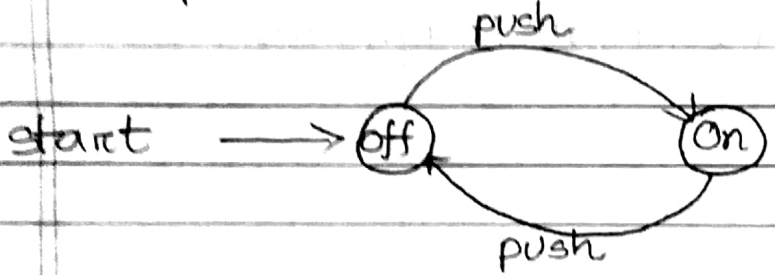
17/09/17
Sunday

Chapter: 1.1 & 1.5

Automata: A mechanism which at any time, stays at (in) one of a finite number of states.

States are used to remember necessary information.

Example: Electrical Switch



Representation:

States: Represented by circles

Arch/Edges: Represents external influence such as input.

Start State: Represented by an arrow "→"

Set of final States: Represented by double circles.

Example: Design a Finite Automata (FA) to recognize 'the'

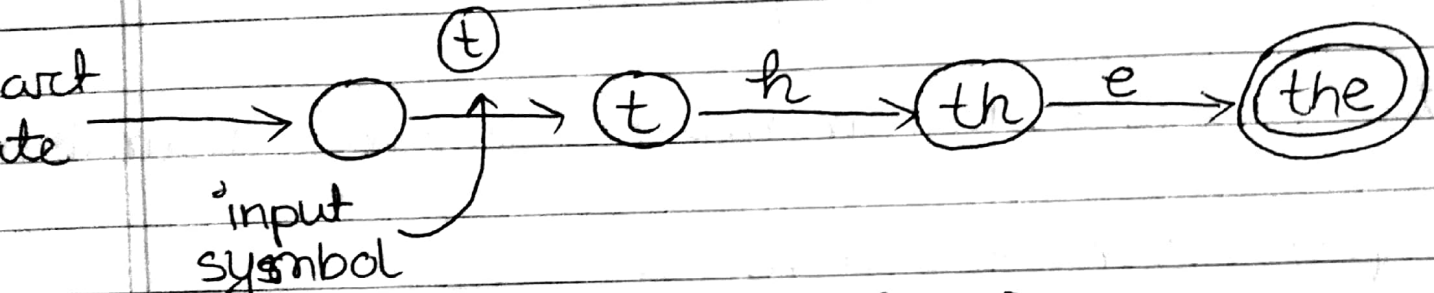


Figure: Transition Diagram

Important Definations:

Alphabet: A set (mathematical set) of symbols
Examples: Binary Alphabets {0, 1}

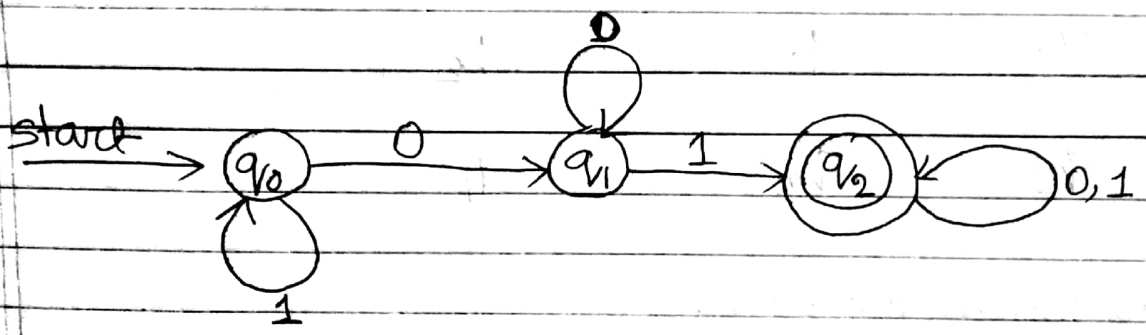
String: A list of symbols from an alphabet.
Examples: 0000, 1111, 0101
Any combination is considered

Language: A set of string from the same alphabet.
Examples: Equivalent Binary Code of even numbers.
{10, 110, 100}

Question Pattern: Given a language, design a DFA which recognizes all the strings that belong to that language.

Example: Design a DFA which accepts string of 0's and 1's having 01 somewhere in the string.

$L = \{w \mid w \text{ is in } \underline{x01y} \text{ form where } x, y \in \{0, 1, \emptyset\}\}$



19/09/17
Tuesday

Problem $L = \{w | w$ is of the form $x01y$ where $x, y \in \{0, 1, \emptyset\}\}$

Transition table:

States	0	1
$\rightarrow$	q_0	q_1	q_0
	q_1	q_1	q_2
*	q_2	q_2	q_1

Transition function

δ (current state, current i/p states) = next state

$\delta(q_0, 0) = q_1$ $\quad$ $\delta(q_1, 1) = q_2$

Extendend Transition function

$\hat{\delta}$ (Initial or current state, i/p string) = Final State

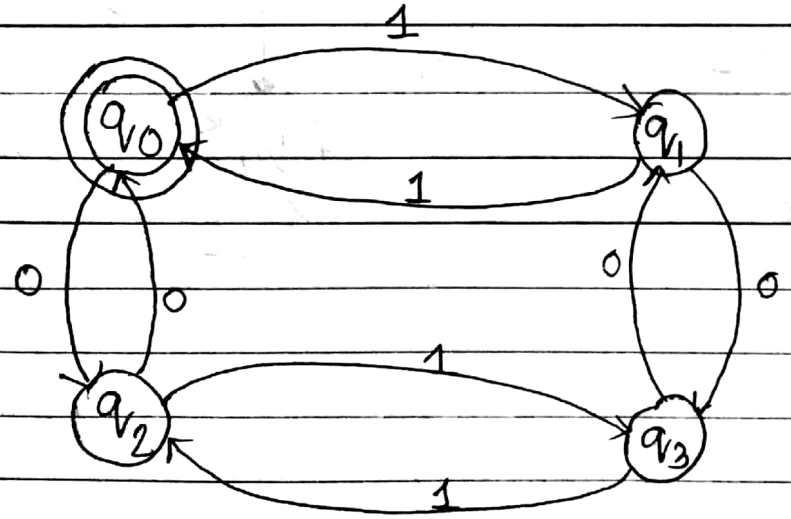
$\hat{\delta}(q_0, 0010) = \delta(\hat{\delta}(q_0, 001), 0)$

last element

Problem 2:

$L = \{ w \mid w$ has both even # of 0's and even # of 1's $\}$

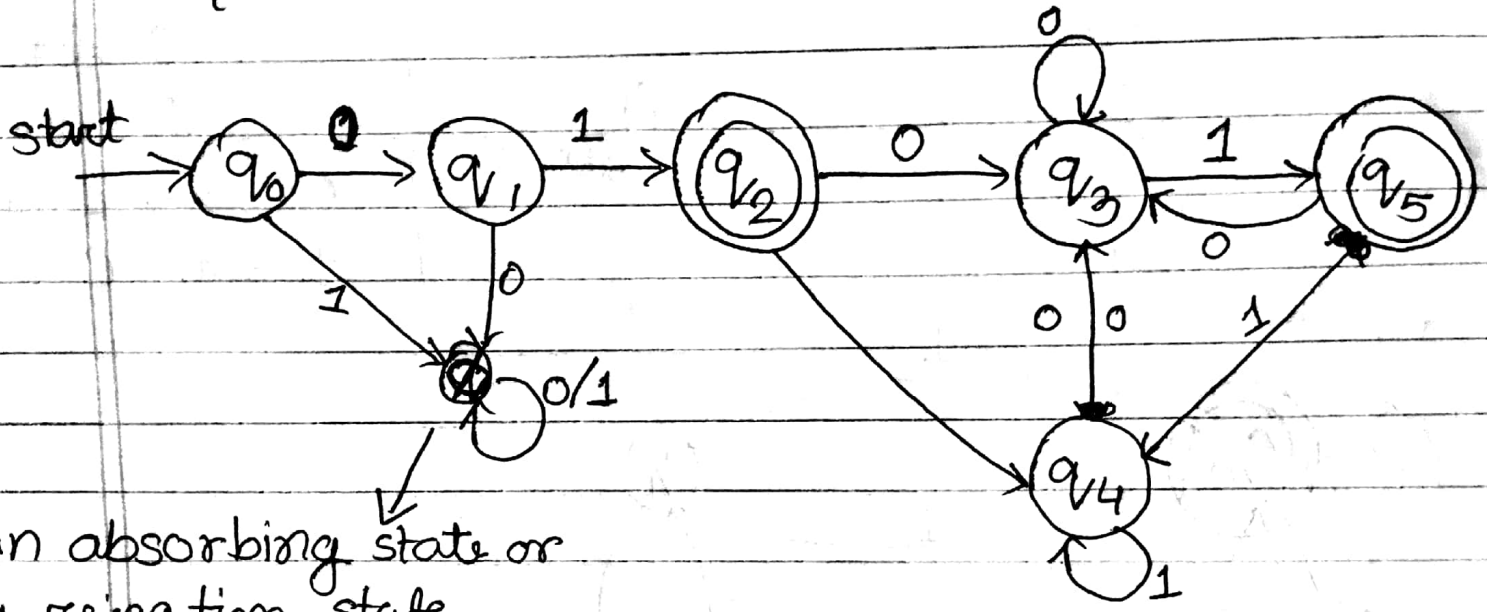
4 States:
- 0 ~~odd~~ even
- 1 even
- 0 even 1 odd
- 1 even 0 odd



Problem 3

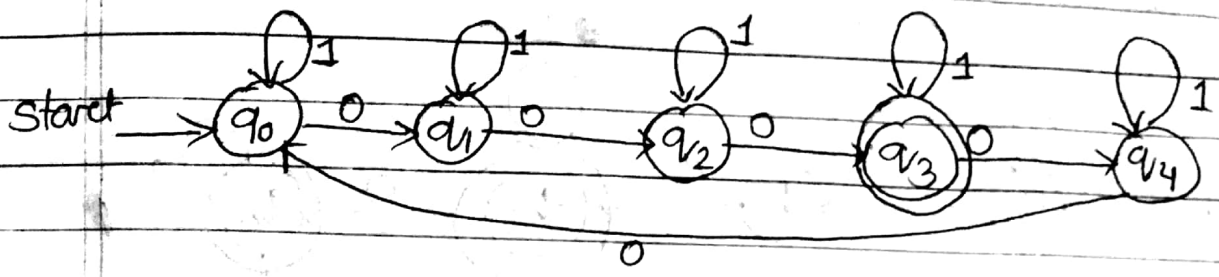
Alphabet, $\Sigma(0, 1)$

$L = \{ w | w$ starts with 01 and ends with 01 $\}$



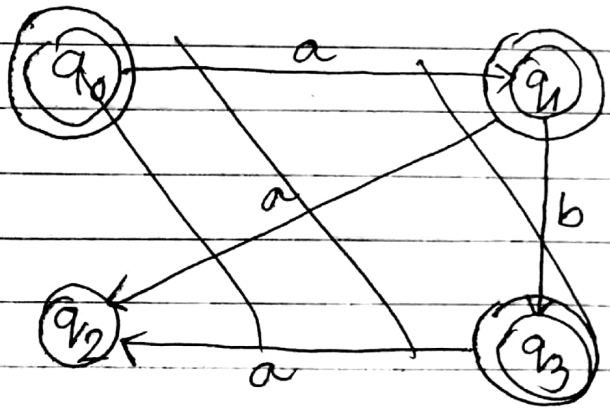
An absorbing state or a rejection state = A dead State

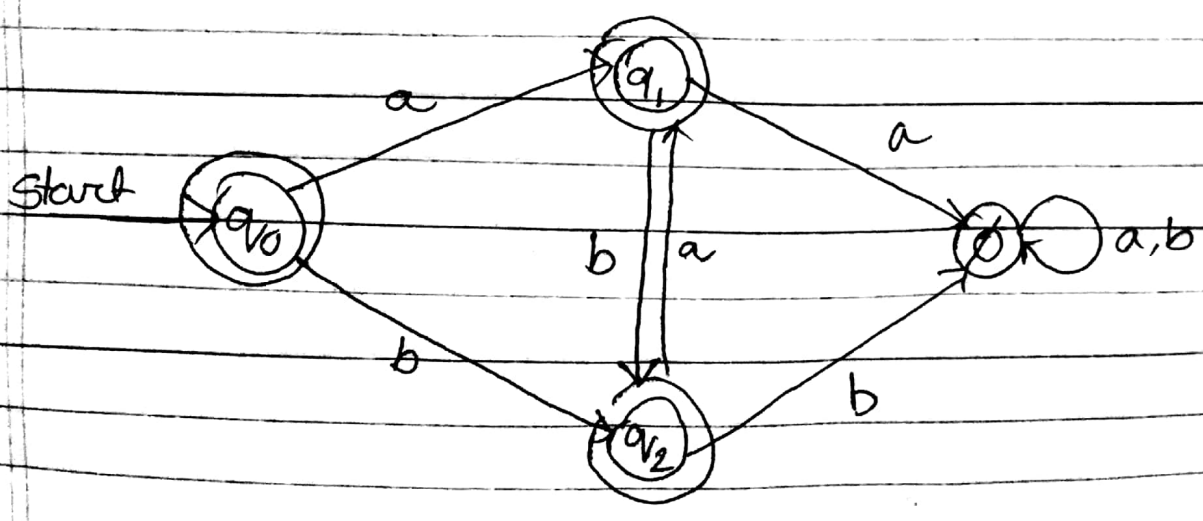
Problem4: $w|w$ has n occurences of 0's where $n \bmod 5$ is 3.



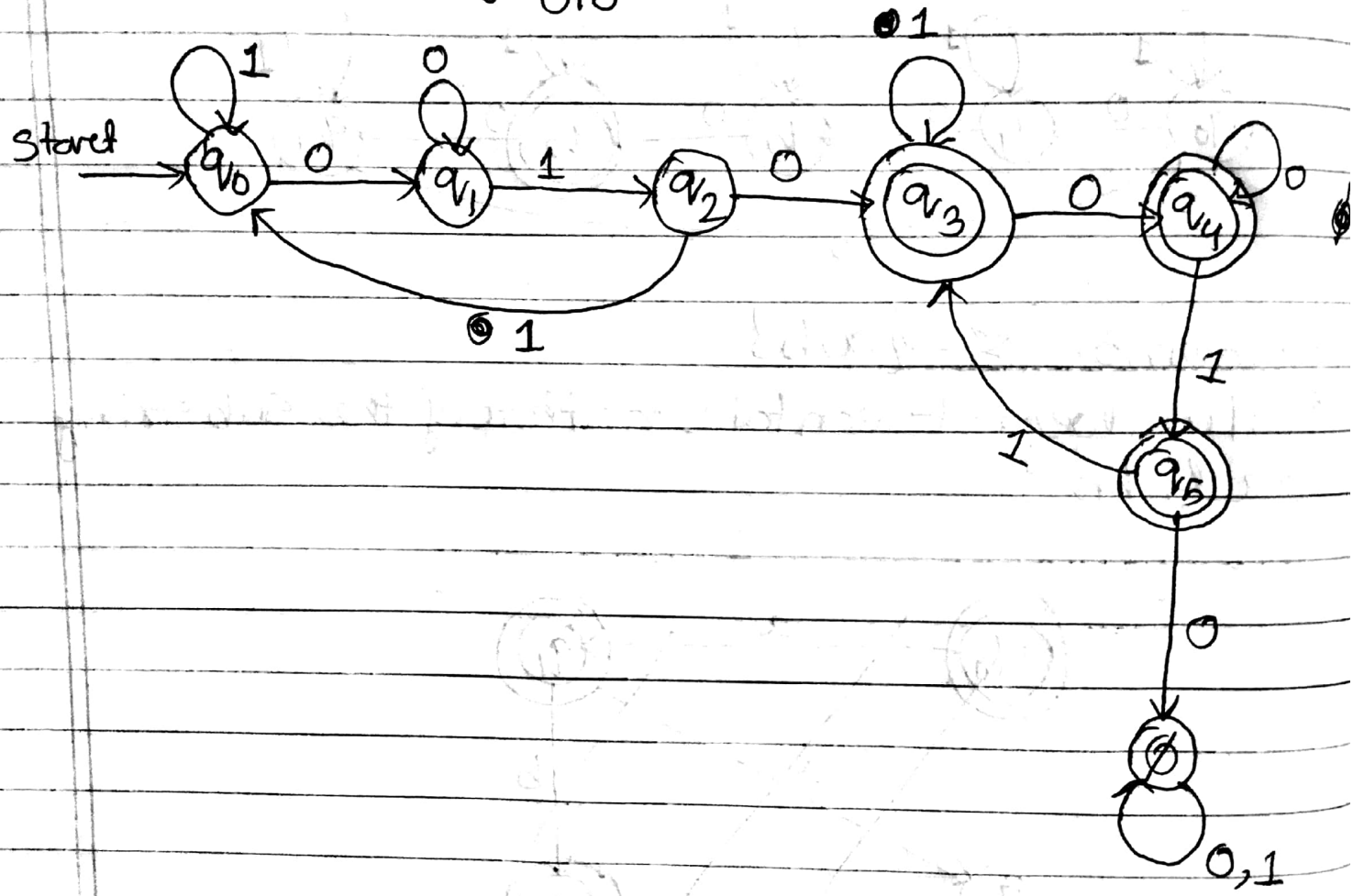
Problem5: $\Sigma = \{a, b\}$

$\{w|w$ does not contain neither of the sub string aa & bb $\}$





Problem 6: $L = \{w | w$ has exactly one occurrence of the substring $010\}$



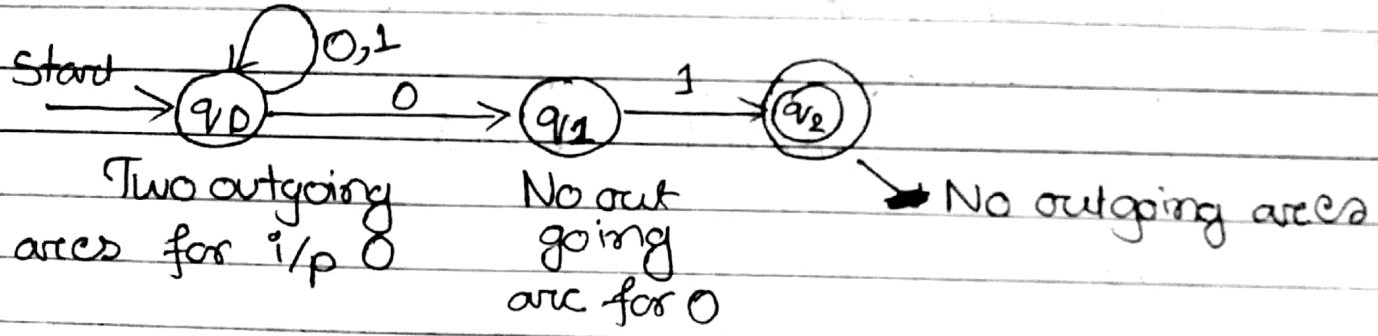
24/9/2017

NFA (Non deterministic FA)

NFA has the ability to be in several states at once.

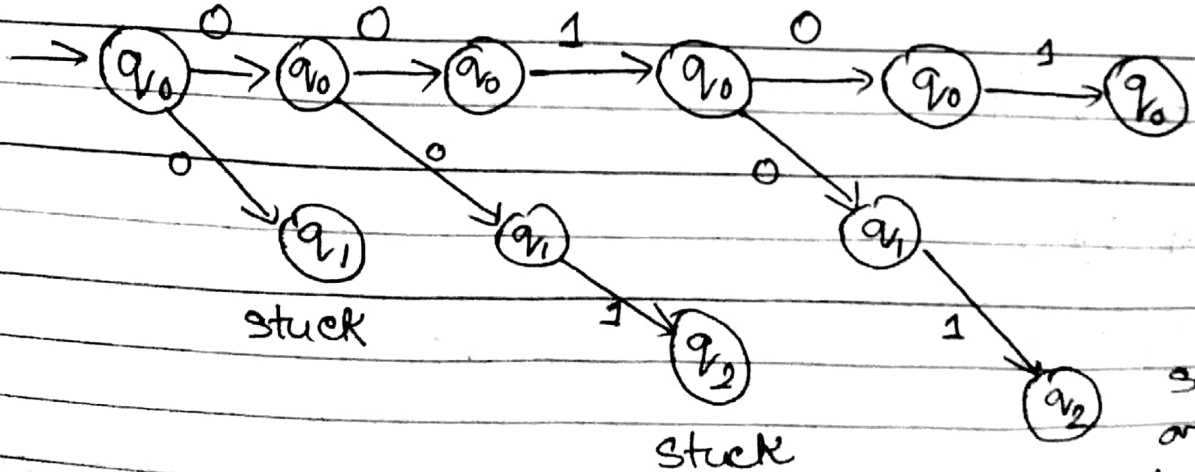
Transition function: δ(current state, i/p letter)
= A set of states

* Example: Design a NFA that accepts all and only strings of 0's and 1's that end in 01



Stimulation for NFA: Think like parallel execution

00101



Equivalence of NFA and DFA:

Can define same language

Smallest NFA - n states

Smallest DFA = 2^n states

Subset Construction:

Draw the equivalent DFA for problem 1.

optional labelling

	States \ I/P	0	1
A	$\{\emptyset\}$	$\{q_0, q_1\}$	$\{q_0\}$
B	$\{q_0\}$		
C	$\{q_1\}$		
D	$\{q_2\}$		
E	$\{q_0, q_1\}$	$\{q_0, q_1\}$	
F	$\{q_0, q_2\}$		
G	$\{q_1, q_2\}$		
H	$\{q_0, q_1, q_2\}$		

$0 \rightarrow \{q_0, q_1\} \cup \{\emptyset\} \rightarrow \{q_0, q_1\}$

Lazy Up Evaluation:

To avoid exponential cost, we can consider only the reachable states.

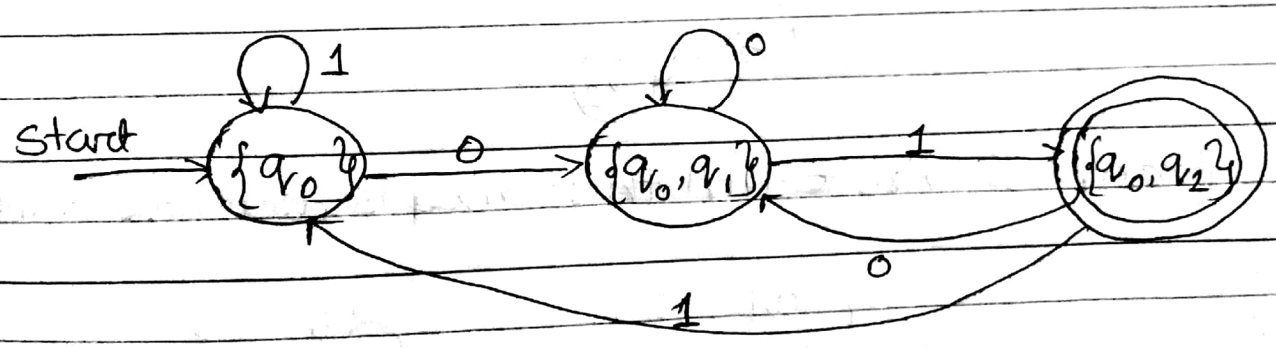
States \ I/P	0	1
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

new state

new state

no new state, so we are done.

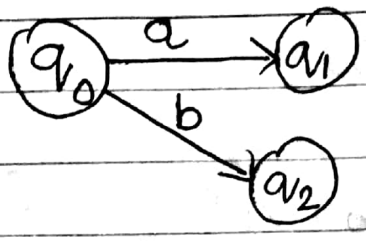
* Transition Diagram of equivalent DFA:



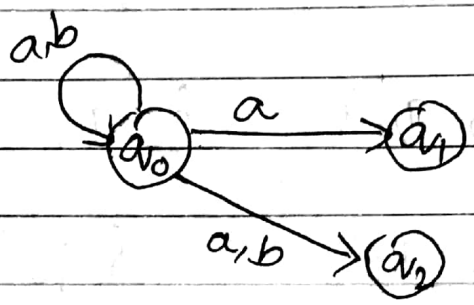
ε - Epsilon Stimulation

↓ empty string

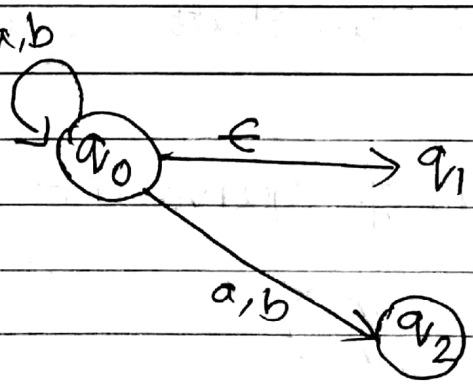
DFA



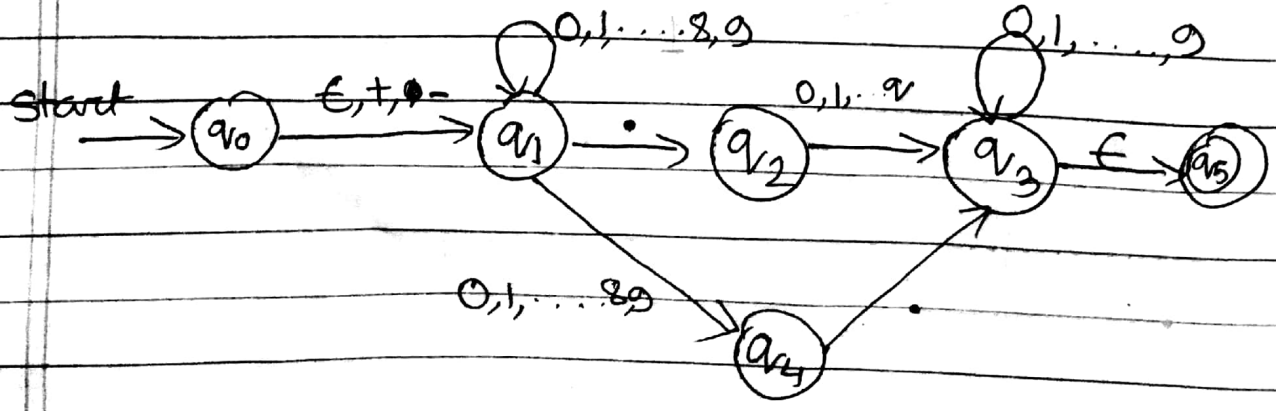
NFA



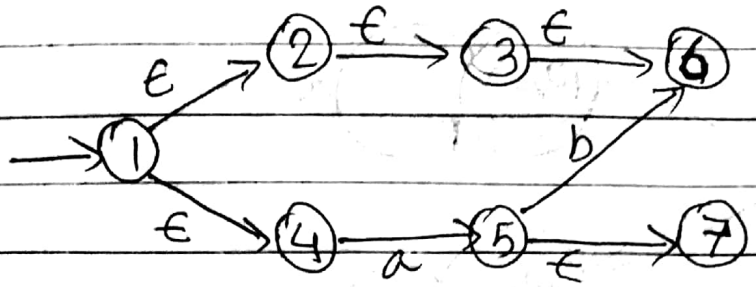
ε-NFA



Example: An ε-NFA that accepts decimal numbers



Epsilon Closures:



ε close (1) = {1, 2, 3, 6, 4}

ε close (4) = {4}

ε close (5) = {5, 7}

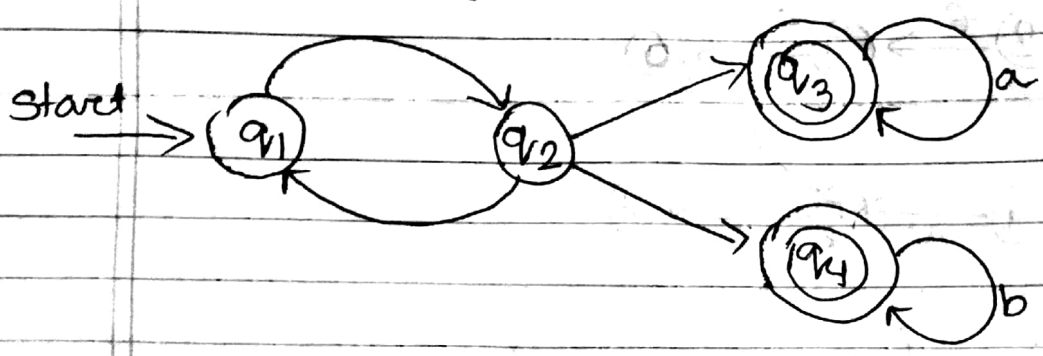
Eliminating ε transitions:

Give any ε-NFA E, we can create the equivalent DFA D, incorporating the mechanism of ε-closure in subset construction.

Example

26-09-17
Tuesday

Conversion of an ∈ NFA to equivalent DFA:

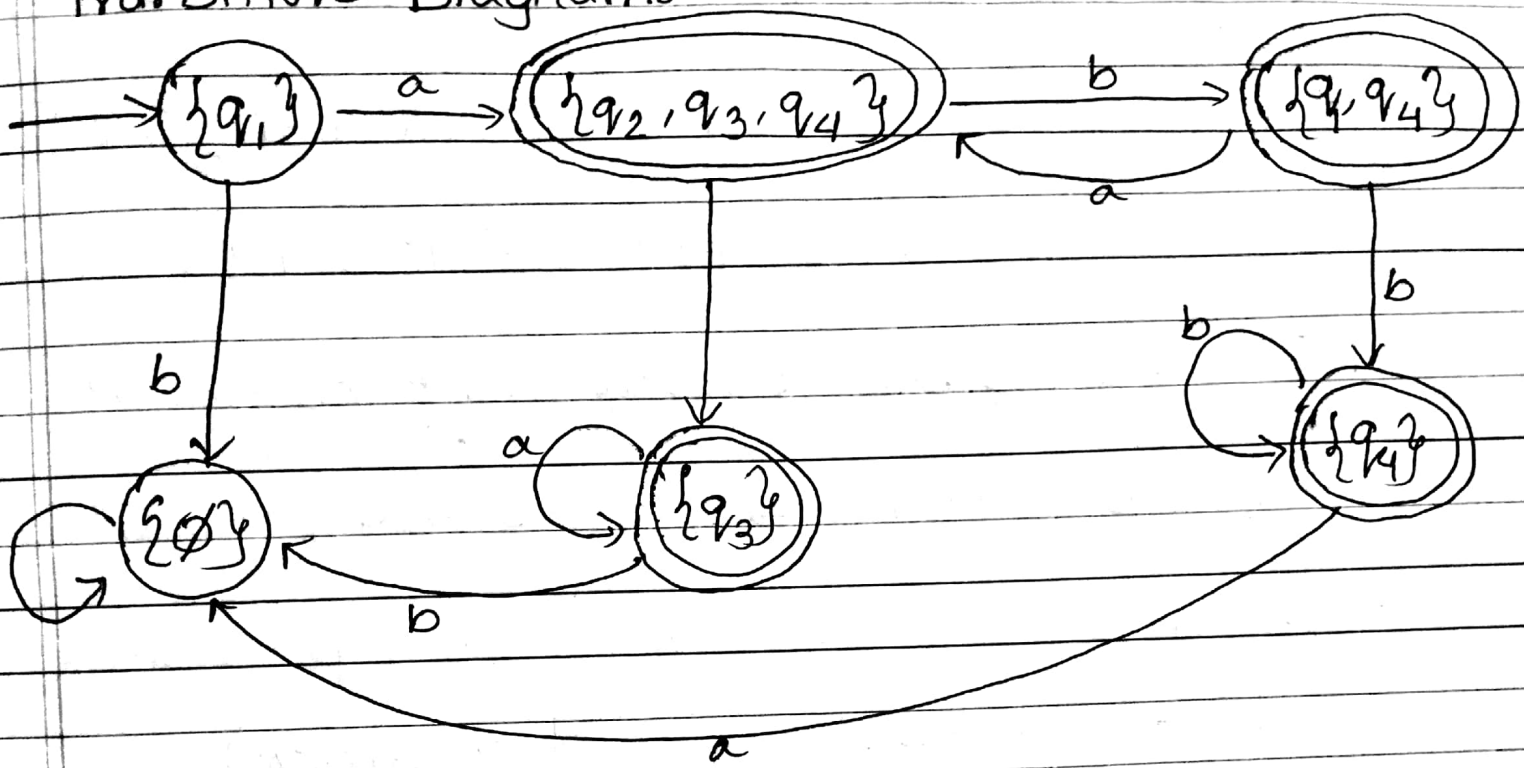


∴ close (q_1) = $\{q_1\}$
∈ close (q_2) = $\{q_2, q_3, q_4\}$
∈ close (q_3) = $\{q_3\}$
∈ close (q_4) = $\{q_4\}$
∈ close (ϕ) = $\{\phi\}$

Using Lazy Evaluation:

I/P / States	a	b
→ $\{q_1\}$	$\{q_2, q_3, q_4\}$	$\{\phi\}$
$\{\phi\}$ ←	$\{\phi\}$	$\{\phi\}$
* $\{q_2, q_3, q_4\}$ ←	$\{\phi\} \cup \{q_3\} \cup \{\phi\} = \{q_3\}$	$\{q_2\} \cup \{\phi\} \cup \{q_4\} = \{q_2, q_4\}$
$\{q_3\}$	$\{q_3\}$	$\{\phi\}$
$\{q_1, q_4\}$ ←	$\{q_2, q_3, q_4\} \cup \{q_4\} = \{q_2, q_3, q_4\}$	$\{\phi\} \cup \{q_4\} = \{q_4\}$
* $\{q_4\}$ ←	$\{\phi\}$	$\{q_4\}$

Transition Diagram:



For two strings

L = 11 & M = 00

then L^* = ∈, 11, 1111, 11 11 11, ...

L·M = 1100 & M·L = 0011

L+M = 11, 00

Regular Expressions:

* Book: Chapter 3: 3.1 ~ 3.2 (excluding 3.2.1)
  → Hopcraft
  Chapter 1: Example 1.27, 1.35, 1.36

RE is the algebraic description of the regular language.
Like algebra, RE has operators and operand.

Operators of RE in descending order of precedence:

1. * (Kleene closure or star operator)
2. . (Dot or concentation operator)
3. + (Union or plus or OR operator)

Problems (more than 1 RE can be designed)

1) Write a regular expression for the set of strings that constists of alternating 0's and 1's.

sol 1

$$(01)^* + (10)^* + 1(01)^* + 0(10)^*$$

sol 2

$$(\epsilon + 1)(01)^*(\epsilon + 0)$$

2) W is the set of all strings with 010 as a substring.

Sol: $(0+1)^* \cdot 010 \cdot (0+1)^*$

3) Sets of all strings which do not have 010 as a substring

Sol: $(1+00^*11)^* (\epsilon + 00^* + 00^*1)$

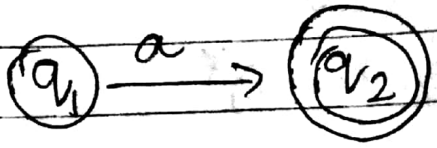
4) Set of strings which have ~~en~~ even number of 0's or even number of 1's

Sol: ~~ϵ~~ $(1^*01^*01^*)^* \cup (0^*10^*10^*)^*$

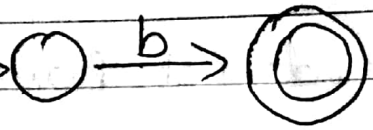
03/10/2017
Tuesday

Converting RE to ENFA.

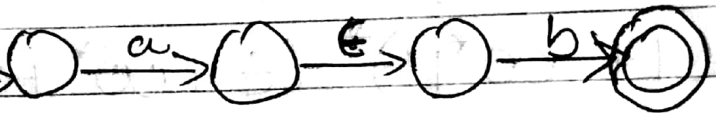
1. a =>



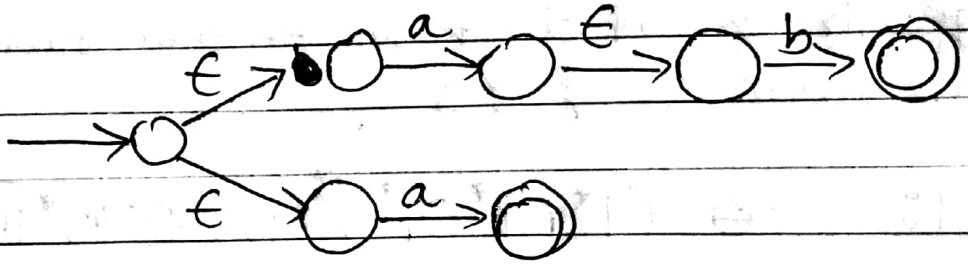
2. b =>



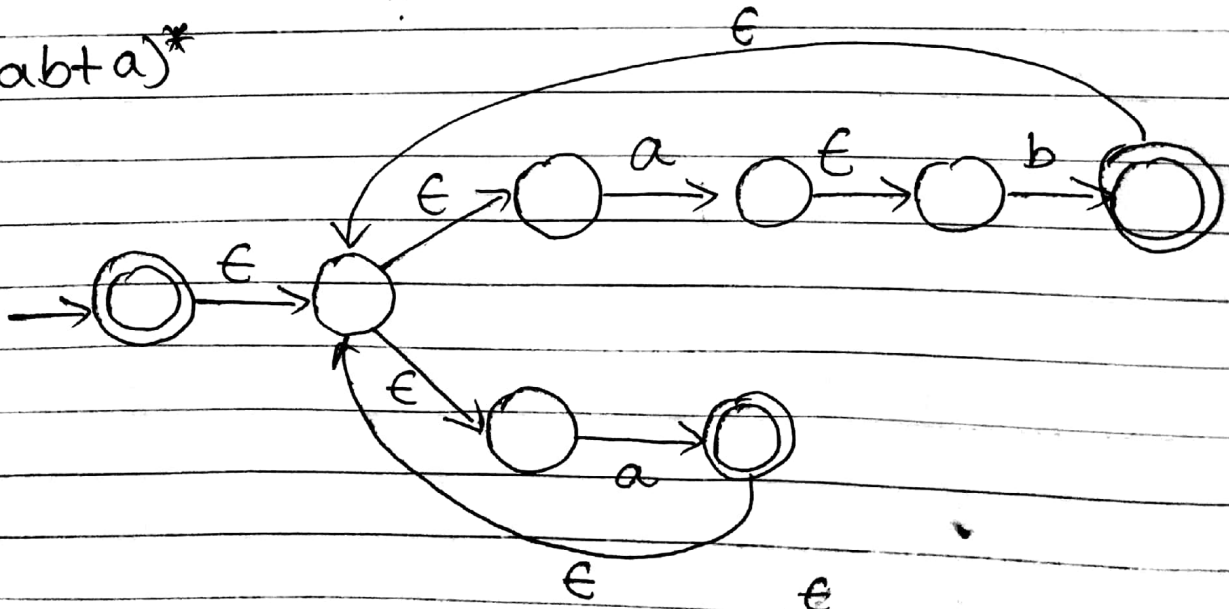
3. ab =>



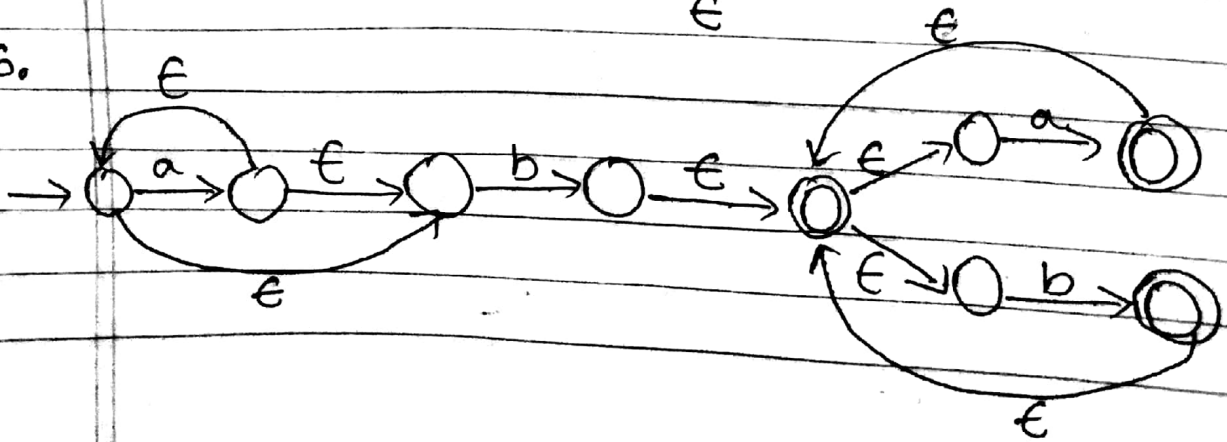
4. ab+a =>



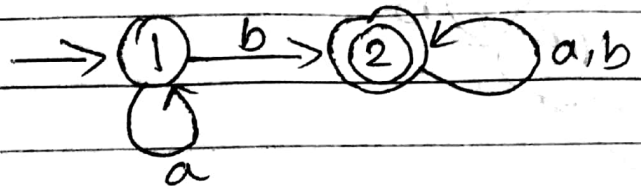
5. $(ab+a)^*$



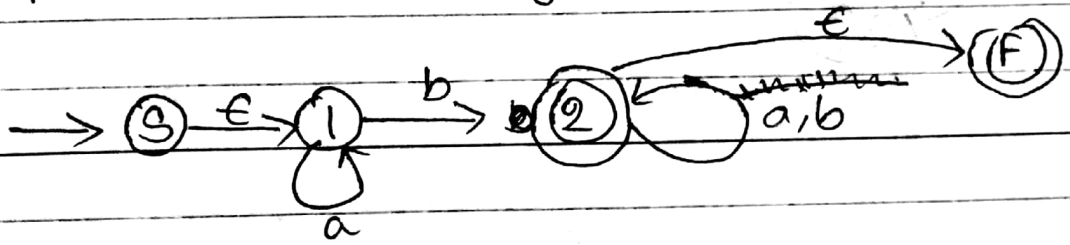
6.



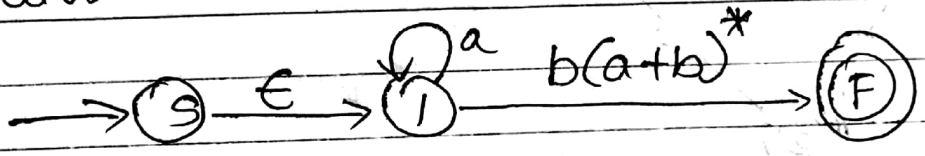
Convert the following DFA into ~~an~~ equivalent R.E:



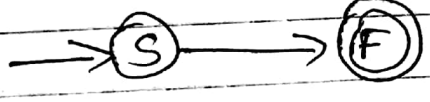
Step 1: Add a starting & final node



Step 2: Remove original nodes one at a time & add equivalent R.E.

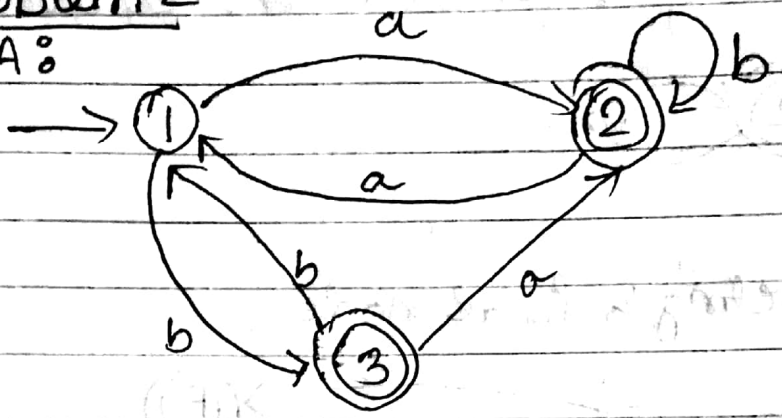


$a^*b(a+b)^*$

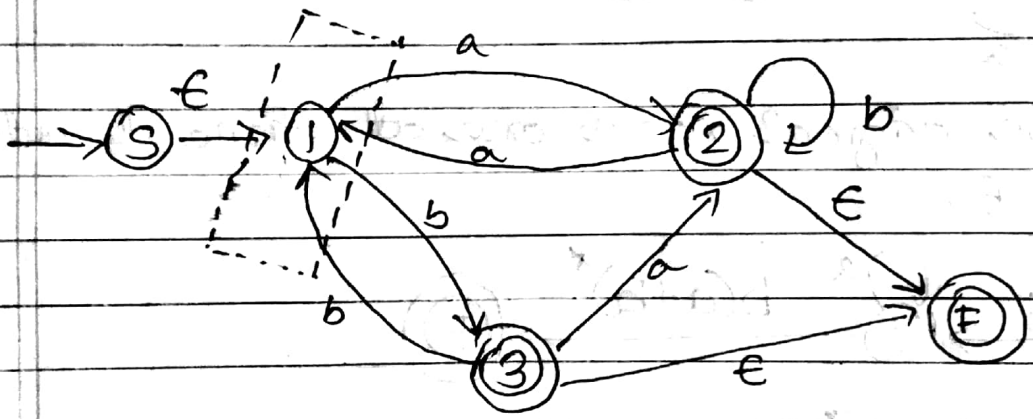


Problem 2

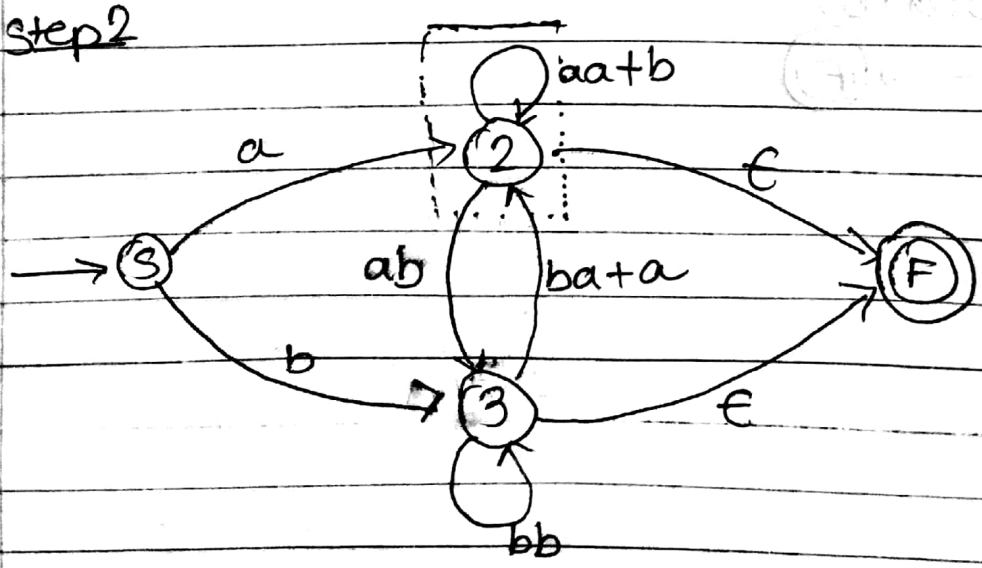
DFA:

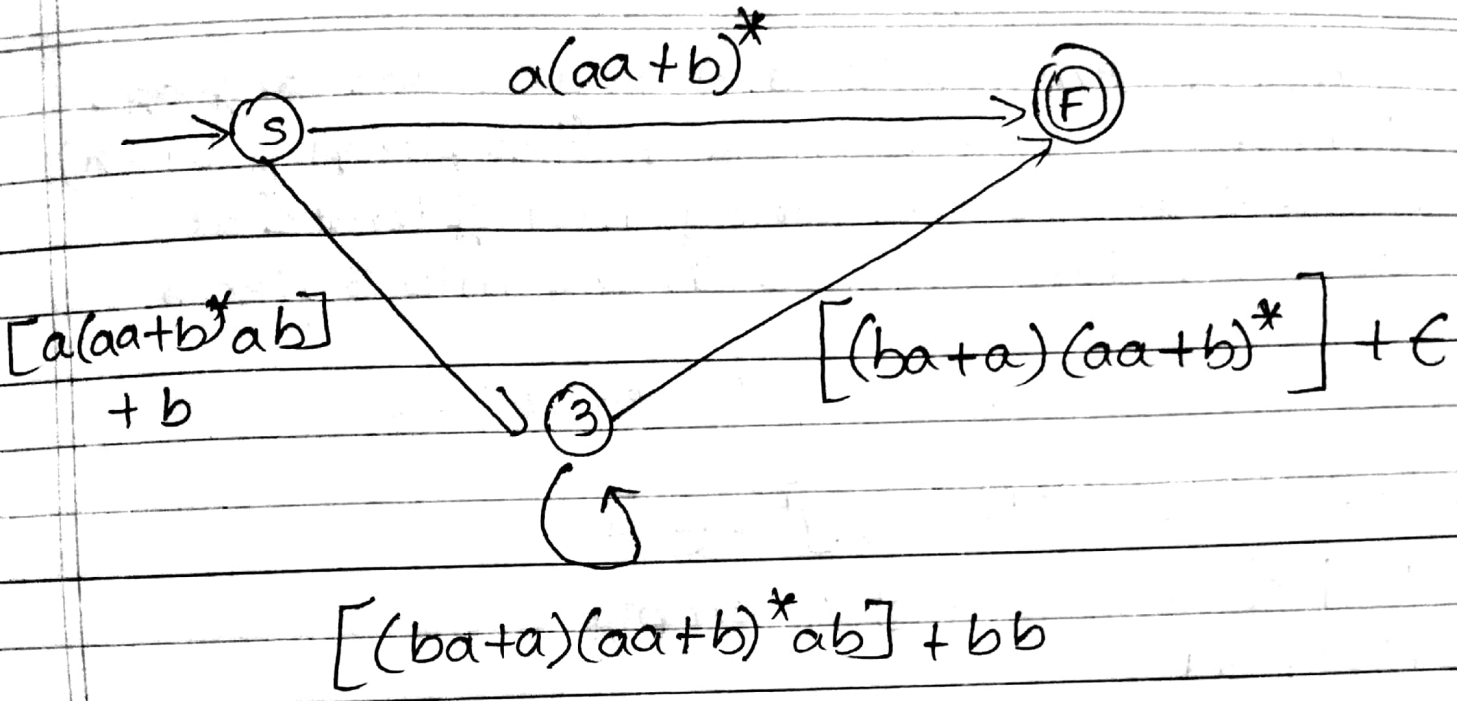


Step 1

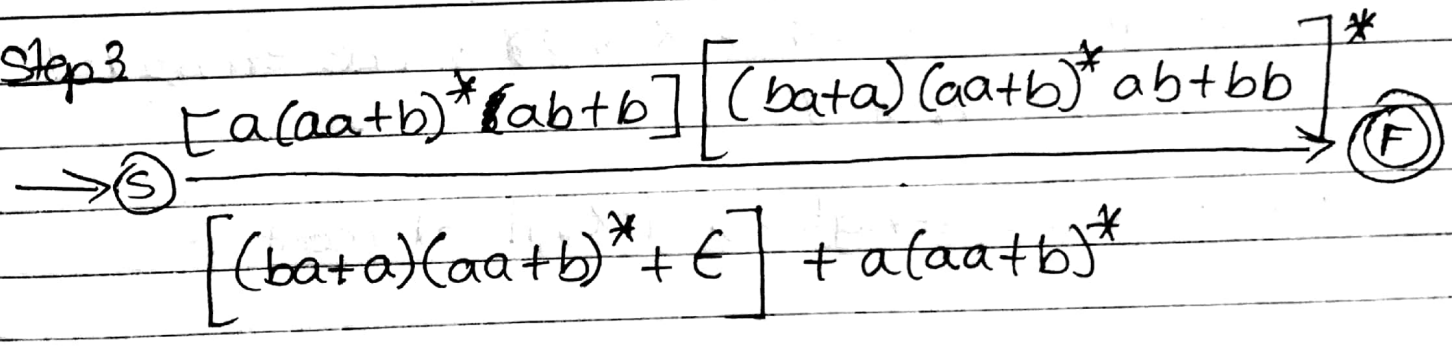


step2





Step 3



* No Fucking clue of what happened

8/10/2017
Sunday

Pumping Lemma: If A is a regular language, then there is a number p (the pumping length) where if S is any string in A of length at least p, then S can be divided into 3 pieces

$S = xyz$ with conditions

1) for each $i \geq 0$, $xy^i z \in A$
2) $|y| > 0$ ∵ $y \neq$ empty
3) $|xy| \leq p$

Condition 2: So either x or z can be empty but not y

Condition 3: Length of $|x,y|$ at most p

Example 1.38: $B = \{0^n 1^n \mid n \geq 0\}$ Prove where B is a regular language or not, using pumping lemma.

⇒ If B is regular then it should satisfy all conditions of pumping lemma

1) Let p be the pumping length
2) Let $S = 0^p 1^p$ because $S \in B$

3) According to pumping lemma

$S = xyz$

for any $i \geq 0$, xy^iz is in B.

4) Now consider 3 different sets of y.

(a) y consists of only 0's. Then xyyz have more 0's than 1's

(b) y consists of only 1's. Then again xyyz has more 1's than 0's

(c) y consists of both 0's and 1's. Now number of 0's of 1's can be same but they will ~~not~~ be out of order.

So contradiction is unavoidable, if we consider B a regular language. Therefore B is not regular.

* Questions come in exam from the examples of Book

10/10/2017
Tuesday

Pumping Lemma:

Sipser: Chapter 1

Example: 1.38, 1.39, 1.40, 1.41, 1.42

1.41 Prove where $D = \{1^{n^2} \mid n \geq 0\}$ is a regular language or not using pumping lemma.

Assume, D is regular & P is pumping length. then ~~$s = i$~~ $s = 1^{p^2}$. $\therefore s = xyz$ & for $i \geq 0$, $xy^iz \in D$

Let $m = n^2$ $\therefore (n+1)^2 - n^2 = n^2 + 2n + 1 - n^2 = 2n + 1$

$= 2\sqrt{m} + 1$

Length of member of D are

0 1 4 9 16 25 36 49

1 3 5 7 9 11 13

and different of length of two consecutive strings

$$|xy^{i+1}z| \sim |xy^iz|$$

$$y$$

Now if D is regular then both $|xy^iz|$ & $|xy^{i+1}z|$ are perfect squares. But if

$$|y| < 2\sqrt{|xy^iz|} + 1$$

then both cannot be perfect squares.

Now, $|y| \leq s = p^2 = \sqrt{p^4}$

$< 2\sqrt{p^4} + 1$

$\leq 2\sqrt{xy^i z} + 1$

so $|y| < 2\sqrt{xy^i z} + 1$

So we have reached a contradiction. Therefore D cannot be a regular language.

4] Context Free Grammar (CFG):

→ We can change a non regular language using CFG

→ L = { a palindrome & $\Sigma = \{0, 1\}$ }

Then CFG:

Start variable 1. P → 0 → Non variable / Terminal Symbols

2. P → 1

P is a variable / Non terminal 3. P → ∈ → 1st 3 rules form the basis portion

4. P → 0P0 → It is a production Rule

5. P → 1P1 → Last 2 rules form the inductive portion

LHS — Only variables are allowed in the LHS

RHS — Both variables & non variables are allowed in RHS

Using CFG find out if string 00100 is a palindrome.

$\# P$

$\Rightarrow 0P0 \rightarrow (4)$

$\Rightarrow 0\ 0P0\ 0 \rightarrow (4)$

$\Rightarrow 00100 \rightarrow (2)$

$\therefore$ String 00100 is a palindrome

15/10/2017
Sunday

[1] A CFG of palindrome whose length in even/odd.

Even	odd
1) $P \to \epsilon$	1) $P \to 0$
2) $P \to 0P0$	2) $P \to 1$
3) $P \to 1P1$	3) $P \to 0P0$
	4) $P \to 1P1$

[2] Design a CGF representing arithmetic expressions (E) with operators + & *, containing identifiers (I) & expressions. & with $\Sigma = \{0, 1, a, b, +, *\}, (,)\}$

Production Rules

Production rules for an expression:

1. $E \to (E)$ → Starting variable
2. $E \to E+E$
3. $E \to E*E$
4. $E \to I$ } Goes from expression to identifier

Production rules for an identifier:

5. $I \to a$ } base case
6. $I \to b$ } base case
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

} Inductive case

Production Rule 5–10, can be written compactly as follows:

$$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

↳ or

Derivations: Derivations are used to find out if a string belongs to the language or not.

→ Find if a string a*(a+b00) belongs to the language E.

{Leftmost Derivations}

$\Rightarrow E * E$
$\Rightarrow I * E$
$\Rightarrow a * E$
$\Rightarrow a * (E)$
$\Rightarrow a * (E + E)$
$\Rightarrow a * (I + E)$
$\Rightarrow a * (a + E)$
$\Rightarrow a * (a + I)$
$\Rightarrow a * (a + I0)$
$\Rightarrow a * (a + I00)$
$\Rightarrow a * (a + b00)$

∴ the string a*(a+b00) belongs to the language.

How to perform derivations:

→ Start with the starting variable

→ Replace one variable at a single step with its production rule.

Two kinds of derivation:

1) Leftmost Derivations: When at each step, the left most variable is replaced with one of its production rules.

2) Rightmost Derivations: When at each step, the right most variable is replaced with one of its production rules.