rightarrow a^+$

2) Language which produce strings that contains a's or b's.

D.F $\Rightarrow \{a, b, ab, ba, ----\}$.

R.E $\Rightarrow (a+b)^+$ or $(a|b)^+$

Check: **1)** $(a+b)^*$ $\Rightarrow \lambda, a, b, ----$

2) $a^*.b^*$ $\Rightarrow \{\lambda, a, b, ab, aa, bb, ----\}$.

↳ due to concatenation sequence matters & a cannot be after b.

3) $a^* + b^*$

↳ $\{\lambda, a, aa, aaa, ----, b, bb, bbb, ---\}$.

↳ only 1 variable due to choice.

Date: Day: M T W T F S

→ because choice is after * operator (producing combination).

$$\boxed{\begin{aligned}(a+b)^2 &= (a+b)(a+b). \\ &= ab, aa, ba, bb.\end{aligned}}$$

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

$\Rightarrow \{\lambda, a, b, ab, ba, aa, ----\}$

- to produce ab from this $\Rightarrow (a^* + b^*)^2$

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

$$= (a^1 + b^0)(a^0 + b^1)$$

$$= a.b = ab.$$

3) Language that starts with a.

$\Rightarrow \{a, ab, aab, ---\}$

$a.(a+b)^*$ ✓

$a.(a^* + b^*)^+$ ✓

$(a^+ . b^*)^+$ ✓

$(a^+ + b^*)^+$ ✗ ba also possible

Date: Day: M T W T F S

TOA-4 (23/01/24)

Regular Expression:

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

1) Start & end with same letter.

$\{0,1,00,11,101,111,\ldots\}$.

$R.E \Rightarrow 0.(0^* + 1^*)^+.0 + 1.(0^* + 1^*)^+.1 + 0 + 1$.

$\hookrightarrow 0.(0+1)^*.0 + 1.(0+1)^*.1 + 0 + 1$.

$\boxed{[A-Z] \Rightarrow \text{range} \rightarrow \text{class Operator}}$

2) Start & end with d/f letter.

$\{01,10,\ldots\}$.

$R.E \Rightarrow 0.(0+1)^*.1 + 1.(0+1)^*.0$.

3) Language which contains atleast 2 0's.

$\{00,100,010,001,1000,\ldots\}$.

$R.E \Rightarrow (0+1)^*.0.(0+1)^*.0.(0+1)^*$.

4) Language which contain consecutive 2 zero's.

$R.E \Rightarrow (0+1)^*.00.(0+1)^*$

5) Even no of 0's. $E = [0,2,4,\ldots]$.

$\{1,11,111,\ldots,00,001,010,101,\ldots\}$

$(1^*.00.1^*)^*$ X

$R.E \Rightarrow (1^*.0.1^*.0.1^*)^* + 1^*$

6) Odd no of 0's.

Wrong $(1^*.00.1^*)^*.0.(1^*.00.1^*)^*$.

$\{0, 01, 10, 0100, ----\}$.

7) Even no of 0's & even no of 1's.

$\{00, 0000, ---, 11, 1111, ---, 0011, 0110, 1100, 1001, ---\}$.

$R.E = (11^*.0.1^*)^*$ ✗

$= \left((00+11)^* + (01+10)(00+11)^*(10+01)\right)^*$

1000 0001

$\Sigma = \{\text{Alphabets, Digits}, \_\}$.

8) start with _ or Alphabet.

1) $(\text{Alphabet} + \_).(\text{Alphabet} + \text{Digit} + \_)^*$

Wrong 2) $((\text{Alphabet} + \_)^*.\text{Digit}^*)^*$

int a = 4501;

↳ check is it valid num/int ?
compiler has pattern stored not all the numbers.

↳ patterns have reg exp behind them.

↳ Reg expression for comment also.

Date: ____________ Day: M T W T F S

TOA-5 (29/01/24)

Construct R.E for a lang:

{ deny pattern is bit difficult }

1) that doesn't contain substring ab.

$\{\lambda, a, b, ba, aa, bb, aaa, bbb, bba, \ldots\}$.

R-E $= a^* + b^*.a^* \Rightarrow b^*.a^*$

as multiple can be after b

2) that doesn't contain substring aa.

$\{\lambda, a, ab, ba, bb, aba, bba, bbb, bab, \ldots\}$

R.E $= ab^* + b^*a + b^*$ X

$= (a+b)b^*(a+b)$ X ababab.

$= b^*(ab^+)^* + a$ X ba.

$= b^*(ab^+)^* + a + (b^+a)^*$ X aba.

$= (ab)^* + (ba)^*(b)^*$ X babb

$= b^* + a(ba)^*b^* + b(ab)^* + a$ It think ok but simplified is

RE $\Rightarrow (b + ab)^*(\lambda + a)$

Precedence of Operators:

* L to R

\+

.

+/|

3) No of a's = No of b's.

$(ba)^*(ab)^* + a(ba)^*b$.

$(ab + ba + a(ba)^*b + b(ab)^*a)^*$.

(a^*b^+)

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

not possible ⇒ no regular language.

Proving Lang through recursive def

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

Regular Lang = RL ⇒ $\{\Sigma, \lambda\}$

1) By default all Σ and λ are part of our language.

2) If r1 and r2 $\in$ RL then $r1^*$, $r1.r2$, $r1 + r2$ belongs to RL.

$a^+ = a.a^*$

TOA-6 (30/01/24)

Automation model ⇒ to construct language.

Finite Automata.

Types

یقینی 1) Deterministic FA (DFA)

غیر یقینی 2) Non-deterministic FA (NDFA/NFA)

DFA = { Σ, Q, F, q_0, δ }

Σ → Alphabets
Q → states
F → final states (more than 1 possible)
q_0 → initial state (only one)
δ → transition function.

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

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

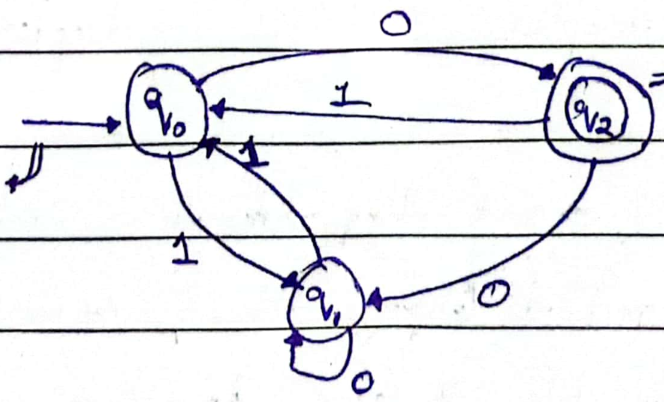
$F = \{q_2\}$.

q_0 = initial state.

Transitions. (Random)

states \ Σ	0	1
q_0	q_2	q_1
q_1	q_1	q_0
q_2	q_1	q_0

Transition Function:



⇒ final state with double circle or + sign.

Incoming arrow with initial state or − sign

q_0^- ⇒ initial state

q_1^+ ⇒ final state

If (input string after processing ends on final state):
accepted
else: rejected.

1) 0100 ⇒ ends on q_1 ⇒ rejected.
2) 110 ⇒ ends on q_2 ⇒ accepted.
q_0 $q_1 q_0 q_2$

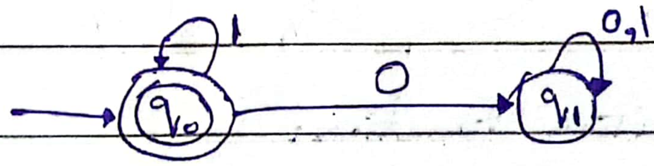
- DFA ⇒ Every letter has corresponding transition.
- NFA ⇒ Atleast one letter has no/multiple transitions.

Constructing Models:

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

$L = \{\lambda, 1, 11, 111, \ldots\}$

$R.E = 1^*$



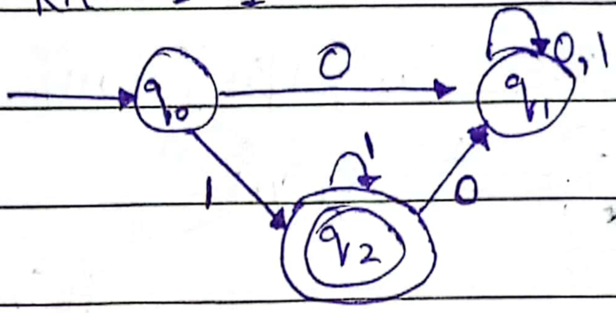
"1" ⇒ Accepted "11" ⇒ Accepted.

- When null ⇒ read nothing ⇒ remain at initial state.
- λ is part of language ⇒ so q_0 can be final state.
- for every state ⇒ define transition against each letter in sigma.

- for 0 (in this case) move to state that ~~th~~ never moves to final state as 0 is not acceptable.

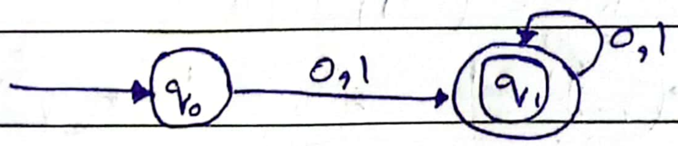
→ **L2** = {1, 11, 111, ----}.

R.E = 1^+



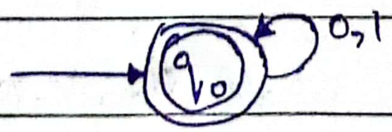
→ **L3** = {0, 1, 01, 00, 10, ----}

R.E = $(0+1)^+$



→ **L4** = {λ, 0, 1, 01, 10, ---}

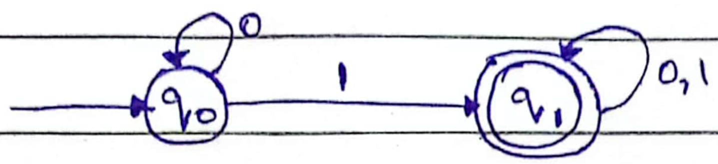
R.E = $(0+1)^*$



Date: ______ Day: M T W T F S

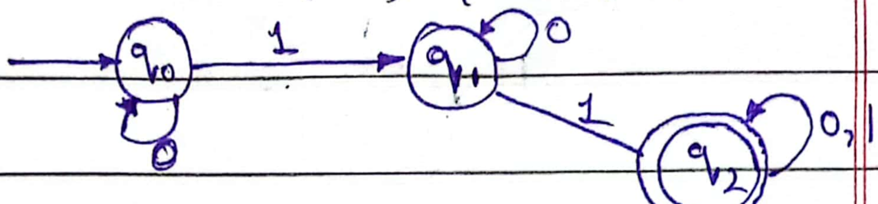
→ **L5** = Atleast one 1

R.E = $(0+1)^*\,1\,(0+1)^*$



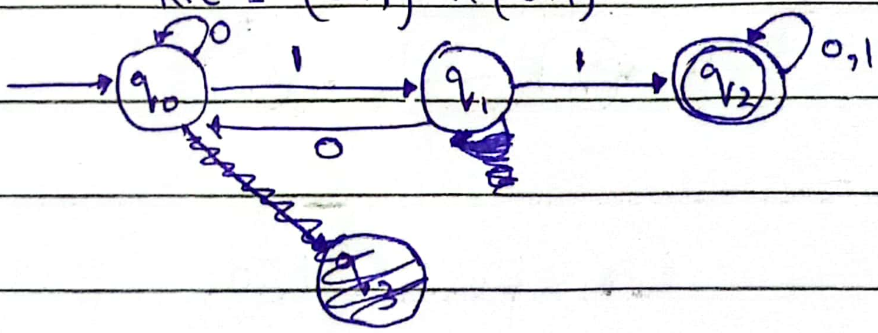
→ **L6** = Atleast two 1's

R.E = $(0+1)^*\,1\,(0+1)^*\,1\,(0+1)^*$



Extra **L** = consecutive two 1's.

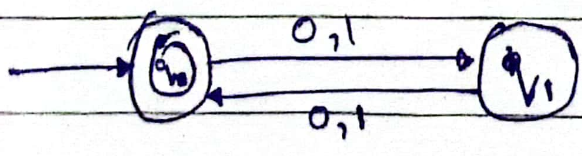
R.E = $(0+1)^*\,11\,(0+1)^*$



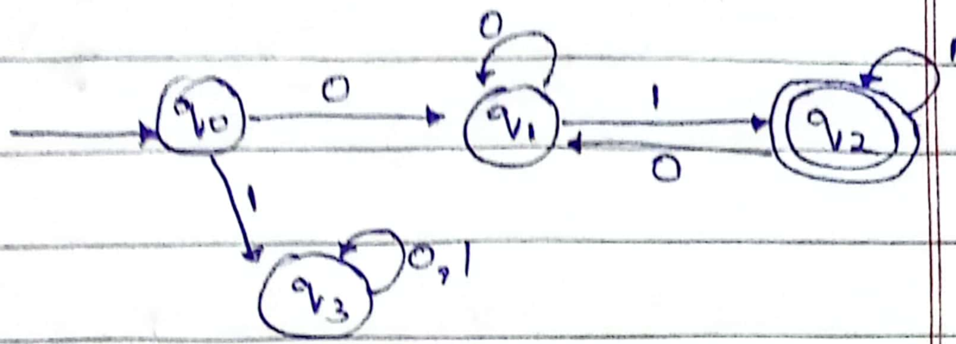
→ **L7** = Even length of string.

$= \{\lambda, 01, 00, 11, 10, \ldots\}$

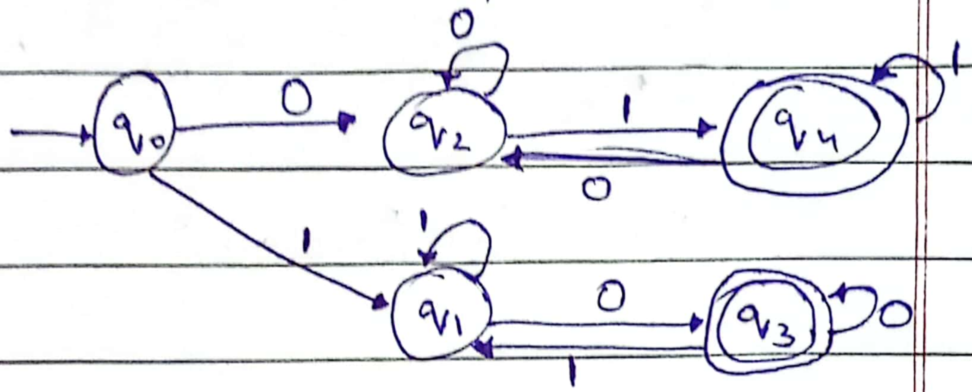
R.E = $((0+1)(0+1))^*$



→ **L8** = start with 0, end on 1.



→ **L9** = start & end with different letter.

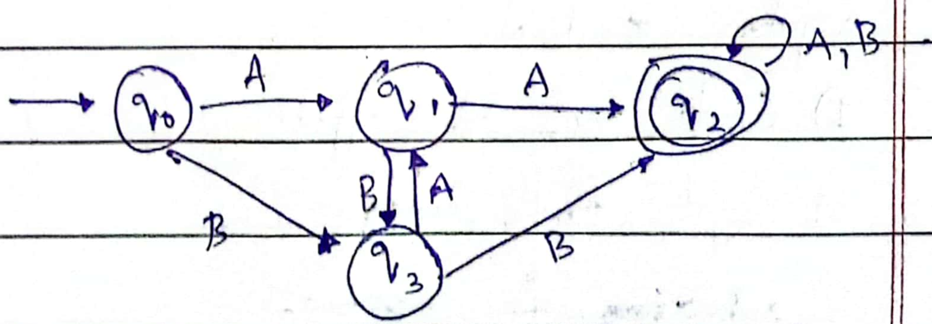


TOA - 7 (12/02/23)

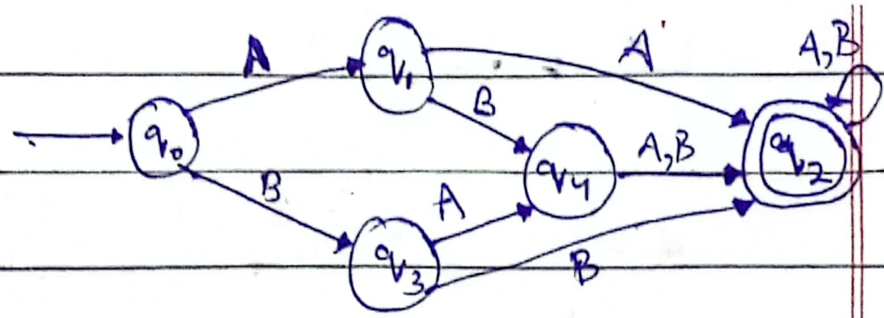
$\Sigma = \{a, b\}$

L1 = 2 consecutive A's or B's

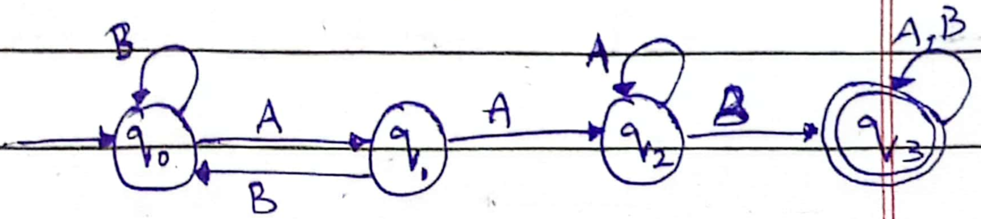
R.E = $(a+b)^*(aa+bb)(a+b)^*$



Extra → 2 A's or B's.



L2 = substring "aab" appears.



R.E = $(a+b)^* aab (a+b)^*$

L3 = All string without ab as substring.

R.E = $b^* a^*$

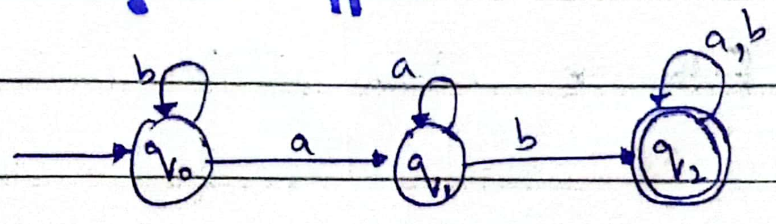
➔ for negation DFA

first create for simple then negation of final states

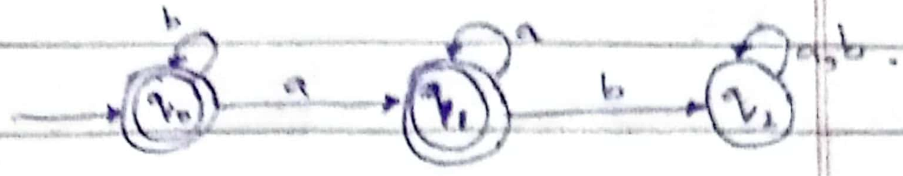
1) Create for ab appear.

2) Negate final states.

substring ab appear.

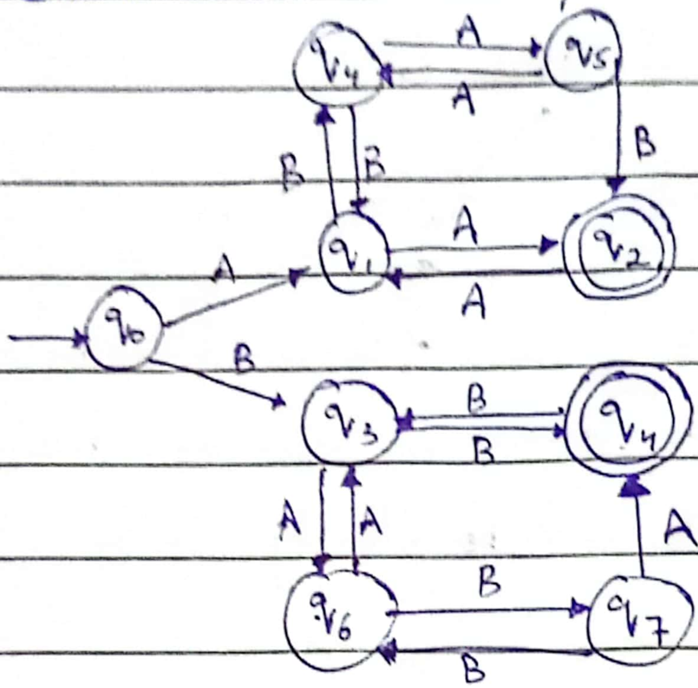


substring ab not appear

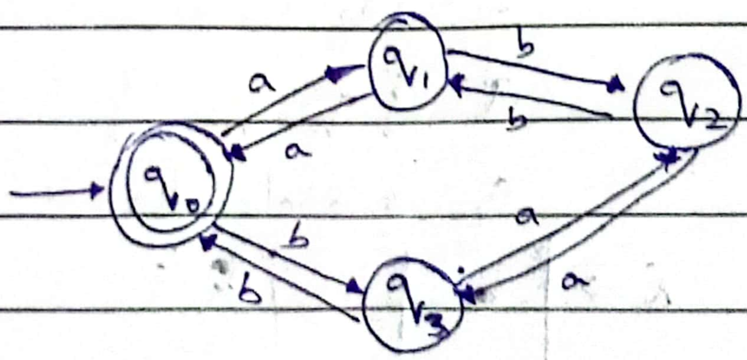


L4 → Even no of A's & Even no of B's

null not accepted



null accepted



TOA-8 (13/02/24)

F.A

1) Deterministic FA $\Rightarrow \{\Sigma, Q, q_0, F, \delta\}$
   - only 1 transition
   - no state can be skipped.

2) Non-Deterministic FA

NFA	Transition Graph.
- only 1 q_0 directly	- multiple q_0 directly defined.
	- string input can be defined.

$\Rightarrow$ NFA $\Rightarrow$ never on computational resources

$\Rightarrow$ NFA must be ultimately converted to DFA.

NFA

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

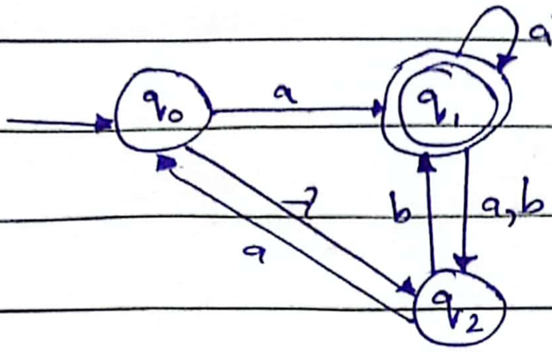
$Q = \{q_0, q_1, q_2\}$

$q_0 \Rightarrow$ Initial State $F = \{q_1\}$

$\delta \Rightarrow$	a	b	λ/ε
q_0	q_1		q_2
q_1	$q_1 q_2$	q_2	
q_2	q_1	q_1	

- missing transition + multiple transitions
- can move without reading anything.

Date: ________ Day: M T W T F S



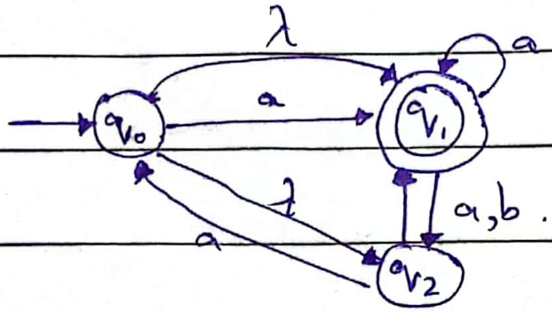
In NFA, same input can be processed by multiple paths

→ Is "b" acceptable? Yes first read λ.

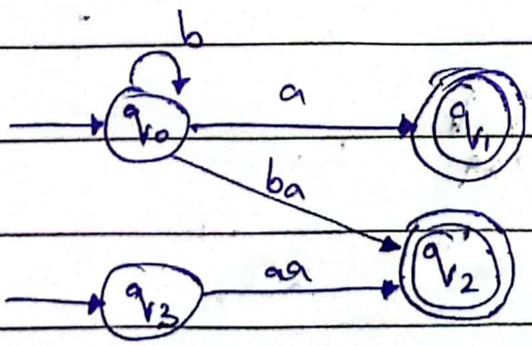
2 paths: q_0 ⇒ path crashed.

q_2 ⇒ accepted.

↳ Atleast one path to be acceptable

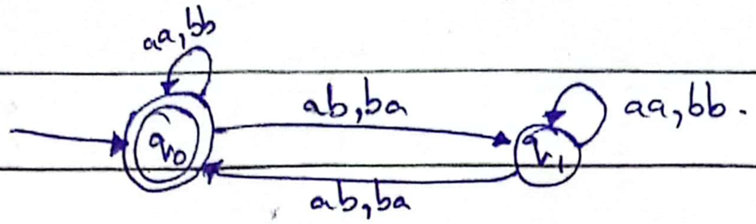


Transition Graph:



(directly more than 1 q_0 possible)

L ⇒ Even no of A's & even no B's.

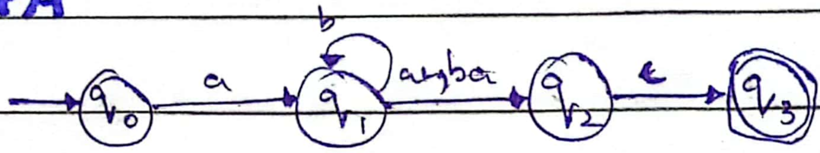


Construct DFA & NFA from R.E

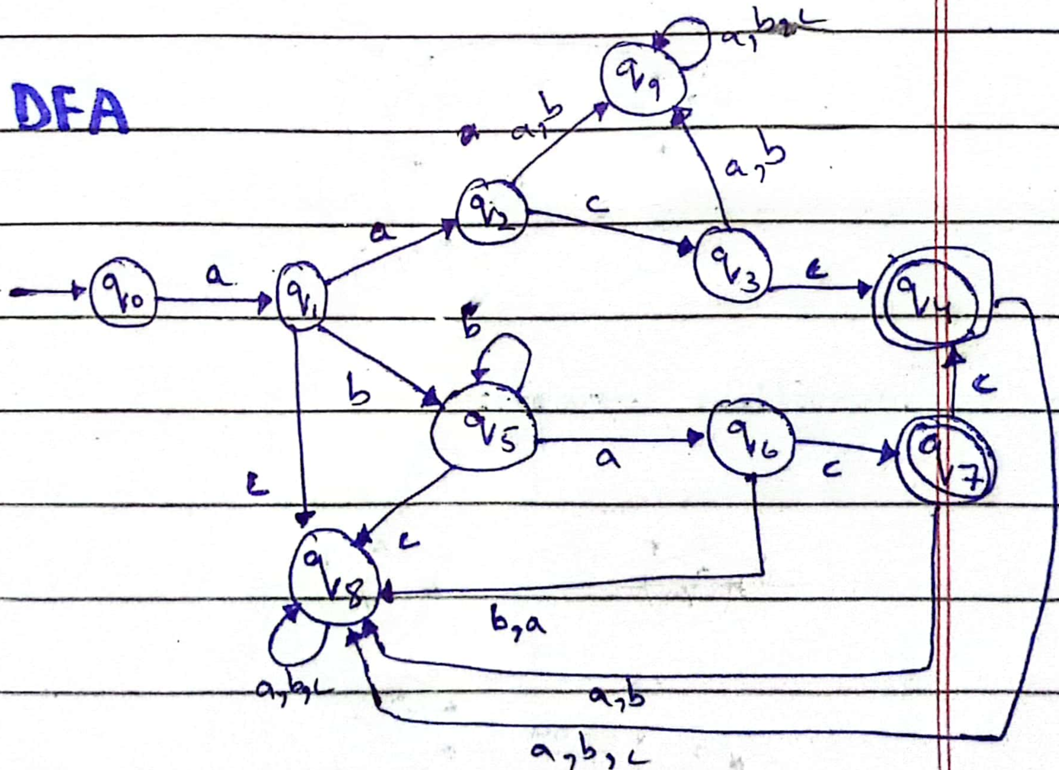
$\Sigma = \{a, b, c\}$

$R.E = ab^*(ac+ba)c.$

NFA



DFA



Kleen's Theorem:

To convert NFA to DFA → activity to perform

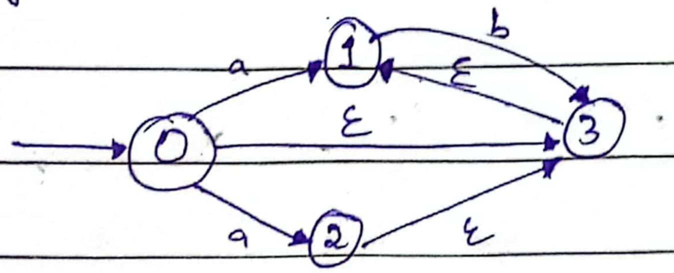
Reg Language ⇒ R.E ⇒ DFA ⇒ NFA ⇒ Transition Graph.

confirm

1) T.G can be converted to R.E
2) R.E can be used to generate NFA/DFA
3) NFA → T.G

NFA → DFA

- By subset construction method:



ε-closure:

{given all ε, λ transitions}

ε-closure{0} ⇒ {0,1,3} ⇒ S_0

{1} ⇒ {1}

{2} ⇒ {1,2,3}

{3} ⇒ {1,3}

0 1 3

Perform transition (S_0, a) ⇒ {1, 2} { as {(1, 2), (1), ()

1) ε-closure (S_0, a) ⇒ {1, 2, 3} ⇒ S_1

↳ from e.c {1}
& e.c {2}

Date: ________ Day: M T W T F S

2) ε-closure $(S0, b) \Rightarrow \{3\} \Rightarrow \{1, 3\} \Rightarrow S2$.

3) ε-closure $(S1, a) \Rightarrow \{\} \Rightarrow S3$

4) ε-closure $(S1, b) \Rightarrow \{3\} \Rightarrow S2$. $\boxed{\text{same as } S2}$

5) " $(S2, a) \Rightarrow \{\} \Rightarrow S3$ $\boxed{\text{same as } S3}$

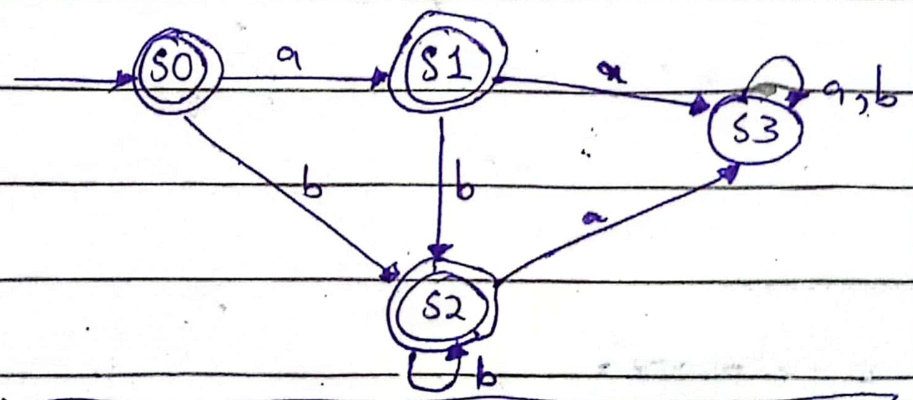
6) " $(S2, b) \Rightarrow \{3\} \Rightarrow S2$. $\boxed{\text{same as } S2}$

optional [

7) " $(S3, a) \Rightarrow \{\} \Rightarrow S3$.

8) " $(S3, b) \Rightarrow \{\} \Rightarrow S3$

DFA



$\boxed{\text{Since 3 is final state in NFA } \&, \text{ 3 is in set } S0, S1, S2 \Rightarrow \text{final state}}$

Date: ______ Day: M T W T F S

TOA - 9 (19/02/24)

Regular language is represented by either

i) TG ii) FA iii) RE

↳ if represented by one can be converted to other form.

i) FA → TG

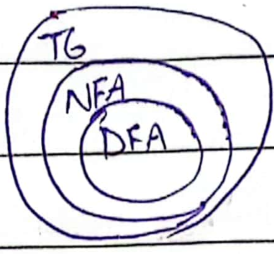
ii) TG → R.E

iii) R.E → FA

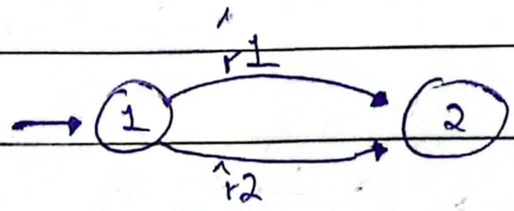
Kleen's Theorem

★ Prove these 3 parts

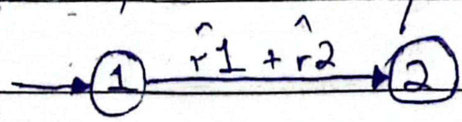
1) FA → TG



2) TG → R.E



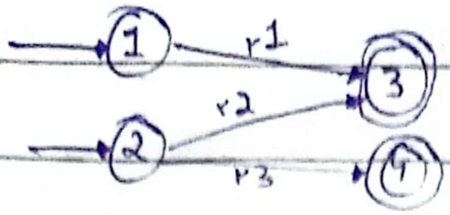
⇒ If multiple paths from one state to another, can be represented as:



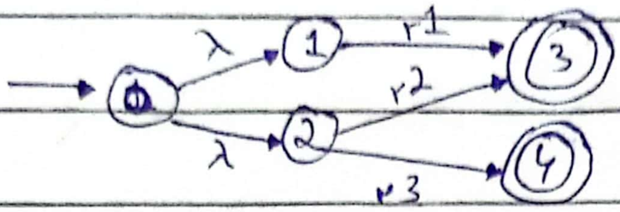
⇒ Resolve multiple initial states.

(by adding 1 state before them).

Date: Day: M T W T F S

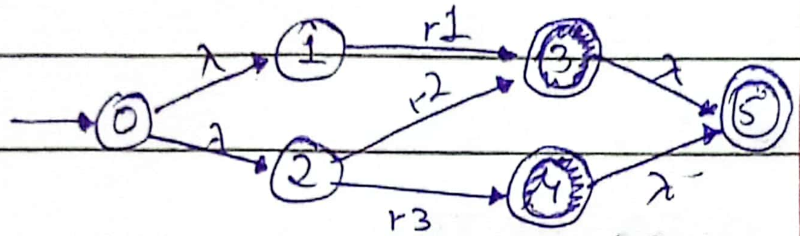


Sol:



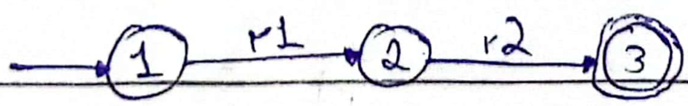
⇒ Resolve multiple final states ⇒ only 1
(by adding 1 state at end)

Sol:



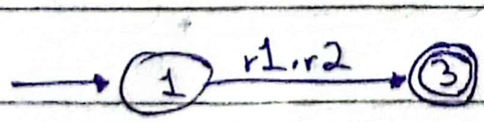
→ ~~[illegible]~~ Remove mid states 1-by-1.
↳ multiple scenarios possible.

1)

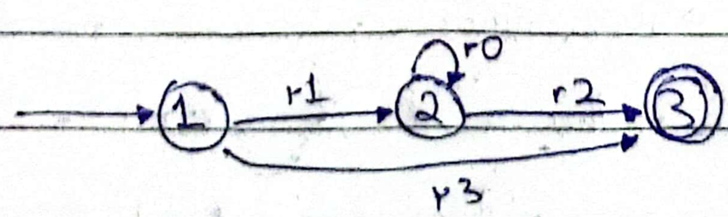


remove 2

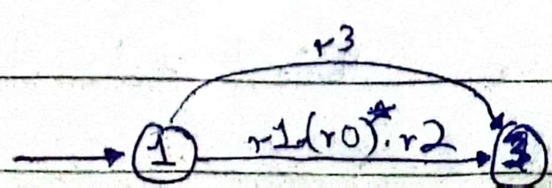
Sol:

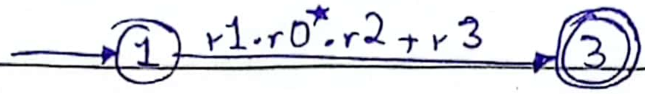


2)

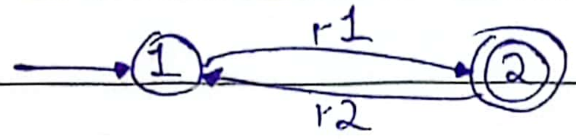


Sol:

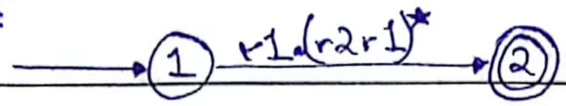




3)



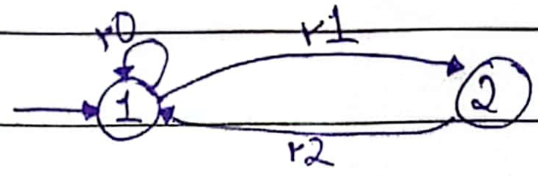
Sol:



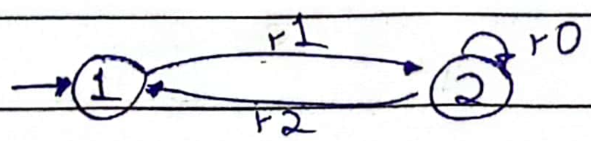
4)

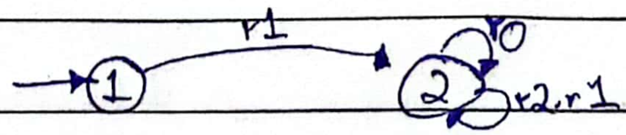
R.E $= (r1 + r2)^*$

5)



R.E $\Rightarrow$ $r0^*.r1.(r2.r0^*.r1)^*$





R.E $\Rightarrow$ $r1.(r0+r2.r1)^*$