UCS701 3L 1T

8/8/2017 website: ucs601

DATE PAGE No.

THEORY OF COMPUTATION (TOC)

Automata Theory

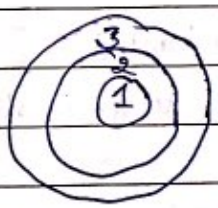
Books :→

1. Introduction to Computer Theory
   — Daniel A. Cohen
2. Introduction to Automata Theory, Languages & Computation
   — Hopcroft, Motwani & Ullman
3. Introduction to the theory of computation
   — Michael Sipser
4. Theory of computation
   — John C. Martin

state of machine

Syllabus :→

1 finite state machine
2 push down stack machine
3 turing machine → model any machine available
4 uncomputability } finding soln as close as possible
5 computational complexity
↳ problems whose no soln available but also no proof available that soln is not solvable.



a problem is solvable or unsolvable (mathematically)

* Basis for compiler design (subject)

- compiler ; assembler; interpreter

languages

Assembly lang → H/w dependent
High level lang → not H/w dependent

DATE
PAGE No.

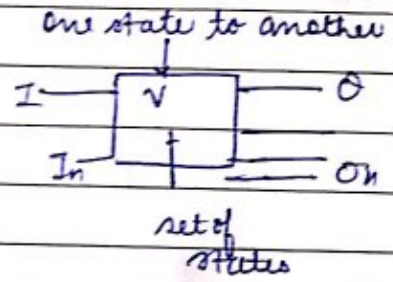
Theory of computation

How exactly, languages do their work.

lang. → recursive defn → specify language

Problem → recursive → soln should also be recursive

finite automata → simple machine



Deterministic → predefined ; fixed

Non-deterministic → dice ; uncertainity
1, 2, ...

int a = 5;

int b = 6;

sum = 11 → (always)

Deterministic finite Automata (DFA)
↳ rules are predefined

I/P → Machine → O/P
↓
change of state

fan regulator OFF, 1, 2, 3, 4, 5
↳ state change

DATE
PAGE No.

1 → 2, 2 → 3, 3 → 4
(change of states) → 3 state changes

finite : no. of states are limited
infinite :→ machine cannot be modeled

input alphabet
↳ {A ——— Z}
ε → {a ——— z} in english, characters

string
ε = { }
ε = {0, 1} → for machine lang.

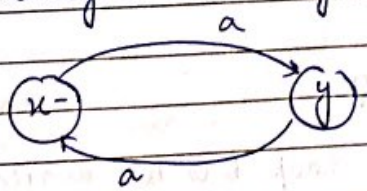
* zero followed by 1 (01)
{ 0, 1, 00, 01, 10, 11, ——— 010 ——— }

meaningful

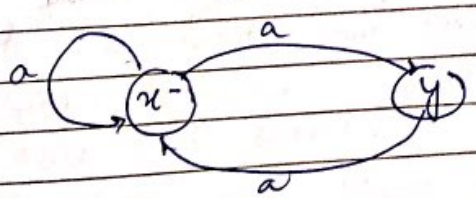
special stages → initial
→ final

final (z) (z+)
initial (x) (x-)

DFA :→ always 1 initial stage; more than 1 final stage is possible.

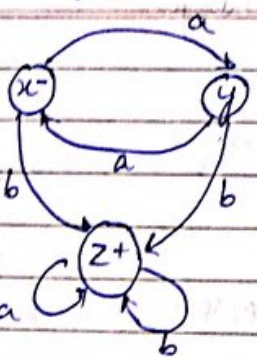


Deterministic
(unique → a)



Non deterministic any stage
(not unique as ~~O/P not~~
may be possible when
I/P is a.

stage → 3
x, y, z



final stage where machine gets halted (terminated)

Deterministic

→ Every DFA is recognizing some lang.

→ lang only has those stage, that takes it to final stage ie z^+

L = { b, ba, bb, baa, bab, ab, aab, ——— infinite }

* Machine producing lang. ——→ but contains infinite no. of languages.
↳ deterministic

→ Set will never contain duplicate elements.

More concepts :→

Regular expression
↳ short form for no. of possibilities in a language.

aba → 3 stage
↳ is final stage

abab → not final if self loop b is not available

lang → used for comm" purposes
alphabets

DATE
PAGE No.

English {a, b, — z}
{A — — Z}

comp lang · C
{ } character set

english → alphabet → words → sentence → paragraphs → stories

language → syntax { rules you need to follow }
↳ semantics { meaning associated, interpretation }

eg Name = { Akhil, Ajay, — }
semantic → meaningful

abc, pqr
↳ correct but doesn't have any meaning.

No role of semantics
Formal lang. (english, hindi) → Natural language
↳ are alphabets ↳ requires some form of intelligence

- rules you want shud be explained explicitly
- no assumption (like abc can't be a name)
- no intelligence is reqd.

NLP

- input char must be finite; lang may be infinite

- Rules { defined explicitly }
you have to define the rules.

Σ → Latin character : contains set of characters
Σ = { — }
↳ form which you generate
↳ defin

$\Sigma = \{x\}$

$L = \{x, xx, xxx, \_\_\_\_\}$ ← possible strings

DATE
PAGE No.

* collects strings of any possible scenario
  → language

only even no. of x

$L = \{xx, xxxx, \_\_\_\_\}$

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

$L_1 = \{0, 1, 00, 01, 10, 11, 000, 001\}$

contain atleast one 0

$L_2 = \{0, 00, 01, 10, 000, 001 \_\_\_\_\}$

* some operations can be implied
  → concatenation
  → length
  → reverse.

$L_4 = \{x, xx, xxx, xxxx \_\_\_\_\}$

a b

ab → xxxxx

it is not necessary after processing operation, the string will belong to the ~~same~~ language.

length →

zero kind of string

$\wedge$ → null string

01 → c

reverse (c) = 10

DATE
PAGE NO.

Two operations that we want:

Kleene closure ⟶ a set of string including null string and of any order or of any length
Kleene star (*)

$\Sigma = \{x\}$

$\Sigma^* = \{ \wedge, x, xx, xxx, xxxx, \text{——} \}$

↳ 0 or more

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

$\Sigma^* \{ \wedge, 0, 1, 00, 01, 10, 11, 000, 001, \text{——} \}$

$\wedge$ creates a lot of problem, so subset is taken of Kleene closure ie.

Σ^+

sigma plus.

$\Sigma^+ = \Sigma^* - \wedge$

↳ 1 or more

$L_1 = \{ x, xx, xxx, xxxx, \text{——} \}$

$L_1 = \{ x^n, n = 1, 2, 3 \text{——} \}$

means

$x^1 = x$
$x^2 = xx$
$x^3 = xxx$

} occurence

lang → collection of strings (can be finite or infinite)

$L = \{ x, xx, xxx, xxxx \text{——} \}$

$L = \{ x^n = n = 1, 2 \text{——} \}$

DATE
PAGE No.

* Recursive defn :→
eg factorial
$n! = n * (n-1)!$

* Recursive problem → recursive s/n → always beneficial
Tower of Hanoi → Recursive
↳ difficult with interactive method

steps :→
i) identify the base object
ii) defni rules to identify new object in the set using existing objects.
iii) Declare that, no other object can be the part of the set other than specified by rule1 or rule2.

EVEN defne an even no.
{2, 4, 6, 8 — }
{x, x is divisible by 2}

recursive defn →
i) 2 is even no.
ii) If x is even no. then x+2 is also even.
iii) step iii)

→ Language # Finite Automata
↳ operations
→ recursive defn New machine needs to be modeled first.

TOC → study some concepts
create models.
Models will be used in development of machine
computer system → most complex machine
Turing machine

length (Λ) = 0
∴ length (x) = 0
∴ x = Λ

$x^0 = \Lambda$ not 1

$3x^2 + 7x - 9$

Rule 1 — 3 is polynomial
Rule 2 — x is polynomial
Rule 3 — 3(x) "
Rule 3 — 3(x)(x) "
Rule 1 — 7 is "
Rule 2 — x is "
Rule 3 — 7(x) "
Rule 3 — $3x^2 + 7x$ "
Rule 1 → −9 "
Rule 2 — $3x^2 + 7x + 9$ "

factorial defn.
Rule 1. 0! = 1
Rule 2. (n+1)! = (n+1)(n!)
n! = n(n−1)!

16/8/17

Finite Automata →

In any lang, we req. alphabets

english {a — z} ← I/P alphabets
{A — Z}
ε = {a}

Now we'll create strings

No Boundations. {a, aa, aaa, aaaa — }

even as = {aa, aaaa, — }

* lang may contain finite no of strings or infinite

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

$L_1 = \{a, ab, ba, b\}$

$L_2 = \{aa, bb, ab, b, c\}$

Another lang. $L_1 \cup L_2 = \{a, ab, ba, b, aa, bb\}$

We need to model (concept) machine before implementation

Finite Automata → most simplest machines

→ input alphabets

→ states

→ transition functions.

Deterministic (finite) finite automata

(By default)

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

fan regulator: 6 states (off, 1, 2, ——— 5)

* If you have to go to 6 from 2, you'll go like 2→3, 3→4, 4→5, 5→6

* Machine cannot be modeled, if it has infinite state.

Then DFA not possible

$Q = \{$ $\}$ // set of contains states that

But should be finite 10,000, ——— 50,000

* capital :→ set
small :→ elements

$Q = \{q_0, q_1, q_2 \text{ —— } q_n\}$ → states (set)

DATE
PAGE No.

$q_0 = q_0$ → previous

→ whenever we use any machine, it must be in some initial state
q_0 = initial state

always part of set of states

$F = \{\ \}$

states that are final

specific states → where we receive O/P / desired O/P

q_0, F { if always well defined }

There can be always one initial state but final states can be more than 1

$q_0 \in Q$ (Belongs to)

$F \subseteq Q$ (subset)

sigma Σ = input alphabet

String can be formed from Σ

δ → transition function

input → state → will be changed to some other stage (Transition)

	State / Input	Input a	Input b
initial	x	y	z
	y	x	z
final	z	z	z

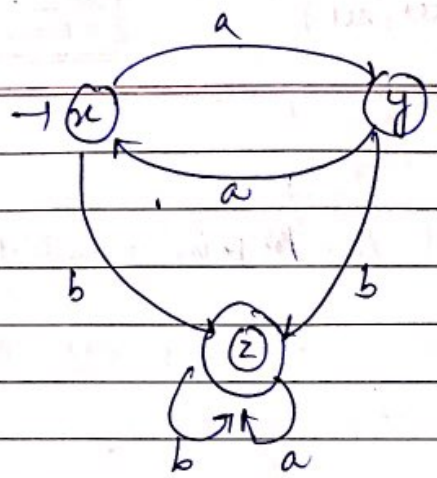
$Q = \{x, y, z\}$

$\Sigma = \{a, b\}$ → I/P characters

$q_0 = x$

$F = \{z\}$

→ initial
◎ final
– initial
+ final



⇒ Transition diagram

infinite

→ aab // If thru this string we can reach the final string than the string will be acceptable

$\delta(x,a) = y$
$\delta(y,a) = x$
$\delta(x,b) = z$

acceptable strings will be lang

aaa
$\delta(x,a) = y$
$\delta(y,a) = x$
$\delta(x,a) = y$

→ string not acceptable
∴ not part of language

* strings recognized by machine shud be part of lang.

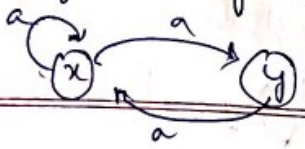
lang = {a, b, ba, bbb, baa, bbbbb ...}
— ✗ ✓

after i/p stage is joined to y (which is not a final stage)

due to ~~self loop~~ after b any no. of ab will be acceptable

{b, ba, bb, baa, ——ab, aba, abb, aab}

finite : fixed no. of states



DATE
PAGE No.

{x → a} = x
{x, a} = y [Non deterministic]

NDFA → more powerful than DFA → always be equivalent to some NDFA.
Non-deterministic Finite Automata

- We have some methods to find the equivalent DFA from NDFA.

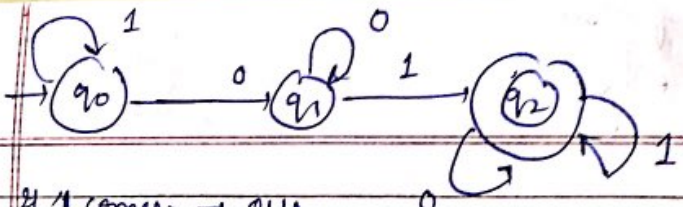
DFA.

accept all strings with substring '101'

Q = {q0}
Σ = {0, 1}
δ =
q0 =
F =

0, 1, 00, 01, 10, 11, 000, 001
✗ ✗ ✗ ✓ ✗ ✗ ✗ ✓

- Note → try to limit no. of states.
- your machine should only fulfil the language not more than the strings that shud be required or not fulfilling the required condition.



DATE
PAGE NO.

* If 1 comes → our criteria is not fulfilled

→ Regular languages.

22/08/2017

Regular Expression :→

language
recursive def^n
finite automata
→ DFA
→ NDFA

C → strings → instructions → function → programs

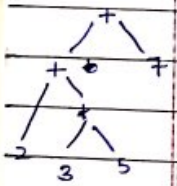
* when compiler process program ↓ firstly tokenize

(TOC → Base of compiler instruction)

Parsing → Compiler point of view

int is a string having special meaning

2 + 3 * 5 + 7



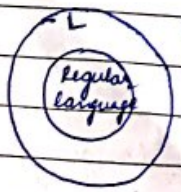
if 7 missing after '+' incomplete tree ∞ error

alphabet always finite. ; no. of strings formed may be infinite or finite

alphabet → $\Sigma = \{a, b\}$ or $\Sigma = \{0, 1, 2\}$

$\Sigma = \{a\}$

language :→ $L_1 = \{a, aa, aaa, aaaa, ----\}$



* If some set consists of all languages, then it consist of some subset called regular language also

* Regular grammer for regular language.

* lang not regular lang can not be represented by regular expr. represented by regular expr.

context free lang; context sensitive lang.

finite automata → simple machines

* How to write regular expression

[only these 3 operators are used]

1 concatenation . ab, ac → abacc (dot is optional)

2 closure * → occurence of any string from zero to more.

3. OR + a+b (either a or b)

( ) : parenthesis

Kleene closure. Σ = {a} → occurence of one element. → {a, b} (not {aa, bb etc.})

Σ* = {Λ, a, aa, aaa, aaaa, ----} (wrt some set)

Null string : ∈, Λ

Λ = {a, b} → elements

name of set

a* (wrt some element)

occurence of a zero or more times.

a* = {Λ, a, aa, aaa, aaaa, ----}

For one element Σ* is same to a*.

* Why are we using regular expr?

lang. may be finite or infinite.

writing as {a, aa, aaa, —} is not reliable or convenient as ... is not acceptable. ∴ regular expr. used.

Σ = {a}

L1 = {Λ, a, aa, aaa, aaaa, ——}

→ Regular expr. = a*

L2 = {a, aa, aaa, aaaa, ...}

→ R.E. = a(a*) = aa*

occurence of a exactly once

R.E.

L = {Λ, aa, aaaa, aaaaaa, aaaaaaaa}

(aa)* is diff. from aa*.

Rules for writing Regular expression :-

Rule 1 : All input alphabets are regular expr. Λ is also regular expr.

Rule 2 : If r1 & r2 are R.E. then r1r2, r1 + r2, r1*, (r1) are also R.E.

Rule 3 : No other expr can be regular expr. if it is not following Rule 1 Rule 2.

→ $\Sigma = \{a, b\}$

R.E. $r_1 = a$, $r_2 = b$, $r_3 = \Lambda$

DATE
PAGE No.

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

$L_3 = \{a, ab, abb, abbb, \ldots\}$

$r_1 = a$, $r_2 = b^*$

$r_1 r_2$ = R.E. $= ab^*$

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

$L_4 = \{$ string starts and ends with a $\}$

↳ $\{a, aa, aba, aaa, \ldots\}$

~~a+a~~

to include a also we OR a.

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

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

$L_5 = \{$ strings over a & b and all a's (if any) come before all b's (if any) $\}$

$a^*.b^*$ or a^*b^*

~~a*b+a*b*a~~

$(ab)^*$ ↘ diff. from it

↓

$\{\Lambda, ab, abab, ababab, \ldots\}$

—×—

23 August, 2017.

DATE
PAGE No.

→ Regular expression (regular lang ; regular gramm)

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

R.E. represented as bold letter

R.E. ≡ **a** ⟶ This set contains only one element

→ Rule 1 : all I/P alphabets are valid R.E

Now the above R.E. will represent $\{a\}$

↳ single occurence

$L = \{\wedge, a, aa, aaa, aaaa, \ldots\}$

⇓ R.E.

a^*

R.E. ≡ ab ⟶ $\{ab\}$ ↳ single occurence of ab bcoz. no operator is used.

R.E. ≡ aa $\{aa\}$

* so for every finite lang, then it is always possible to write R.E for it.

R.E ≡ $(a+b)$ ⟶ $\{a, b\}$

r_1 r_2

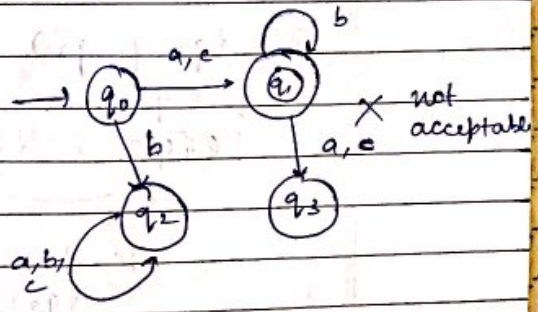
R.E ≡ $(a+b)^*$ ⟶ $\{\wedge, a+b,$

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

\# Strings start with a or c followed by any no. of b's.

$L = \{a, c, ab, abb, cb, cbb, \ldots\}$

R.E. = $(a+c)\,b^*$



Kleene closure ← $\varepsilon+$ → not applicable in R.E.

$\varepsilon = \{a, b\}$ null.

set of all strings that have an 'a' somewhere

R.E. = $(a+b)^* a (a+b)^*$ ✓

✗ $[(a+b)^* a^* (a+b)^*$ ⟵ it includes Λ.]

Λ & b are not part of this language

DFA & NDFA

→ finite automata

↳ it is DFA.

[Deterministic Finite Automata]

DFA is represented by 5-tuple $(Q, \varepsilon, \delta, q_0, F)$

Q = set of states

ε = input alphabet

δ = transition function

q_0 = initial state (only one initial state)

F = set of final state

Modeling is first, then implementation

$[q_0 \in Q]$

$[F \subseteq Q]$

→ It represents some DFA.

state / ε	a	b
→ q_0	q_1	q_2
q_1	q_0	q_2
q_2	q_2	q_2

arrow represents this is an initial state

$\delta \equiv Q \times \varepsilon \to Q$

(result is element of Q)

transition function is represented by this table.

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

$q_0 = q_0$

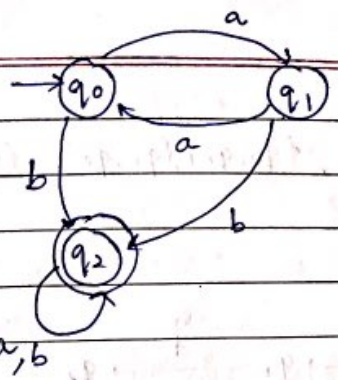
$\varepsilon = \{a, b\}$

δ:

$q_0 = q_0$

$F = \{q_2\}$

NDFA



Is aaba is acceptable as a string to this DFA or not?

Yes.

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = q_0$

$\delta(q_0, b) = q_2$

$\delta(q_2, a) = q_2$ → final state.

After processing the string completely, if machine is in final state, then string is acceptable.

aaa → not acceptable

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = q_0$

$\delta(q_0, a) = q_1$

→ Deterministic (behaviour of m/c is deterministic

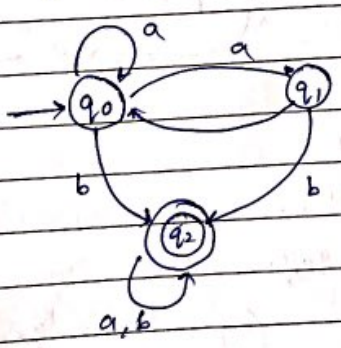
If we have dice, we know any of {1, 2, 3, 4, 5, 6} will be the result)

DFA

finite → no. of states are finite.

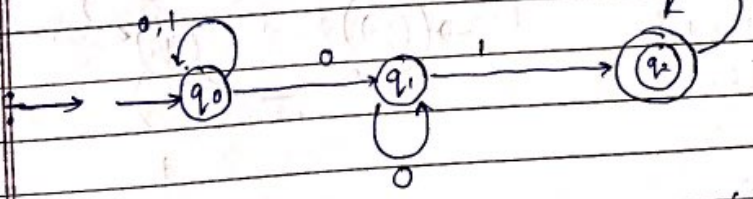
We cannot model if m/c has infinite no. of states.

* Probability of falling down of fan → zero



NDFA → non-deterministic

→ if at same state for same I/P diff. transitions possible



NDFA

$\delta \equiv Q \times \Sigma \rightarrow 2^Q$

(power set (set of all subsets) $)

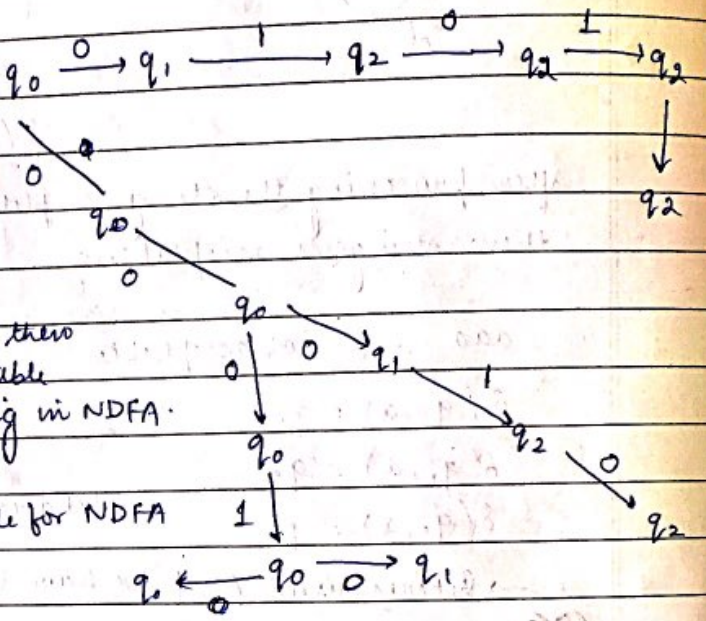
* NDFA is also represented by 5-tuple $(Q, \Sigma, \delta, q_0, F)$

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

$2^Q = \{\phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_1, q_2\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}\}$

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

0 1 0 1 0



If following any path, we reach final state, then it is acceptable as a string in NDFA.

NDFA always more powerful than DFA.

A DFA is always possible for NDFA

X

NDFA to DFA

$Q \times \Sigma \longrightarrow Q$

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

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

$Q \times \Sigma \longrightarrow 2^Q$

state / Σ	0	1
$\rightarrow (q_0)$	q_0	q_1
q_1	q_1	q_0, q_1

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

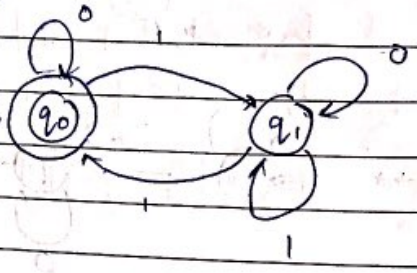
$Q = \{q_0, q_1\}$

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

$\delta =$

$q_0 = q_0$

$F = \{q_0\}$



DFA

$A' = (Q', \Sigma', \delta', q_0', F')$

square brkts. represent equivalence Class.

$Q' = 2^Q$ * they are simply a symbol not subsets externally.

so

Possible states but not necessary all states used will be $Q' = \{\phi, [q_0], [q_1], [q_0, q_1]\}$ → use square brackets.

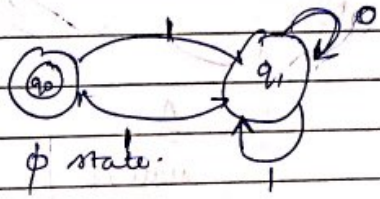
either give other name to states i.e. 'q0', q1'

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

$\delta' =$

No defⁿ for q_0 when I/P is 0 & → then m/c is —.

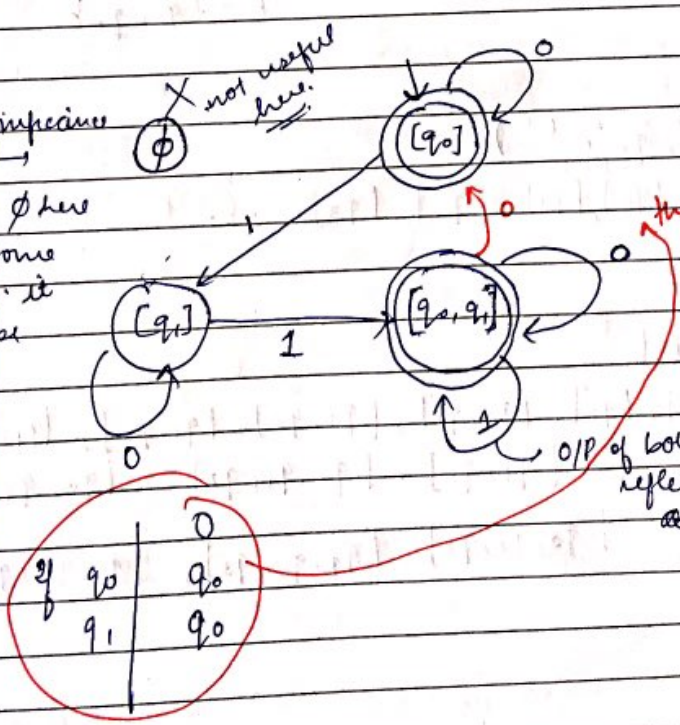
↳ it represents that m/c is in ϕ state.



$q_0' = [q_0]$

& $F' = \{[q_0], [q_0, q_1]\}$ → this included bcoz it contains q_0 which was final state of NDFA.

here no significance of ϕ → no use of ϕ here but in some other ques. it may be used



$[q_0, q_1]$ → here behaviour of q_0 & q_1 is applied/ copied here

At 0 → $q_0 \to q_0$

$q_1 \to q_1$

$\therefore (q_0, q_1) - \{q_0, q_1\}$

O/P of both reflected at 1

	0
→ q_0	q_0
q_1	q_0

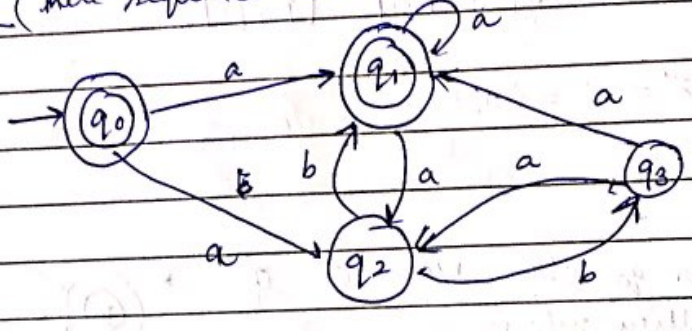
DATE
PAGE No.

When large no. of states given in NDFA.

NDFA
4 states.

DFA
2^4 states = 16. (max)
↳ some of states may not be useful

- So make transition table instead of transition diagram
  ↑ (Here sequence doesn't matter)



A
NDFA

A

state/Σ	a	b
→ (q0)	q1, q2	-
(q1)	q1, q2	—
q2	—	q1, q3
q3	q1, q2	—

$A \equiv (Q, \Sigma, \delta, \Sigma_0, F)$

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

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

$\delta =$

$q_0 = q_0$

$F = \{q_0, q_1\}$

$A' \equiv (Q', \Sigma', \delta', q_0', F')$

$Q' = \{\phi, [q_0], [q_1], [q_2], [q_3], \ldots\}$

$\Sigma' = \Sigma$

$d \equiv$

$q'_0 = [q_0]$

$F' = \{[q_0], [q_1], [q_0, q_1], [q_0, q_2], [q_0, q_3], [q_1, q_2], [q_1, q_3], [q_0, q_1, q_2], [q_0, q_1, q_3], [q_0, q_2, q_3], [q_1, q_2, q_3], [q_0, q_1, q_2, q_3]\}$

This is reqd. DFA.

DATE
PAGE No.

δ' =

state	a	b
ϕ	ϕ	ϕ
$[q_0]$	$[q_1, q_2]$	ϕ
$[q_1, q_2]$	$[q_1, q_2]$	$[q_1, q_3]$
$[q_1, q_3]$	$[q_1, q_2]$	ϕ

these states are final state

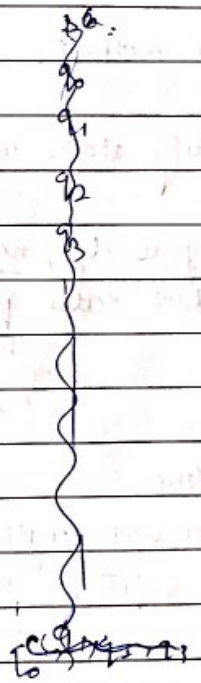
define every new state

→ now choose new state & write its possible transitions.
eg here $[q_1, q_2]$ used instead of $[q_1]$

go like this till no new state is found. or all states are done

∴ so it will have 4 states

H Minimization of DFA (in next class)



29 Aug, 2017

Minimization of DFA :-

If NDFA → convert to DFA.

If in DFA, not required states → then minimize it

$A = (Q, \Sigma, \delta, q_0, F)$. $Q = \{q_0, q_1, q_2, \dots\}$

We have to construct A' which is equivalent to A

↳ recognize same language.

$A' = (Q', \Sigma', \delta', q_0', F')$

equivalence method.

→ identify states which have same behavior :- belong to same equivalence class

Def 1 : general ; no restriction on length of string (maybe infinite). ← infeasible, can't be checked

$q_1 \equiv q_2$ ↳ If two states q_1 & q_2 are there, these q_1 & q_2 are equivalent if they recognize same string of any length

Def 2 (use def 2) ↓

k-equivalence

• same behaviour of q_1 & q_2 : $\delta(q_1, x)$ and $\delta(q_2, x)$

↳ x is a string of any length

→ equivalence wrt length of string.

$q_1 \equiv q_2$ upto the length k.

Four steps for minimization of DFA :

π_k — π subset of Q, k upto length k of string

→ Q divided into subsets & that subset shud have same behaviour as Q upto length k of string.

$Q \equiv \{q_0, q_1, q_2, q_3, q_4\}$

$\pi_0 = \{\{q_2\}\ \{q_0\ q_1\ q_3\ q_4\}\}$ → behaviour same as Q for string of length 0.

π_1

π_2

π_3

K is a finite no.

$\pi_K \longrightarrow \pi_{K+1}$

→ We generate π_{K+1} using π_K.

Recursive procedure for minimisation of DFA

Step1 Construction of π_0 (Base step)

step 2 construction of π_{K+1} from π_K
for $k = 0, 1, 2 \ldots$

Step 3 construct π_{n+1} upto
$\pi_{n+1} = \pi_n$
[for 8-10 states, only do upto π_3.]

Step 4. Minimum states

$\pi_0 =$ {{set of all final states}{set of all non-final states}}

Step 1.

→ If on any state, we process zero length of string, we rest at same state

input alphabet) Σ → always finite

Step 2. If q_1, q_2 belong to same subset; then $\delta(q_1, a) =$ } whether they lie in same subset.
$\delta(q_2, a) =$ }

If they lie in diff. subset, then don't lie in same subset

Step3 When we can't partition further, then stop.

* Transition table more useful here. so if Transition diagram is given convert it to transition table.

Next Page

* If NDFA, then multiple entries in a cell of transition table.

DATE
PAGE No.

DFA:

state/Σ	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2
(q_2)	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

Step 1

$\pi_0 = \{\{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}\}$

no further partition; ↳ so no need to check

use π_0 for it

$\pi_1 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$.

→ for $\delta(q_0, 0) = q_1$ ✓ ; $(q_0, 1) = q_5$ ✓ same subset

for $\delta(q_1, 0) = q_6$; $(q_1, 1) = q_2$

↳ from diff subset

so for string of length 1, q_1 & q_2 behaviour's diff; so they can't belong to same subset.

→ Now check behaviour of q_0 & q_6; if behaviour same then no need to check for q_3 & q_5.

Behaviour of q_0 & q_3 is diff. so q_3 can't be in same subset as q_0

behaviour of q_3 is not same as q_1 too. coz for $0 \to q_1 \to q_6$
$0 \to q_3 \to q_2$

→ just check q_4 with q_0 coz it belongs to same; no need to check other. diffsets.

$\pi_2 = \{\{q_2\}\ \{q_0, q_4\}\ \{q_6\}\ \{q_1, q_7\}\ \{q_3, q_5\}\}$

↓

$\pi_3 = \pi_2$.

these are no. of min no. of states.

$A' = (Q', \Sigma', \delta', q_0', F')$

$Q' = ([q_0, q_4], [q_1, q_7], [q_2], [q_3, q_5], [q_6])$

merging states

$\Sigma' = \Sigma$

q_0' = that particular subset which contains q_0.

⟶ $[q_0, q_4]$

F' = all those equivalence states which contain final state in it.

$[q_2]$

δ' =

state / Σ	0	1
→ $[q_0, q_4]$	$[q_1, q_7]$	$[q_3, q_5]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
$[q_2]$	$[q_0, q_4]$	$[q_2]$
$[q_3, q_5]$	$[q_2]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_0, q_4]$

30 August, 2017

ϵ or $\wedge$ (null) string with zero length

DATE
PAGE No.

Whenever there is $\wedge$ moves, then that is NDFA.

There is no DFA with $\wedge$ moves.

ϵ-NDFA :→ NDFA consisting of ϵ moves.

⇓ finding equivalent DFA.

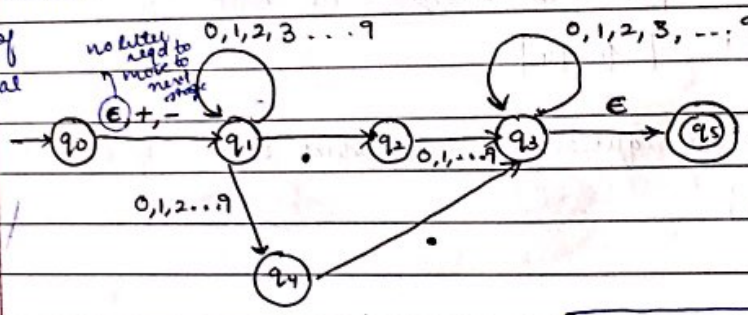
DFA → representing same lang.

- NDFA with null moves → null move removal.
- Regular expr. equivalent to given DFA & vice versa.
- DFA accepting same lang as represented by given R.E.

Motive :→ How to create equivalent ϵ-DFA from NDFA. ?

Obj of this NDFA :→ Don't focus on how NDFA was created

possibility of creating a decimal no.



NDFA bcoz at 0 on q_1 it can either remain at q_1 or can move to q_4.

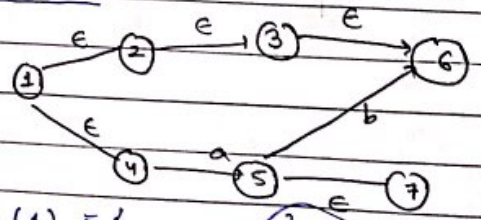
$\Sigma = \{+, -, 0, 1, 2, \ldots 9, .\}$

ϵ or $\wedge$ is not part of Σ

I/P alphabet for this NDFA = $\Sigma \cup \epsilon$

sometimes DFA we get is minimized ; sometimes not !

Epsilon closure :→



ϵ-closure (1) = { } atleast one element

bcoz every state can remain in same state without giving any i/p (using ϵ i/p) so ϵ-closure(1) always has atleast one elmt

what other states are reachable using ϵ-moves.

DATE
PAGE No.

trivially

Eclose(1) = {1, 2, 3, 6, 4, 6}
Eclose(2) = {2, 3, 6}
Eclose(3) = {3, 6}
Eclose(4) = {4}
Eclose(5) = {5, 7}
Eclose(6) = {6}
Eclose(7) = {7}

ϵ-NDFA $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$

||

DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ → 2^{Q_E} (max states) ← sab aayegi zaroori nahi

1. Formation of Q_D → $S \subseteq Q_E$, $S \in Q_D$ → power set → all possible subsets of Q_E are potential candidates for Q_D.
2. Formation of q_D = Eclose(q_0), $q_0 \in Q_E$
3. Formation of F_D = first as same as previously done. → $\{S \mid S$ is in Q_D and $S \cap F_E \neq \phi\}$ — if some F_E is included in subset then that subset is part of F_D
4. Formation of δ_D → which states are reachable.

4) $\forall a \in \Sigma$ $\delta_D(S, a)$ $S \in Q_D$ [S is subset of Q_E & part of Q_D]

(a) Let $S = \{P_1, P_2 \cdots P_K\}$

(b) compute $\bigcup_{i=1}^{k} \delta_E\{P_i, a\}$

let set be $\{r_1, r_2 \cdots r_m\}$

(c) then $\delta_D(S, a) = \bigcup_{j=1}^{m} \text{Eclose}(r_j)$

ϵ-NDFA $Q_E = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
$\Sigma = \{+, -, 0, 1, 2, \ldots 9, .\}$
δ_E - ✓ (prev. dia.)

$q_0 = q_0$
$F_E = \{q_5\}$

DFA $Q_D = \{\{q_0, q_1\}, \ldots\}$ $2^6 = 64$ states
Σ = same as ϵ-NDFA
q_D = Eclose(q_0) = $\{[q_0, q_1]\}$ ← symbol of state

not necessary that all states will be final states.

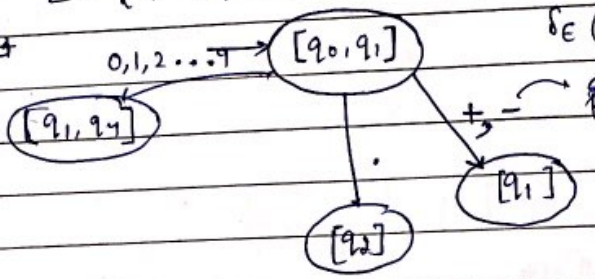
$F_D = \{[q_5], [q_0, q_5], [q_1, q_5], [q_2, q_5] \ldots\}$

$\Sigma = \{+, -, ., 0, 1, 2, \ldots 9\}$ $\quad a = +, -$

$S = \{q_0, q_1\}$ $\quad \delta_\epsilon(S, a) = \{q_1\}$ reachable by +

$\epsilon\text{close}(q_1) = \{q_1\}$

Dry run



firstly do everything for initial state & then move forward

$a = .$

$S = \{q_0, q_1\}$

$\delta_\epsilon(S, .) = \{q_2\}$

$\epsilon\text{close}(q_2) = [q_2]$

$a = 0$ (same for 1-9)

$S = \{q_0, q_1\}$

$\delta(S, 0) = \{q_1, q_4\}$

$\epsilon\text{close}(q_1) = [q_1]$

$\epsilon\text{close}(q_4) = [q_4]$

Now process for $[q_0, q_1]$ is complete.

eg in Ullman.

DATE
PAGE No.

Finite Automata & regular expression

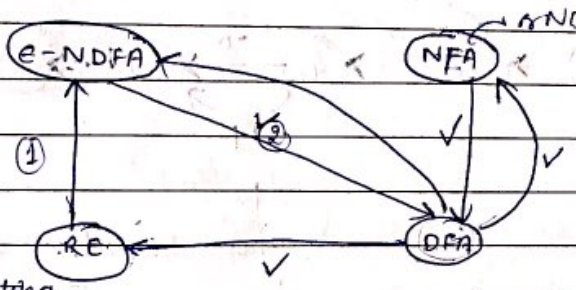
∈-NDFA to DFA

FA
- DFA
- NDFA
- ∈-NDFA
- RE

→ L = { } ≡ R.E ∈-NDFA

→ L = { }

* There is No DFA with null moves.



both representing same lang.

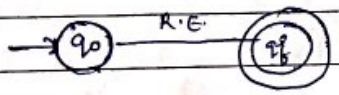
RE to DFA

RE

* → 1
\+ → 1

Step 1. RE to ∈-NDFA

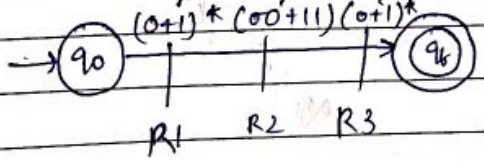
Step 2. Remove ∈ moves from ∈-NDFA to achieve DFA.

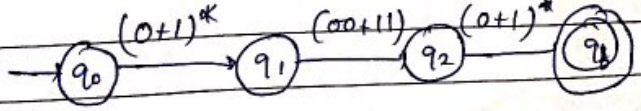


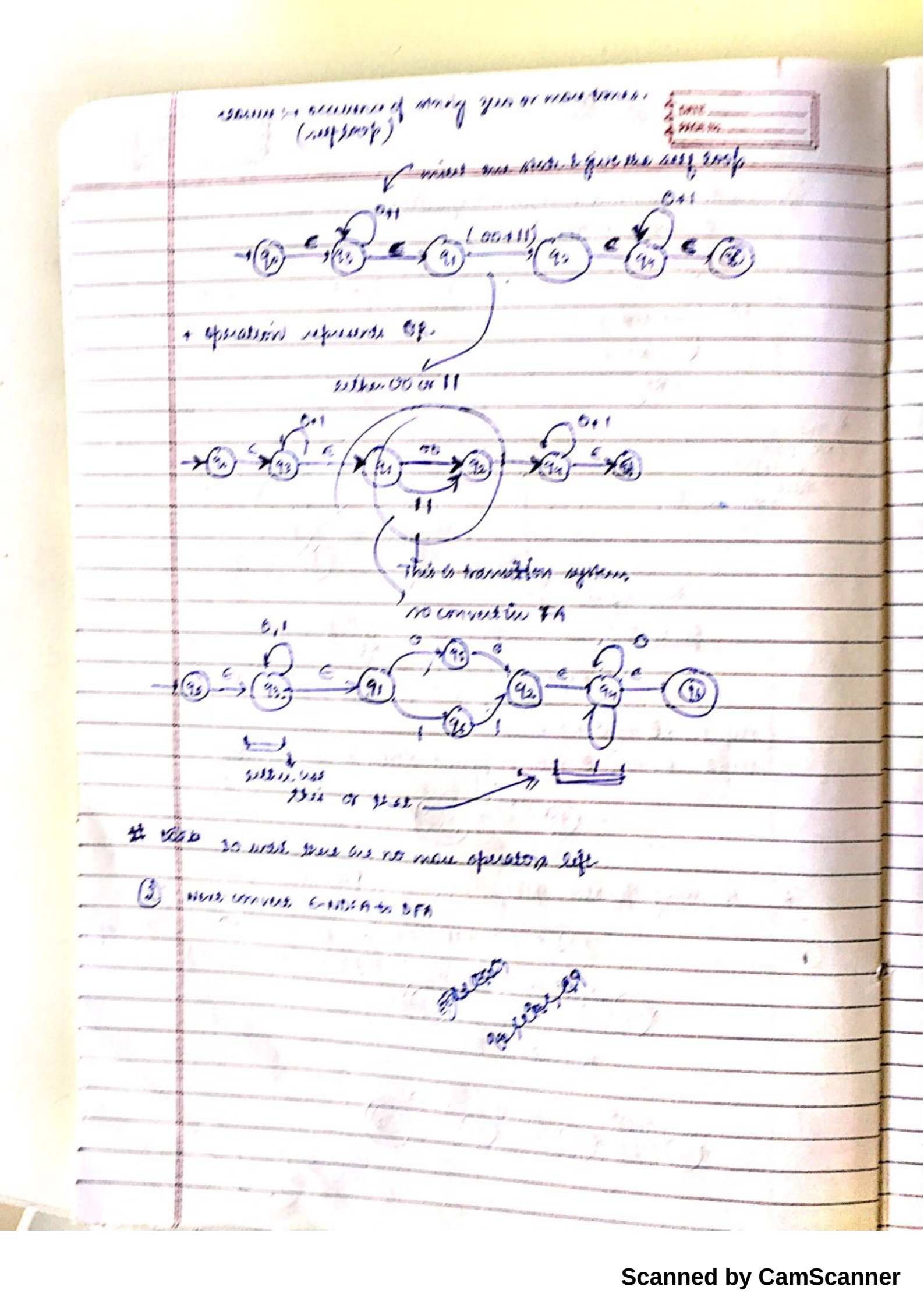
* for every operator, specific action is performed.

① $(0+1)^*(00+11)(0+1)^*$ → representing some finite or infinite lang

3 parts (R1 · R2 · R3)

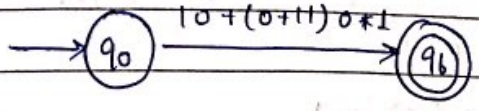


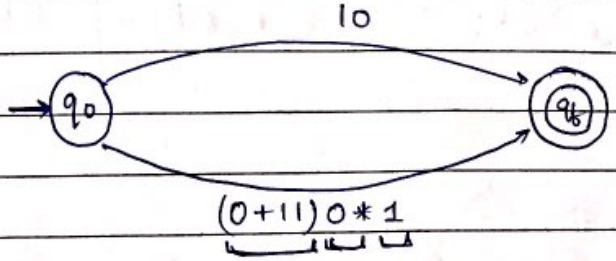


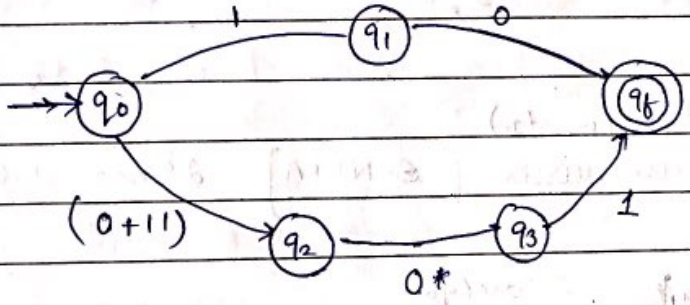


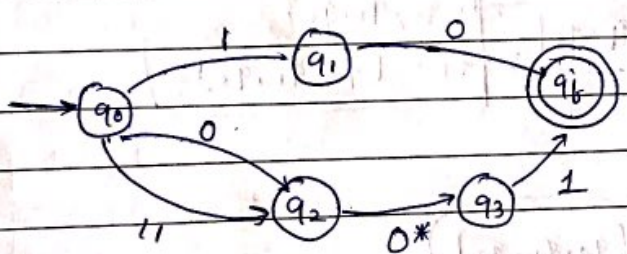
DATE
PAGE No.

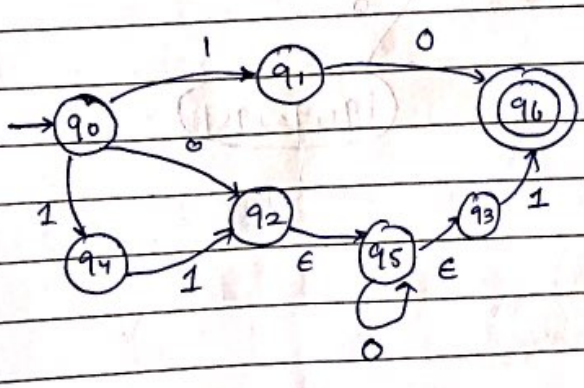
$10 + (0+11)0*1$











1) Next convert ϵ-NDFA to DFA

DATE
PAGE No.

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

ϵ-NDFA

$E = (Q_E, \Sigma, d_\epsilon, q_0, F_\epsilon)$

$Q_E = (q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_f)$

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

$q_0 = q_0$

$F_E = q_f$

δ = just eg

DFA

$D = (Q_D, \Sigma, \delta_D, q_D, f_D)$

Q_D = not [all possible subsets of ϵ-NDFA] 2^8 max states

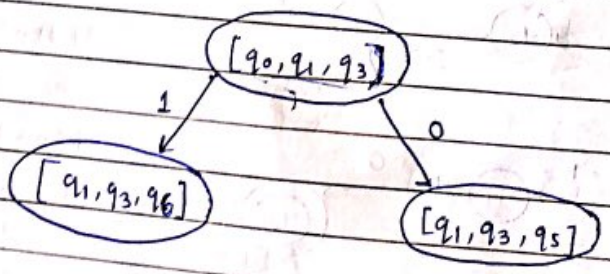
$\Sigma = \{0, 1\}$

$q_D = \{q_0, q_3, q_1\}$ — ϵclose(q_0) = $\{q_0, q_1, q_3\}$

$q_D = [q_0, q_1, q_3]$

$f_D = \{[q_f], [q_1, q_f], \ldots\}$

↳ $f_D \subseteq Q_D$



(f_{q_0})

$\forall a \in \Sigma$ $\quad S \subseteq Q_E$

$\delta_D(S,a) = ?$

$\delta_E(S,a) = \{\quad\}$

Let this set be $\{r_1, r_2 \dots r_n\}$

$\delta_D(S,a) = \bigcup_{i=1}^{m} \in close(r_j)$

$a = 0$

$S = \{q_0, q_1, q_3\}$

$\delta_D(S,0) =$ ____

$\delta_D(S,0) = \{q_3, q_5\}$

$\in close(q_3) = \{q_3, q_1\}$

$\in close(q_5) = \{q_5\}$

$\in close(q_3) \cup \in close(q_5) = \{q_3, q_1, q_5\}$

$a = 1$

$S = \{q_0, q_1, q_3\}$

$\delta_D = (S,1) =$ ____

$\delta_D \in S,1) = \{q_3, q_6\}$

$\in close(q_3) \cup \in close(q_6) = \{q_3, q_1, q_6\}$

$a = 0$

$S = [q_1, q_3, q_5]$

$\delta_D(S,0) =$ ____

$\delta_D(S,0) = \{q_5, q_3, q_2\}$

→ given language → $L_1 = \{ab, bc\}$

$L_2 = \{a, b, ab, aa, ba, bb\}$

find out.

$a \rightarrow L_1 \cup L_2$ ⟶ {ab, bc, a, b, ab, aa, ba, bb}

$b \rightarrow L_1 . L_2$

$c \rightarrow L_1^*$

{aba, abb, abab, abaa, abba, abbb,
bca, bcb, bcab, bcaa, bcba, bcbb}

$(ab+bc)^*$. 2

{Λ, ab, bc, abab, abbc, bcab, bcbc, ababab, abbabbc, ...}

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

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

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

(d) b^*ab^*

(e) $b^*(\Lambda+a)b^*$

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

(g) $b^*ab^*ab^*$

(h) $b^*(\Lambda+a)b^*(\Lambda+a)b^*$

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

Λ, ab, bc, abab, abbc, bcab, bcbc, ababab, abbabbc, ...

Λ, ab, bc, abbc, bcab.

1 1 2

Λ, ab, bc, abbc,

~~$(a+b)^* = \{\Lambda, a, b, ab, aa, ba, bb, \ldots\}$~~

DATE
PAGE No.

~~$\Sigma, a, b, ab, aa, ba, bb, \ldots$~~

(2) Given, $\Sigma = \{a, b\}$, write down regular expr. for the following

a) That accepts all strings that starts with a

b) " starts with a & ends with b

c) " having atleast one a.

d) " having exactly one a

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

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

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

d) $b^* a b^*$

e) " having atmost one a

f) " atleast two a

g) " exactly two a

h) " atmost two a

i) " having starting & ending different letter.

(e) $b^*(\Lambda + a) b^*$

(h) $b^*(\Lambda+a) b^*(\Lambda+a) b^*$

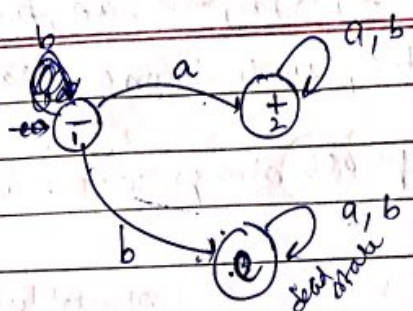
(i) $a(a+b)^* b + b(a+b)^* a$

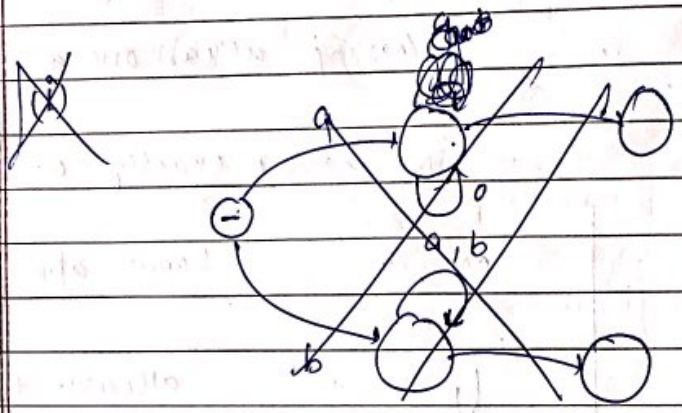
$(a+b)(a+b)^*(a+b)$

a a

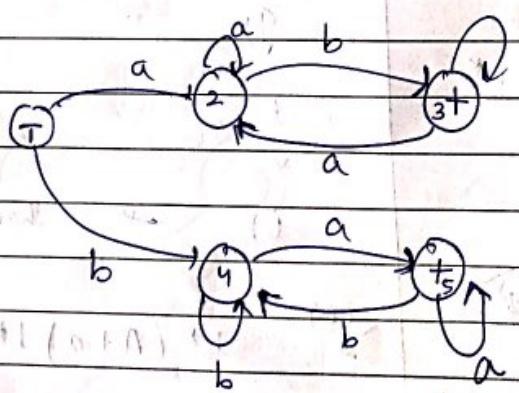
$a(a+b)^* b + b(a+b)^* a$

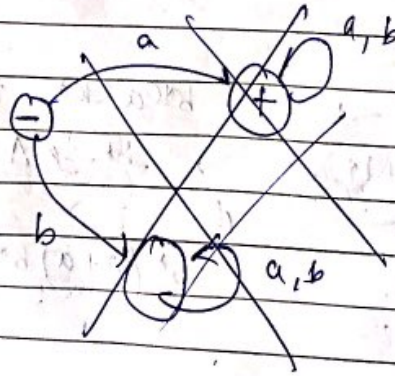
(a)





(i)





(upto 9) Lecture 16

DATE
PAGE No.

DFA to regular expr

→ state elimination method (restricted)

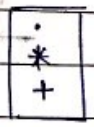
→ Algebraic method using Arden's method (general)

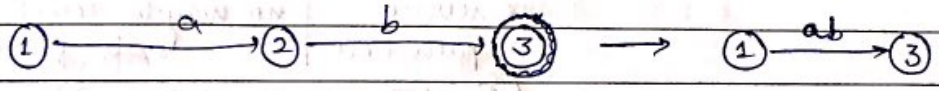
ε-NDFA to R.E.

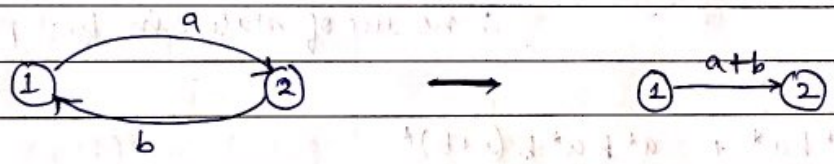
↓

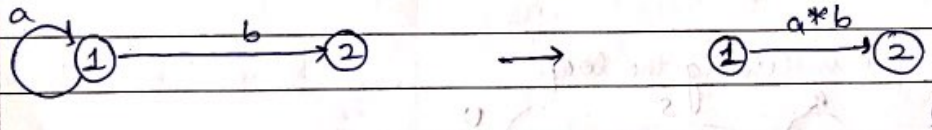
convert to DFA first

eliminate (2)

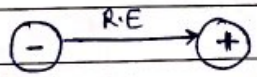




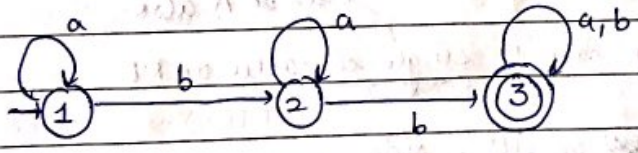




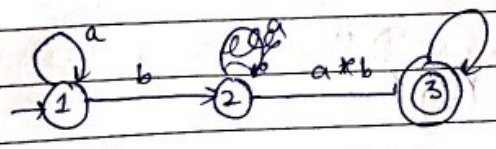
* end up as



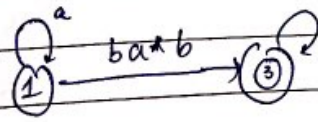
eg



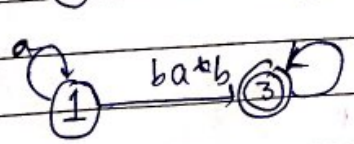
R.E. a = $a^* b a^* b (a+b)^*$

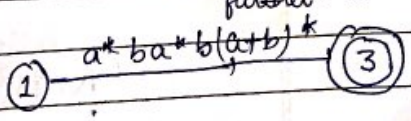


don't eliminate initial & final states



* It is possible that these R.E. be simplified further

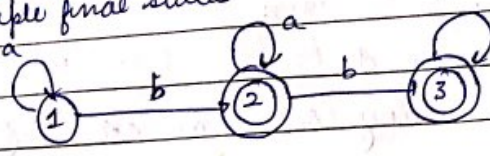




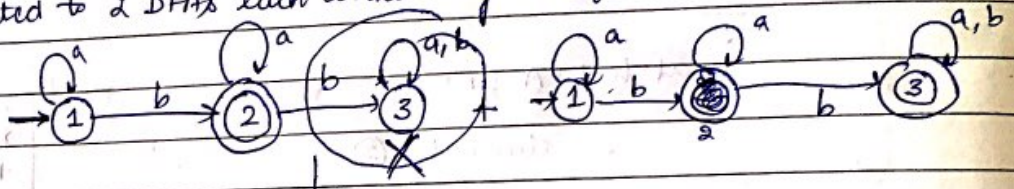
* state elimination in Ullman → complex

DATE
PAGE No.

* Multiple final states



→ converted to 2 DFAs each containing one final state



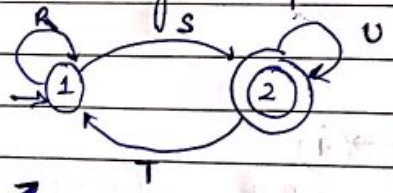
no reverse path here | no way to reach back to final state

(final to non-final state?)

∴ no use of state 3 in first part

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

* If we are including the loop

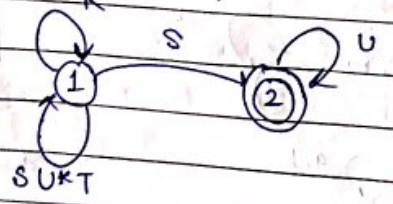


reverse edges in DFA; loops then that DFA ends up like this

R, S, T, U may be single character or R.E. → can be Λ also

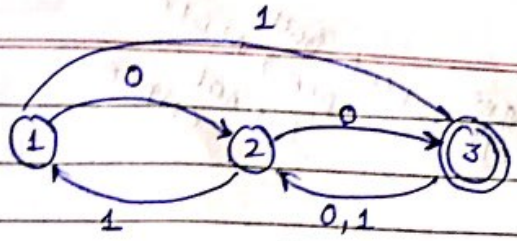
shift T on either side

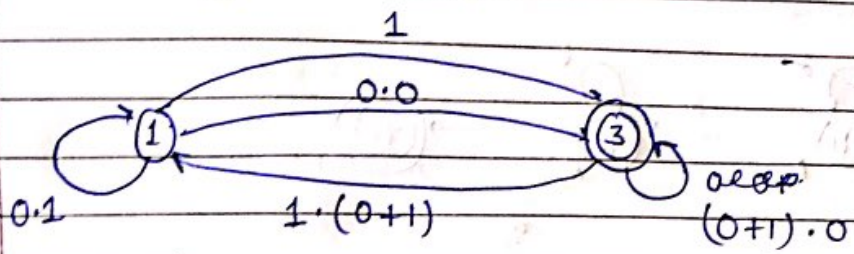
we are shifting T on 1



~~$(R+SU^*T)^*(S)(U^*)$~~

$(R+SU^*T)^*S\,U^*$



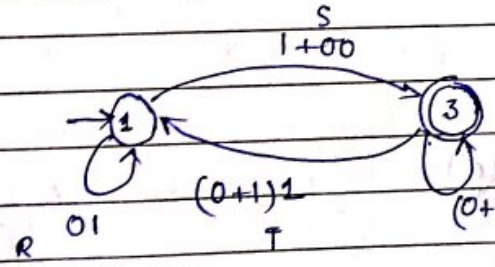


~~Q4~~ ~~$(0\cdot1)^*(1+0\cdot0+1(0+1)(0+1))((0+1)\cdot0)^*$~~ $(0+1)1$

now many paths are there? (direct or indirect)

	Direct	using intermediate node ②		
11	~~∅~~	0·1	1→2→1	∅+01
13	1	00	1→2→3	1+00
31	~~∅~~	(0+1)1	3→2→1	(0+1)1
33	~~∅~~	(0+1)0	3→2→3	(0+1)0

~~∅ is null set ; not even includes null(λ)~~



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

$(R+SU^*T)^*(S)(U)^*$