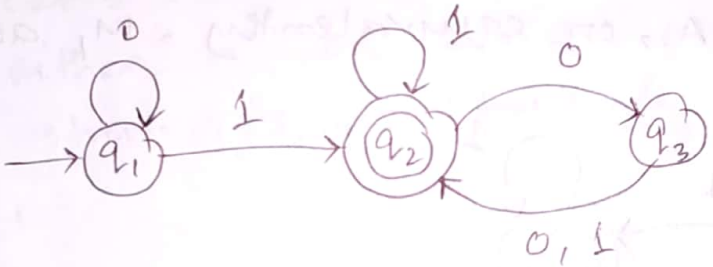
ToC

Formal Def^n of Finite Automata

A finite automation is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q : Finite set of states
2. Σ : Finite set of alphabets
3. δ : $Q \times \Sigma$ - transition table
4. q_0 : $q_0 \in Q$; the starting state
5. F : $F \subseteq Q$; set of accepted states

Ex 1.6



M_1

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. $\delta \Rightarrow$

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4: q_1 is the start state

5: $F = \{q_2\}$

Nayeem Mahmood / 6AM / C161026

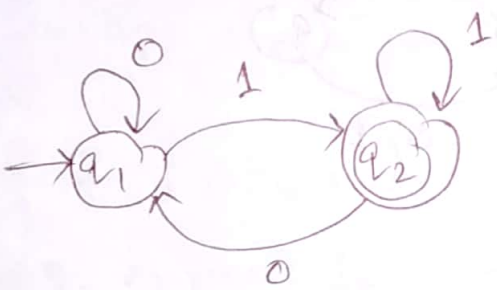
* If "A" be a set of all strings the a machine "M" accepts, we say that A is the language of machine M and write $L(M) = A$. We say that M recognizes A or M accepts A.

* In the preceding example,

let $A = \{w \mid w$ contains at least one 1 and an even number of 0s follows the last 1$\}$

Then $L(M_1) = A$, or equivalently, M_1 accepts A.

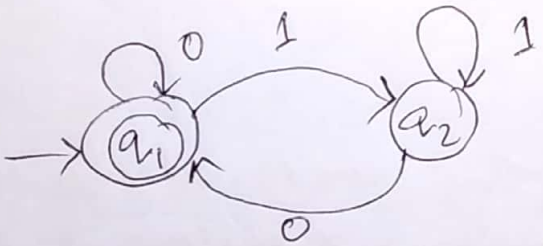
Ex



M_2

$L(M_2) = \{w \mid w$ ends in a 1$\}$

Ex

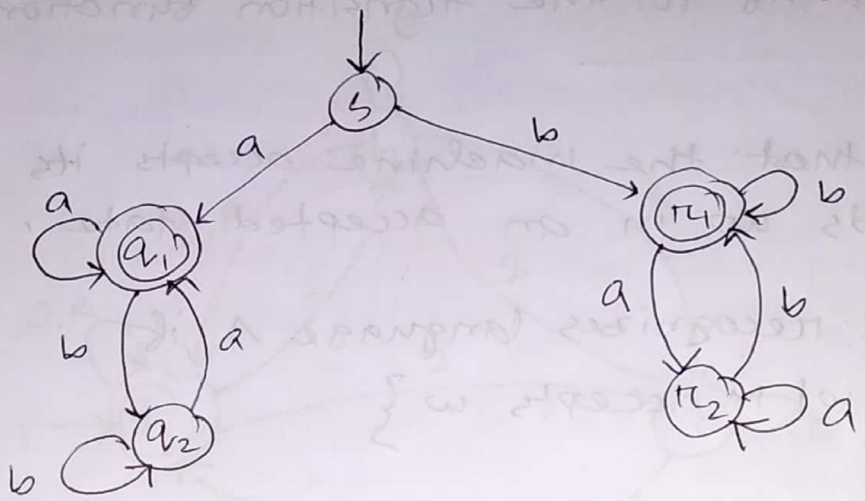


M_3

$L(M_3) = \{w \mid w$ is an empty string ε or ends in a 0$\}$

Nayeem Mahmood / 6AM / C161026

Ex 1.12



M_4

M_4 accepts all strings that start and end with either a or b. In other words, M_4 accepts strings that start & end with the same symbol.

Formal Def^n of computation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automation and let $w = w_1 w_2 \cdots w_n$ be a string where each w_i is a member of the alphabet Σ. Then M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with 3 conditions:

(1) $r_0 = q_0$

It says that the machine starts in the starting state (q_0).

(2) $\delta(r_i, w_{i+1}) = r_{i+1}$; for $i = 0$ to $n-1$

It says that the machine goes from state to

state according to the transition function.

(3) $r_n \in F$

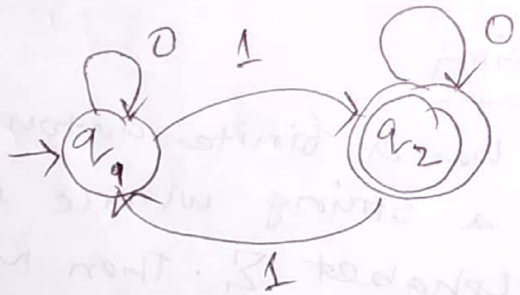
It says that the machine accepts its input if it ends up in an accepted state.

We say, M recognizes language A if

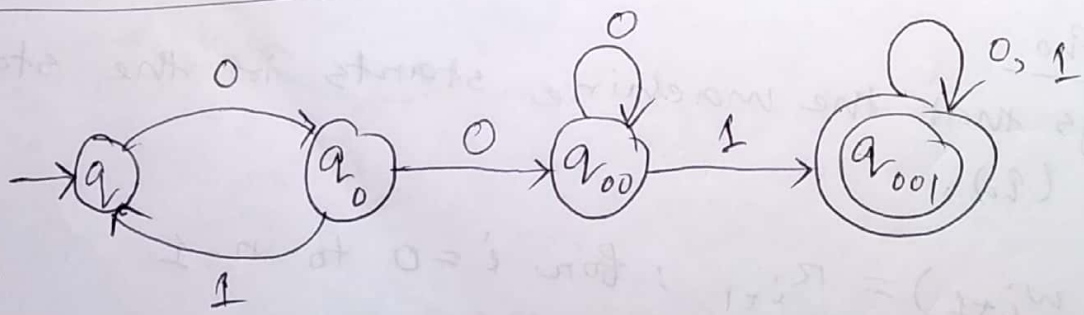
$$A = \{w \mid M \text{ accepts } w\}$$

* A language is called a Regular Language if some finite automation recognizes it.

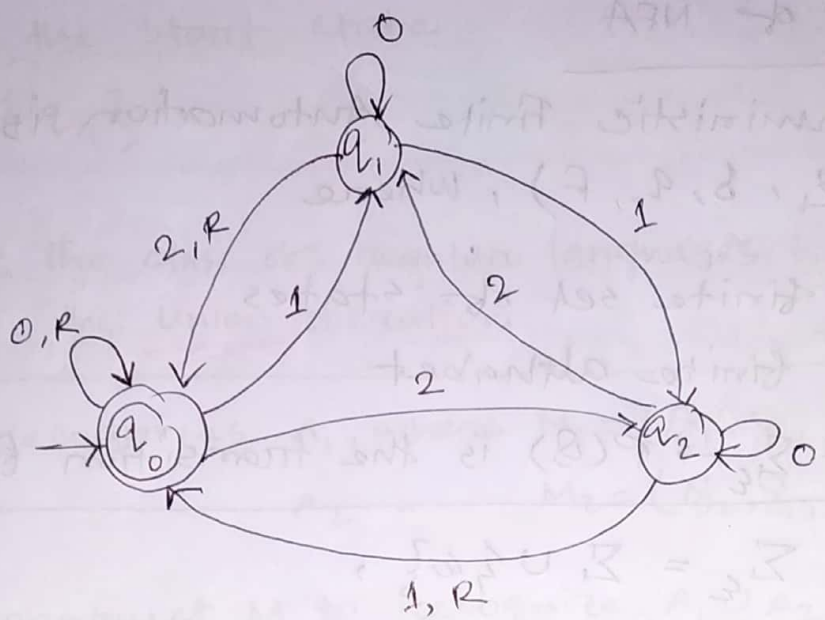
* Machine that accepts all strings with an odd number of 1s.



* Machine that accepts all strings that contain the string 001 as a substring



Ex



MS

MS keeps a running count of the sum of the numerical input symbols it reads, modulo 3. Every times it receives R (Reset) symbol, it resets the count to 0. It accepts if the sum is 0 modulo 3, or in other words, if the sum is a multiple of 3.

Formal Def^n of NFA

A Non-deterministic Finite Automation is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

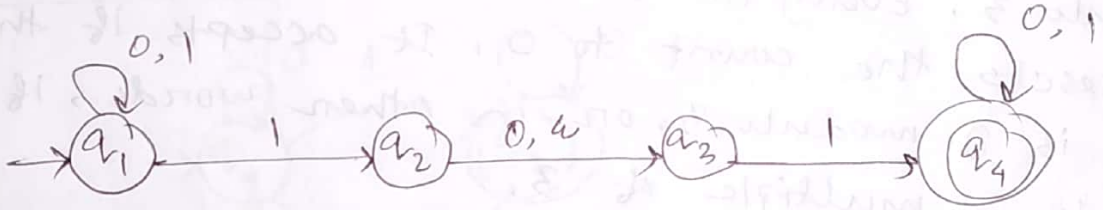
1. Q is a finite set of states
2. Σ is a finite alphabet
3. $\delta : Q \times \Sigma_\varepsilon \to P(Q)$ is the transition function

$$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\},$$

$P(Q)$ = Power set of Q

4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is set of accepted states.

Ex



N_1

1. $Q = \{q_1, q_2, q_3, q_4\}$
2. $\Sigma = \{0, 1\}$
3. δ is

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

4. q_1 is the start state

5. $F = \{q_4\}$

Theorem : The class of regular languages is closed under the union operation

Let M_1 recognizes A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and

M_2 " A_2 " $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

We'll construct M to recognize $A_1 \cup A_2$ where

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

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
2. Σ, the alphabet, is the same as in M_1 and M_2.
3. δ, is defined as follows:

   For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

4. q_0 is the pair (q_1, q_2)
5. $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$

For intersection,

$$F = F_1 \times F_2$$

Every non-deterministic finite automation has an equivalent deterministic finite automation.

Let,
$N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing some language A. We construct DFA $M = (Q', \Sigma, \delta', q_0', F')$ recognizing A.

1. $Q' = P(Q)$
2. For $R \in Q'$ and $a \in \Sigma$,

   let,
   $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R\}$.
   If R is a state of M, it is also a state of N.

   so, $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$

   [The notation $\bigcup_{r \in R} \delta(r, a)$ means: the union of the sets $\delta(r, a)$ for each possible r in R]
3. $q_0' = \{q_0\}$
4. $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$

* A language is regular iff some NFA recognizes it.

* The class of regular languages is closed under the union operation

Let,

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 and

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2

constructing $N = (Q, \Sigma, \delta, q, F)$ to recognize $A_1 \cup A_2$

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$
2. The state q_0 is the start state of N.
3. The set of accept state $F = F_1 \cup F_2$
4. $$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

$\hookrightarrow q \in Q, a \in \Sigma_\varepsilon$

The class of regular languages is closed under the concatenation operation.

Let
$N_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ recongizes A_1 and
$N_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ recognizes A_2

Constructing $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state q_1 is the same as the start state of N_1.
3. The accept states F_2 are the same as the accept states of N_2.
4. Def^n of δ : $q \in Q$ and any $a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

The class of regular languages is closed under star operation.

Let $N_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ ~~recog~~ recognize A_1.
Constructing $N = (Q, \Sigma_1, \delta, q_0, F)$ to recognize A_1^*.

1. $Q = \{q_0\} \cup Q_1$
2. The state q_0 is the new start state
3. $F = \{q_0\} \cup F_1$
4. Def^n of δ : $q \in Q$ and any $a \in \Sigma_4$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

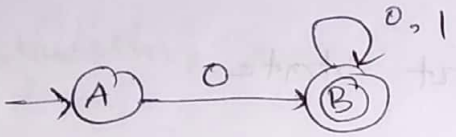
* Conversion of NFA to DFA

Ex 1 ['Sub-set Construction Method']

L = {Set of all strings over (0,1) that starts with '0'}

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

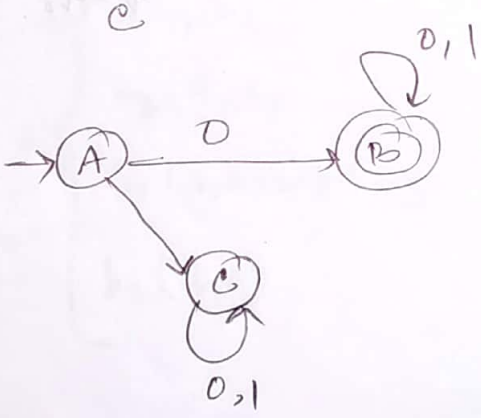
NFA



	0	1
A	B	∅
B	B	B

DFA

	0	1
A	B	C, where C is dead/ trap state
B	B	B
C	C	C

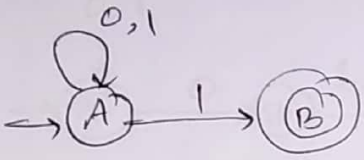


Ex 2

$L = \{$Set of all strings over $(0,1)$ that ends with '1'$\}$

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

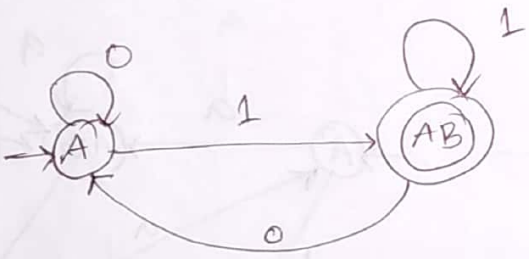
NfA



	0	1
A	{A}	{A, B}
→ B	∅	∅

DfA

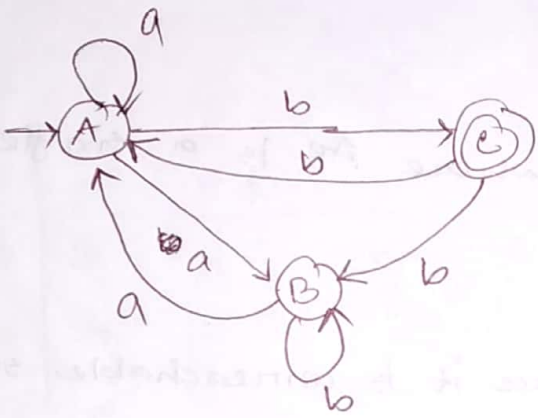
	0	1
A	{A}	{AB}, where AB is a single state
AB (AUB)	{A}	{AB}
~~B~~	[Not required because it is unreachable so far]	



Find the equivalent DFA for the NFA given by $M = [\{A, B, C\}, (a, b), \delta, A, \{C\}]$ where δ is given by

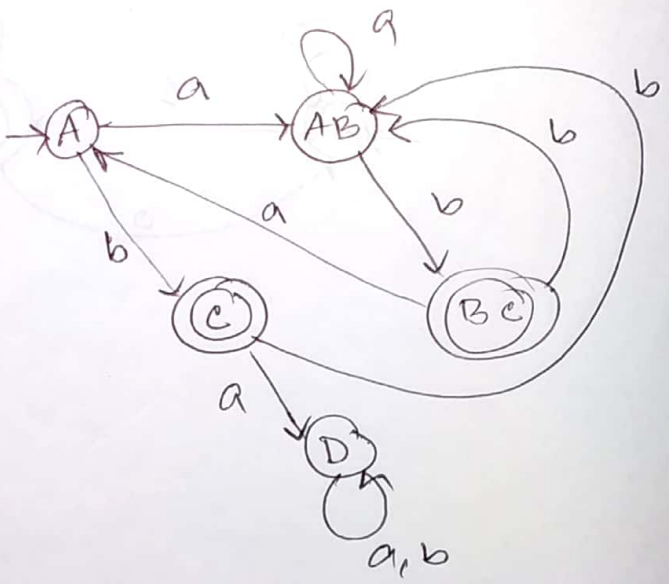
	a	b
→ A	A, B	C
B	A	B
(C)	-	A, B

NFA



DFA

	a	b
→ A	AB	C
AB	AB	BC
(BC)	A	AB
(C)	D	AB
	↓ Dead	
D	D	D



F = {BC, C}, because both of them hold C

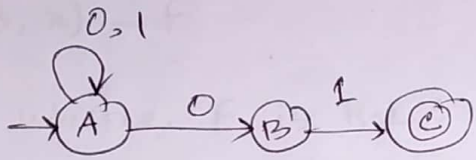
Nayeem Mahmood / 6AM / C161026

Ex 3

$L = \{$ set of all strings over $(0,1)$ that ends with '01' $\}$

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

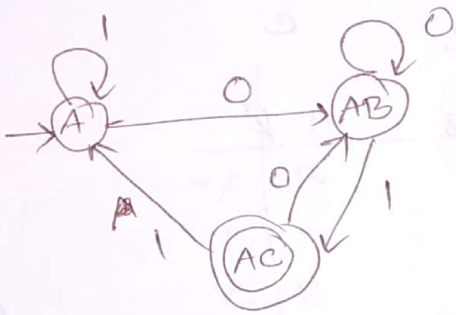
NFA



	0	1
→ A	A, B	A
B	∅	C
(C)	∅	∅

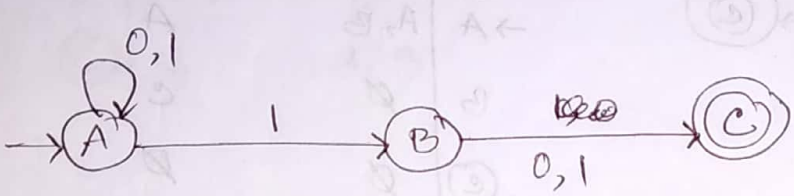
DFA

	0	1
→ A	AB	A
AB	AB	AC
~~C~~	~~D~~	D
D	D	D
(AC)	AB	A



Design an NFA for a language that accepts all strings over (0,1) in which the second last symbol is always '1'. Then convert it into DFA

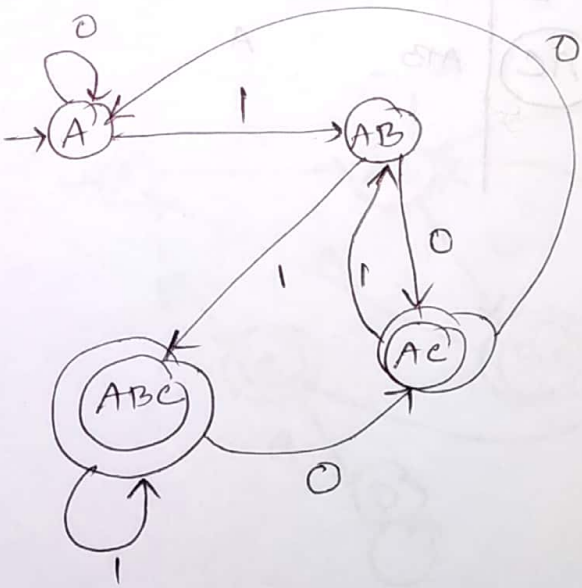
NFA



	0	1
→A	A	A, B
B	C	C
(C)	∅	∅

DFA

	0	1
→A	A	AB
AB	AC	ABC
(AC)	A	AB
(ABC)	AC	ABC



* Two states 'A' and 'B' are said to be equivalent if

$\delta(A, x) \to F$ and $\delta(B, x) \to F$

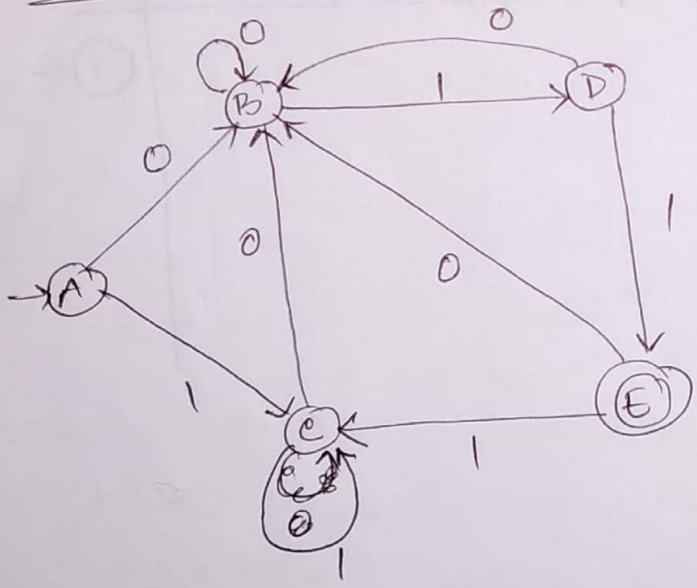
OR

$\delta(A, x) \not\to F$ OR $\delta(B, x) \not\to F$

where, F is the set of accept states,
x is any input string

* If $|x| = n$, then A and B are said to be n equivalent.

* Minimization of DFA



	0	1
→A	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C

0 Equivalence : Non-Final States → {A, B, C, D} Final States → {E}

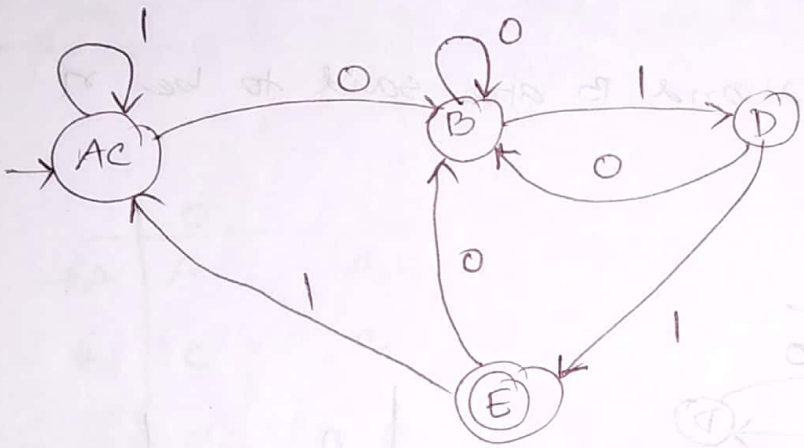
1 Equivalence : {A, B, C} {D} {E}

2 Equivalence : {A, C} {B} {D} {E}

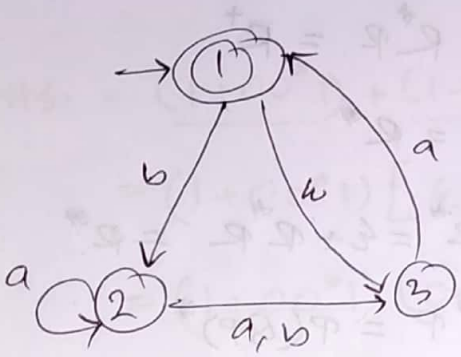
3 Equivalence : {A, C} {B} {D} {E}

Same

Single state



Convert NFA to DFA [Slide]



	a	b	a
→ 1	∅	2	3
2	{2,3}	3	∅
3	~~Select 1~~	~~∅~~ {3}	~~∅~~ ∅ ∅
{2,3}	{1,2,3}	{2,3}	{3}
{1,2,3}	{1,2,3}		

	a	b
→ 1		

Identities of Regular Exp

1. $\emptyset + R = R$
2. $\emptyset R + R\emptyset = \emptyset$
3. $\epsilon R = R\epsilon = R$
4. $\epsilon^* = \epsilon$ and $\emptyset^* = \epsilon$
5. $R + R = R$
6. $R^* R^* = R^*$
7. $RR^* = R^* R = R^+$
8. $(R^*)^* = R^*$
9. $\epsilon + RR^* = \epsilon + R^* R = R^*$
10. $(PQ)^* P = P(QP)^*$
11. $(P+Q)^* = (P^* Q^*)^*$
    $= (P^* + Q^*)^*$
12. $(P+Q)R = PR + QR$
    and $R(P+Q) = RP + RQ$

Arden's Theorem

If P and Q are two Regex over Σ, and if P does not contain ϵ, then the following eqn ~~given~~ in R given by $R = Q + RP$ has a unique sol^n i.e. $R = QP^*$.

Given,

$$R = Q + RP$$
$$= Q + QP^*P \quad [\because R = QP^*]$$
$$= Q(\epsilon + P^*P)$$
$$= QP^* \quad [\text{Proved}]$$

Prove that $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1) = 0^*1(0+10^*1)^*$

L.H.S. $= (1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$

$= (1+00^*1)[\epsilon + (0+10^*1)^*(0+10^*1)]$

$= (1+00^*1)(0+10^*1)^*$ $[\because \epsilon + R^*R = R^*]$

~~$= (\epsilon + 1 + 00^*1)(0+10^*1)^*$ $[\because \epsilon R = R]$~~

~~$= (\epsilon + 00^*)1(0+$~~

$= (\epsilon + 00^*)1(0+10^*1)^*$

$= 0^*1(0+10^*1)^*$ $[\because \epsilon R^* = R^*]$

(R.H.S)

Design Regex for the following languages over $\{a, b\}$

1) Language accepting strings of length exactly 2
2) " " " " atleast 2
3) " " " " at most 2

Soln:

1) $L_1 = \{aa, ab, ba, bb\}$

$\therefore R = aa + ab + ba + bb$

$= a(a+b) + b(a+b)$

$= (a+b)(a+b)$

2) $L_1 = \{aa, ab, ba, bb, aaa, \ldots\}$

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

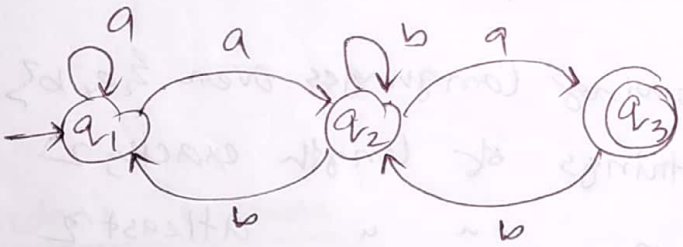
3) $L_1 = \{a, b, ab, aa, ba, bb\}, \varepsilon\}$

$R = \varepsilon + a + b + aa + ab + ba + bb$

~~$= \varepsilon + a + b + ab (\varepsilon + a + b)$~~

$= (\varepsilon + a + b)(\varepsilon + a + b)$

Find the Regex for the following NfA



Incoming transitions

$q_3 = q_2 a$ — (1)

$q_2 = q_1 a + q_2 b + q_3 b$ — (2)

$q_1 = \varepsilon + q_1 a + q_2 b$ — (3)

↓ start

(1) ⇒

$q_3 = q_2 a$

$= (q_1 a + q_2 b + q_3 b)\, a$ [using (2)]

$= q_1 aa + q_2 ba + q_3 ba$ ← (4)

(2) ⇒ $q_2 = q_1 a + q_2 b + q_3 b$

$= q_1 a + q_2 b + q_2 ab$ [using (1)]

$\Rightarrow q_2 = \underbrace{q_1 a}_{Q} + \underbrace{q_2}_{R} \underbrace{(b + ab)}_{P}$ [$R = Q + RP$ from]

$= QP^*$

$\therefore q_2 = (q_1 a)(b + ab)^*$ [using Arden's theorem]

← (5)

(3) $q_1 = \varepsilon + q_1 a + q_2 b$

$= \varepsilon + q_1 a + ((q_1 a)(b + ab)^*)\, b$ [q_2 from (2)]

$\Rightarrow \underbrace{q_1}_{R} = \underbrace{\varepsilon}_{Q} + \underbrace{q_1}_{R} \underbrace{(a + a(b + ab)^*)\, b}_{P}$

$\Rightarrow q_1 = \varepsilon ((a + a(b + ab)^*)\, b)^*$

$\therefore q_1 = ((a + a(b + ab)^*)\, b)^*$ ← (6)

Final state,

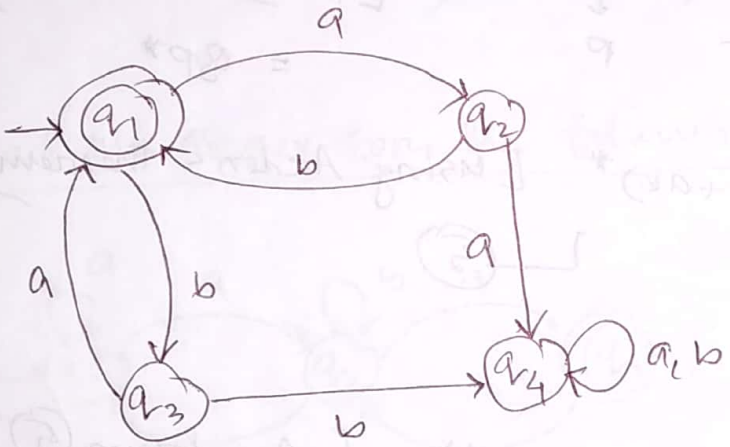
$q_3 = q_2 a$

$= q_1 a (b + ab)^* a$ [q_2 from 2]

$\therefore q_3 = (a + a(b+ab)^* b)^* a(b+ab)^* a$ [q_1 from 6]

(Ans).

Find the Regex for DFA



Incoming Transitions

$q_1 = \varepsilon + q_2 b + q_3 a$ — (1)

$q_2 = q_1 a$ — (2)

$q_3 = q_1 b$ — (3)

$q_4 = q_2 a + q_3 b + q_4 a + q_4 b$ — (4)

(1) ⇒

$q_1 = \epsilon + q_2 b + q_3 a$

$= \epsilon + (q_1 a) b + (q_1 b) a$ [using 2, 3]

$= \epsilon + q_1 ab + q_2 ba$

$\Rightarrow \underset{R}{q_1} = \underset{Q}{\epsilon} + \underset{R}{q_1} \underset{P}{(ab+ba)}$

$= \epsilon (ab+ba)^*$

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

(Ans)

* Regex for DFA (Multiple Final State)

$q_1 = \epsilon + q_1 0$ ——— (1)

$q_2 = q_1 1 + q_2 1$ ——— (2)

$q_3 = q_2 0 + q_3 0 + q_3 1$ ——— (3)

(1) ⇒

$q_1 = \epsilon + q_1 0$

$= \epsilon 0^*$ [Arden's Theorem]

$\therefore q_1 = 0^*$ ——— (4)

(2) ⇒

$$q_2 = q_1 1 + q_2 1$$

$$\underset{R}{q_2} = \underbrace{0^* 1}_{Q} + \underbrace{q_2 1}_{R\ P} \quad [\text{using } (4)]$$

$$\therefore q_2 = (0^* 1)(1^*)^*$$

$\therefore R$ = union of final states

$$= 0^* + \epsilon 0^* 1 1^*$$

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

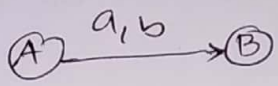
$$= 0^* 1^* \quad [\epsilon + RR^* = R^*]$$

Ans.

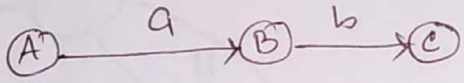
Conversion from Regex to Finita Automata

$(a+b)$ →

or



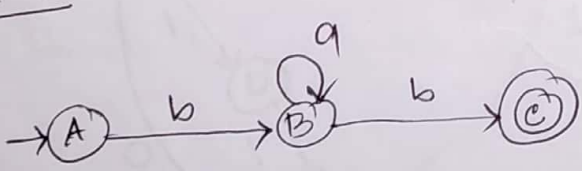
$(a.b)$ →



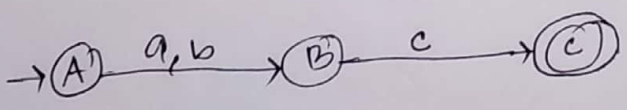
a^* →

Examples

① ba^*b [e.g. bb, bab, baab, ...]



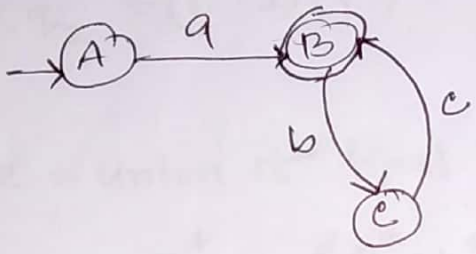
② $(a+b)c$



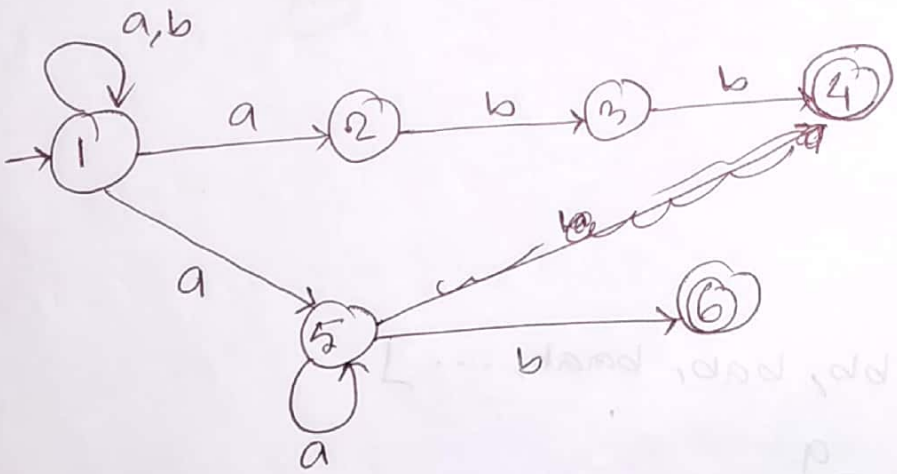
Nayeem Mahmood / 6AM / C161026

③ $a(bc)^*$

e.g. a, abc, abcbc, abcbcbc, ...

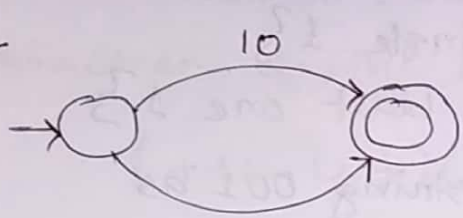


④ $(a|b)^*(abb|a^+b)$



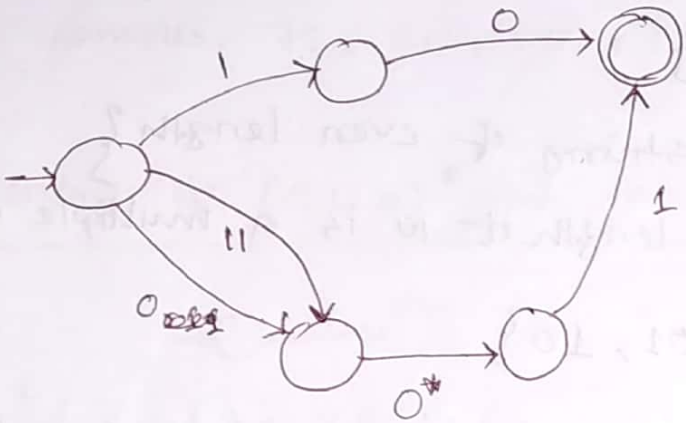
(5) $10 + (0 + 11)0^* 1$

Step 1

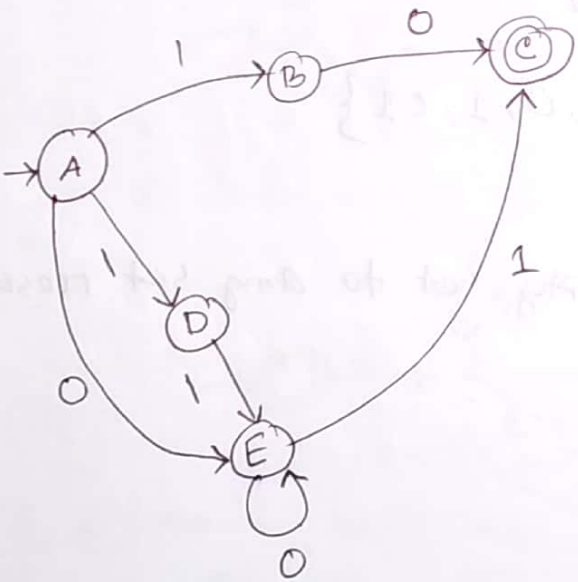


$(0+11)0^*1$

Step 2



Step 3



Some examples of Regex (Here $\Sigma = \{0, 1\}$)

1. $0^* 1 0^* = \{w \mid w$ contains a single $1\}$
2. $\Sigma^* 1 \Sigma^* = \{w \mid w$ contains at least one $1\}$
3. $\Sigma^* 001 \Sigma^* = \{w \mid w$ contains the string 001 as substring$\}$
4. $(01^+)^* = \{w \mid$ every 0 in w is followed by at least one $1\}$
5. $(\Sigma\Sigma)^* = \{w \mid w$ is a string of even length$\}$
6. $(\Sigma\Sigma\Sigma)^* = \{w \mid$ the length of w is a multiple of $3\}$
7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w$ starts and ends with the same symbol$\}$
9. $(0 \cup \epsilon)1^* = 01^* \cup 1^*$
10. $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$
11. $1^*\emptyset = \emptyset$

[Concatenating an empty set to any set results in empty set]

12. $\emptyset^* = \{\epsilon\}$

Nayeem Mahmood / 6AM / C161026

* A numerical constant that may include a fractional part and/or a sign may be described as a member of the language :

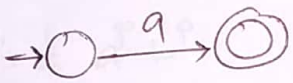
$$(+ \cup - \cup \epsilon)(D^+ \cup D^+ . D^* \cup D^* . D^+)$$

where, $D = \{0, 1, 2, 3, \ldots, 9\}$ is the alphabet of decimal digits.

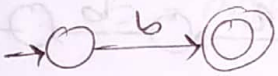
For examples, 72, 3.14159, +7., -.01

* Conversion of $(a \cup b)^* aba$ into NFA

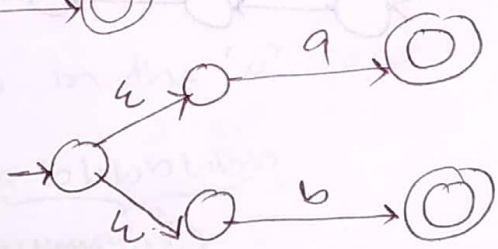
a :



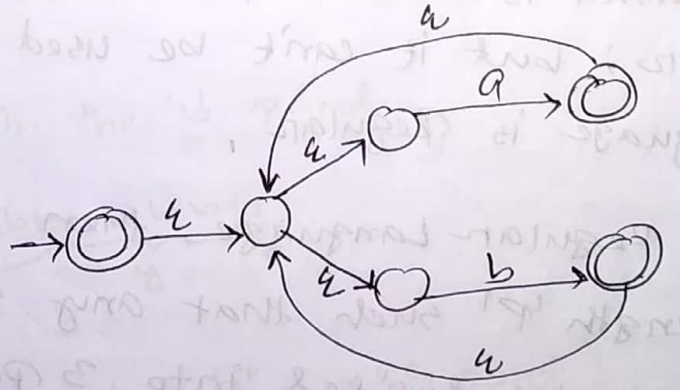
b :



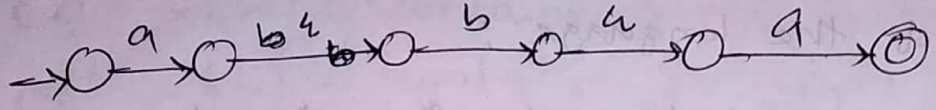
$a \cup b$:



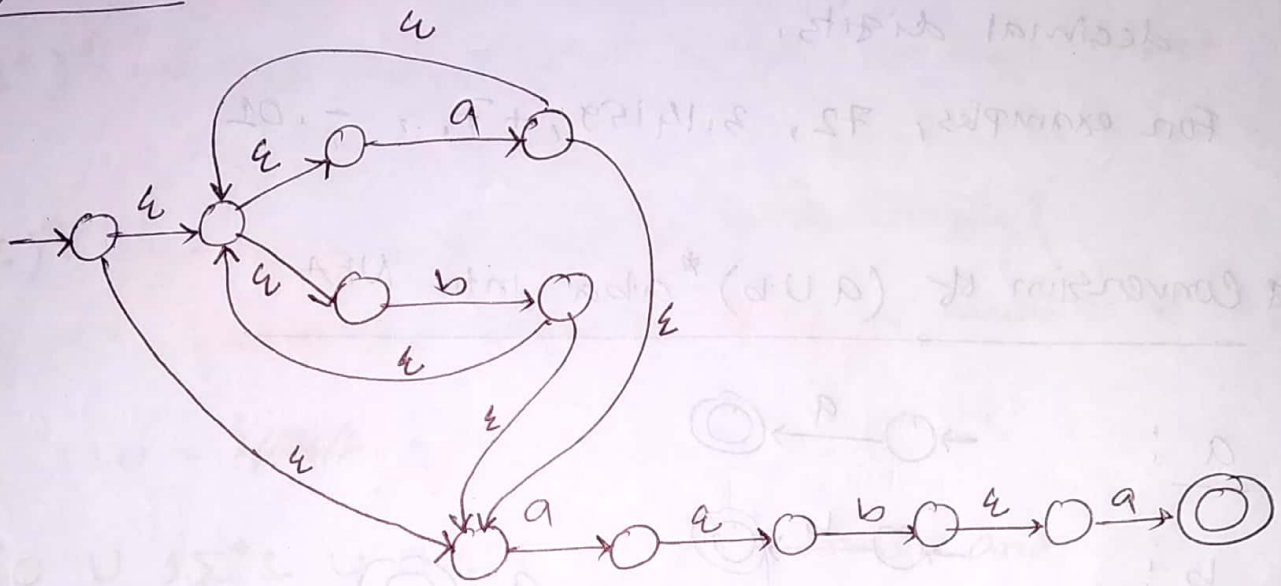
$(a \cup b)^*$:



aba:



$(a \cup b)^* aba$:



Pumping lemma is used to prove that a language is NOT Regular; but it can't be used to prove that a language is Regular.

If A is a Regular Language, then A has a Pumping length 'P' such that any string 's' where $|s| \geq P$, may be divided into 3 parts:

$$s = x\ y\ z$$

such that the following conditions must be true:

1) $xy^iz \in A$ for every $i \geq 0$

2) $|y| > 0$

3) $|xy| \leq p$

Using Pumping Lemma, Prove that the language $A = \{a^n b^n \mid n \geq 0\}$ is NOT Regular.

Proof:

Let us assume,

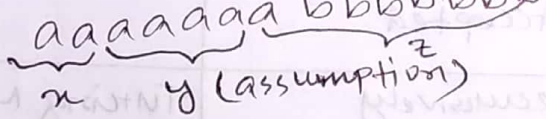
A is regular

Pumping length $= p$

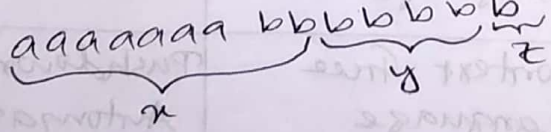
$s = a^p b^p$

Let $p = 7$. $\therefore s = aaaaaaabbbbbbb$

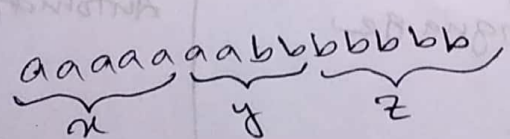
Case 1: The 'y' is in the 'a' part



Case 2: The 'y' is in the 'b' part



Case 3: The 'y' is in the 'a' and 'b' part



Nayeem Mahmood / 6AM / C161026

For Case 1:

$xy^iz \Rightarrow xy^2z \quad [i=2]$

$\underbrace{aa}_{x}\ \underbrace{aaaa}_{y}\ \underbrace{aaaa}_{y}\ \underbrace{abbbbbbb}_{z}$

Total a = 11, b = 7

but $11 \neq 7$ ✗

Since, according to question, number of a's must be equal to number of b's.

Regular Grammar

Noam Chomsky gave a Mathematical model of grammar which is effective for writing computer languages:

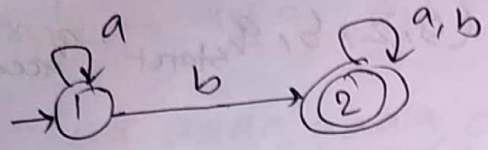
Grammar Type	Grammar Accepted	Language Accepted	Automation
Type-0	Unrestricted Grammar	Recursively Enumerable Language	Turing Machine
Type-1	Context Sensitive Grammar	Context Sensitive Language	Linear Bounded Automation
Type-2	Context Free Grammar	Context Free Language	Pushdown Automata
Type-3	Regular Grammar	Regular Language	Finite State Automata

Formal Def^n of GNFA

A Generalized NFA is a 5-tuple $(Q, \Sigma, \delta, q_{start}, q_{accept})$ where,

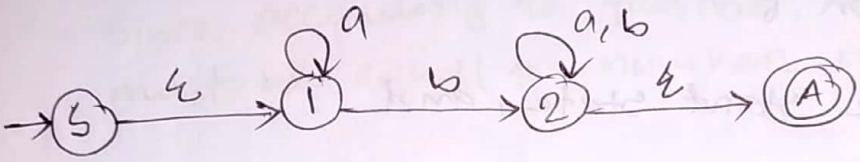
1. Q is the finite set of states
2. Σ is the input alphabet
3. $\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \rightarrow R$ is the transition f^n
4. q_{start} is the start state, and
5. q_{accept} is the accept state

Converting DFA to Regex using GNFA



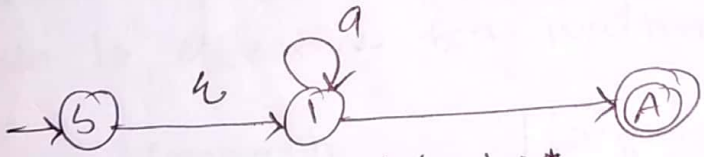
(A)

[Adding new start and accept states]
(S) (A)



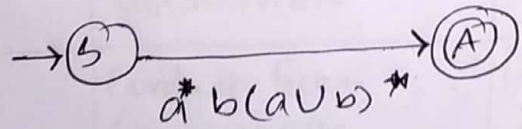
(B)

[Merging (2) and (A)]



(C)

[Merging (1) and (A)]



(D)

Context Free Grammar (CFG)

A CFG has 4 tuple (V, Σ, R, S) where,

1. V is a finite set called the variables
2. Σ is a finite set, disjoint from V, called the terminals,
3. R is a finite set of rules, with each rule being a variable and a string of variables and terminals, and
4. $S \in V$ is the start variable

Construct the grammar for the language

$$\{0^n 1^n \mid n \geq 0\} \cup \{1^n 0^n \mid n \geq 0\}$$

Soln. For the language $\{0^n 1^n \mid n \geq 0\}$, the grammar be

$$S_1 \to 0S_1 1 \mid \epsilon$$

and for " " $\{1^n 0^n \mid n \geq 0\}$, " " "

$$S_2 \to 1S_2 0 \mid \epsilon$$

Now adding the rules $S \to S_1 \mid S_2$ to give the required grammar,

$$S \to S_1 \mid S_2$$
$$S_1 \to 0S_1 1 \mid \epsilon$$
$$S_2 \to 1S_2 0 \mid \epsilon$$

Nayeem Mahmood / 6AM / C161026

Ambiguity

A string w is derived ambiguously in CFG G if it has two or more different leftmost derivations. Grammar G is ambiguous if it generates some string ambiguously.

Ex: $G = (\{S\}, \{a+b, +, *\}, S, P)$ where P consists of

$S \to S+S \mid S*S \mid a \mid b$

The string $a+a*b$ can be generated as:

$S \to S+S$
$\to a+S \quad [S \to a]$
$\to a+S*S \quad [S \to S*S]$
$\to a+a*S \quad [S \to a]$
$\to a+a*b \quad [S \to b]$

$S \to S*S$
$\to S+S*S \quad [S \to S+S]$
$\to a+S*S \quad [S \to a]$
$\to a+a*S \quad [S \to a]$
$\to a+a*b \quad [S \to b]$

Thus, this Grammar is ambiguous

Chomsky Normal Form (CNF)

In CNF, we have a restriction on the length of RHS; which is; elements in RHS should either be two variables or a terminal.

A CFG is in CNF if the productions are in the following forms:

$A \rightarrow a$

$A \rightarrow BC$

where A, B, C are non-terminals and a is terminal.

Convert the following CFG to CNF

P: $S \rightarrow ASA \mid aB$, $A \rightarrow B \mid S$, $B \rightarrow b \mid \varepsilon$

Step 1: [Substitute start state from RHS]

P: $S' \rightarrow S$, $S \rightarrow ASA \mid aB$, $A \rightarrow B \mid S$, $B \rightarrow b \mid \varepsilon$

Step 2: [Remove Null Production]

$B \rightarrow \varepsilon$; $A \rightarrow B$
$\rightarrow \varepsilon$

After removing $B \rightarrow \varepsilon$:

P: $S' \rightarrow S$, $S \rightarrow ASA \mid aB \mid a$, $A \rightarrow B \mid S \mid \varepsilon$, $B \rightarrow b$

After removing $A \rightarrow \varepsilon$:

P: $S' \rightarrow S$, $S \rightarrow ASA \mid aB \mid a \mid AS \mid SA \mid S$, $A \rightarrow B \mid S$, $B \rightarrow b$

Step 3 : [Removing Unit Production]

Unit Productions where are :

$S' \to S$, $S \to S$, $A \to B$, $A \to S$

Removing $S \to S$

P: $S \to ASA \mid aB \mid a \mid AS \mid SA$, ...

Removing $S' \to S$:

P: $S' \to ASA \mid aB \mid a \mid AS \mid SA$,

$S \to ASA \mid aB \mid a \mid AS \mid SA$,

$A \to B \mid S$, $B \to b$

Removing $A \to B$

P: $S' \to \ldots$

$S \to \ldots$

$A \to b \mid S$, $B \to b$

Removing $A \to S$

P: $S' \to \ldots$

$S \to \ldots$

$A \to b \mid ASA \mid aB \mid a \mid AS \mid SA$

$B \to b$

Step 4: [Finding out Production rules that have more than Two variables in RHS]

$S' \to ASA$, $S \to ASA$ and $A \to ASA$

After removing these, we get,

P: $S' \to AX|aB|a|AS|SA$,
$S \to AX|aB|a|AS|SA$,
$A \to b|AX|aB|a|AS|SA$,
$B \to b$
$X \to SA$

Step 5: Changing the productions!

$S' \to aB$, $S \to aB$ and $A \to aB$

Finally we get,

P: $S' \to AX|YB|a|AS|SA$,
$S \to AX|YB|a|AS|SA$,
$A \to b|AX|YB|a|AS|SA$,
$B \to b$,
$X \to SA$,
$Y \to a$

which is the required CNF.

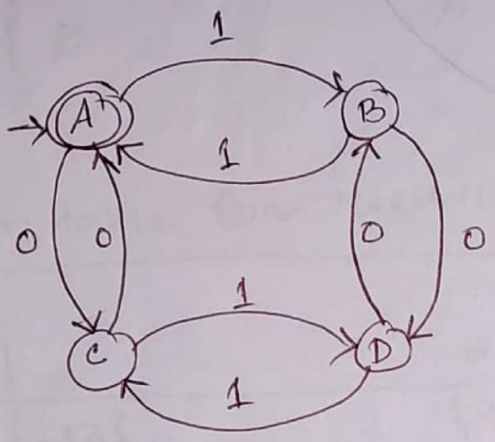
Nayeem Mahmood / 6AM / C161026

Sp 16

1b

Given,

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

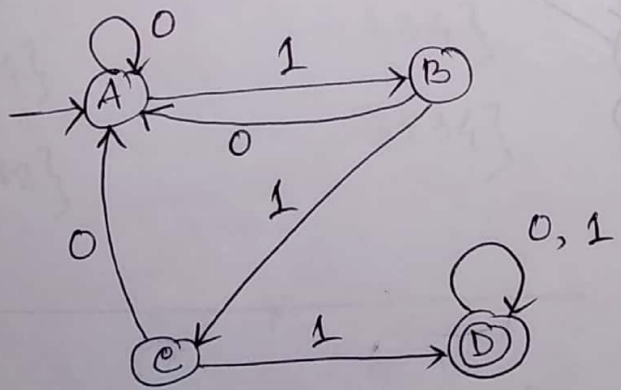


For the input 110101, sequence of states is :

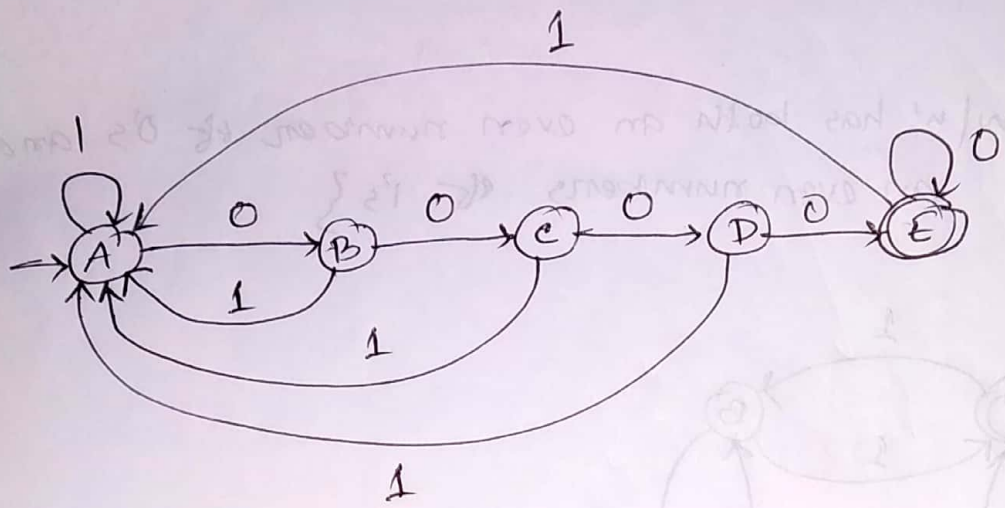
$A \xrightarrow{1} B \xrightarrow{1} A \xrightarrow{0} C \xrightarrow{1} D \xrightarrow{0} B \xrightarrow{1} A$

1c

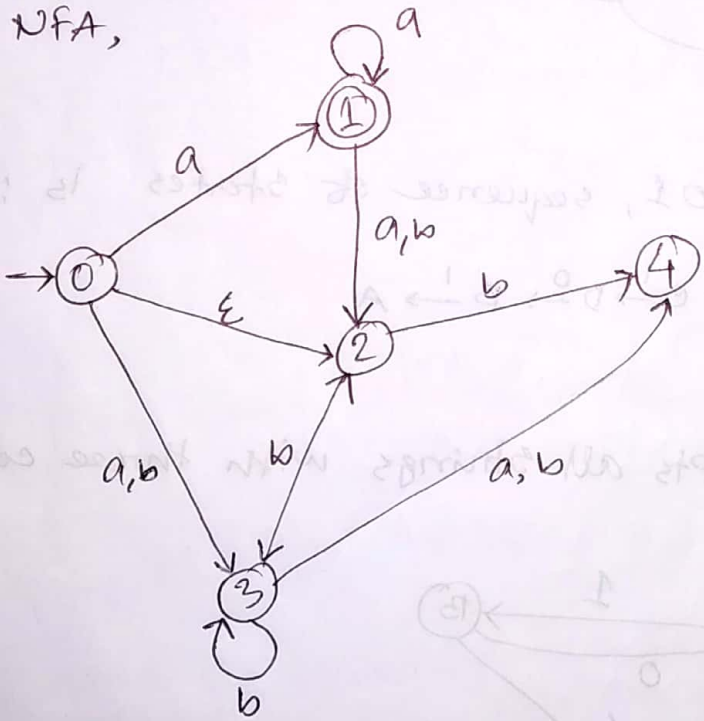
DFA which accepts all strings with three consecutive 1's :



DFA which accepts the set of all strings ending with 0000.



2b Given NFA,



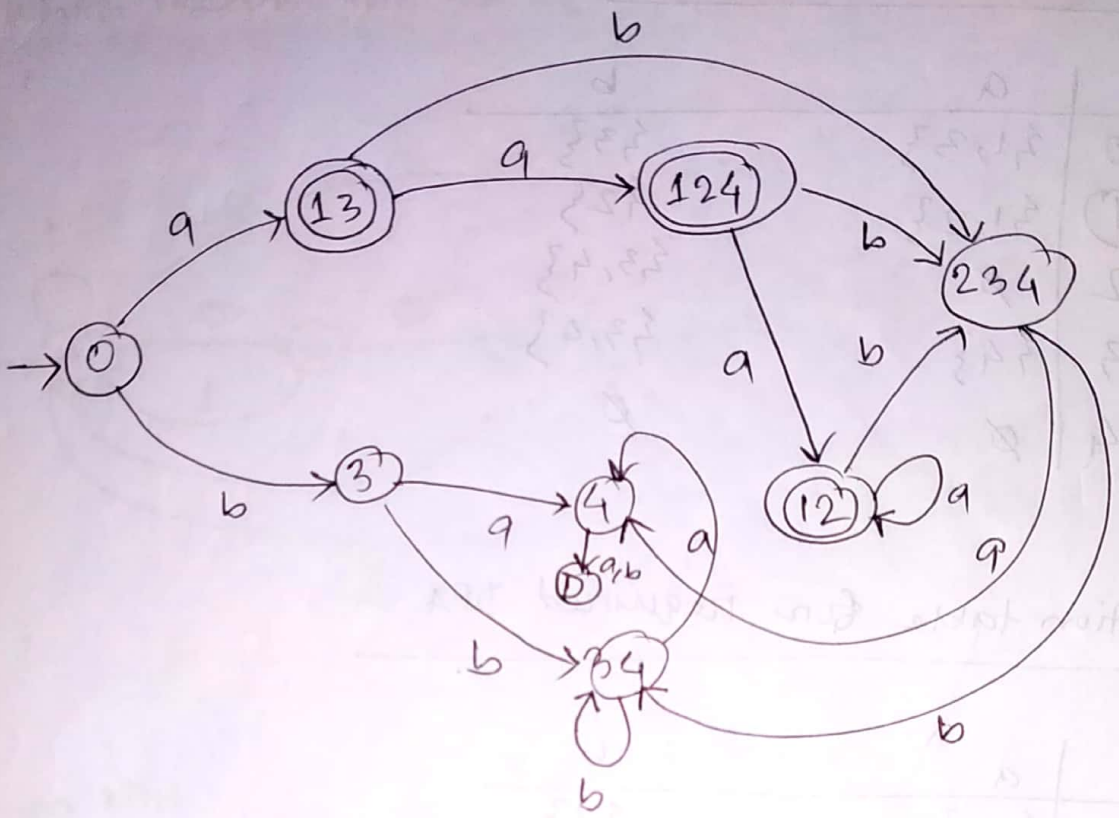
Transition table for given NFA

	a	b
→0	{1,3}	{3}
(1)	{1,2}	{2}
2	∅	{3,4}
3	{4}	{3,4}
4	∅	∅

Transition table for required DFA

	a	b
→0	{13}	{3}
{13}	{124}	{234}
{3}	{4}	{34}
{124}	{12}	{234}
{234}	{4}	{34}
{4}	D	D [D is a dead/trap state]
{34}	{4}	{34}
{12}	{12}	{234}

Nayeem Mahmood / 6AM / C161026

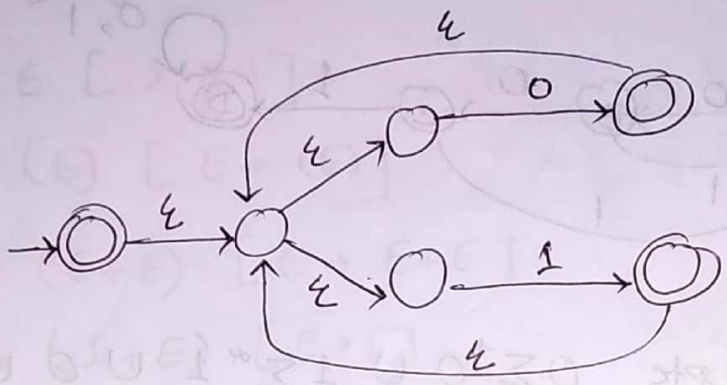


This is the required DFA.

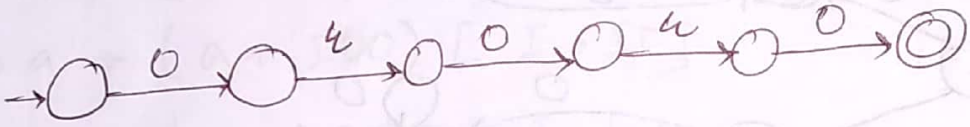
2c/ NFA that accepts all strings where either 101 or 11 is present as substring:

36/ Converting $(0 \cup 1)^* 000 (0 \cup 1)^*$ into NFA :

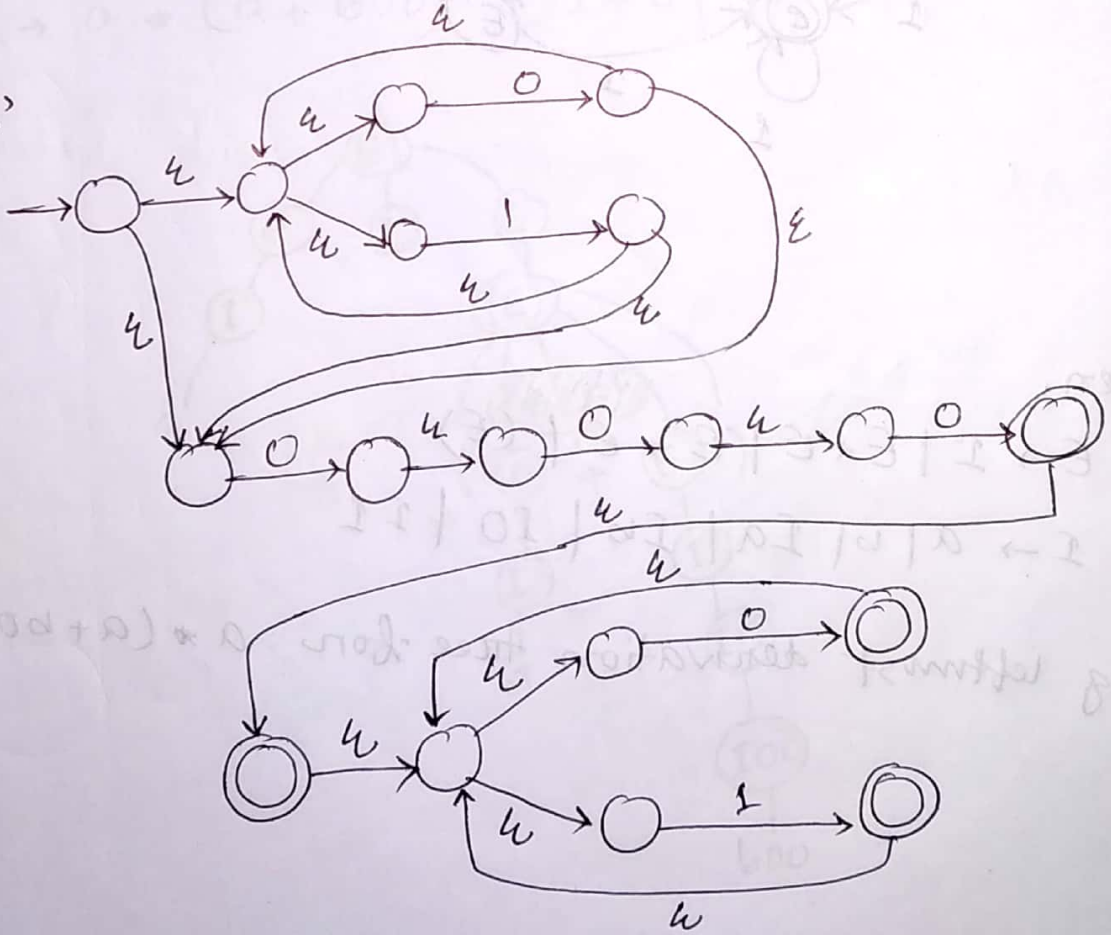
$(0 \cup 1)^*$



000

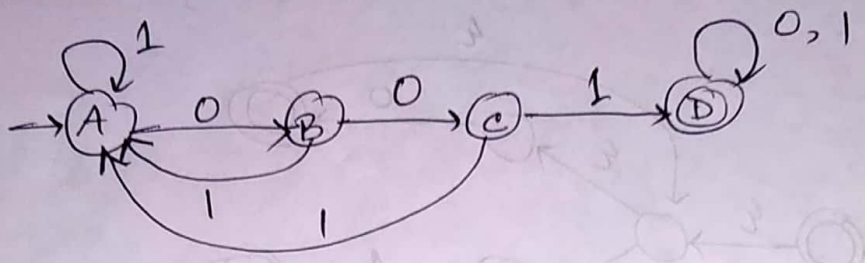


Finally,



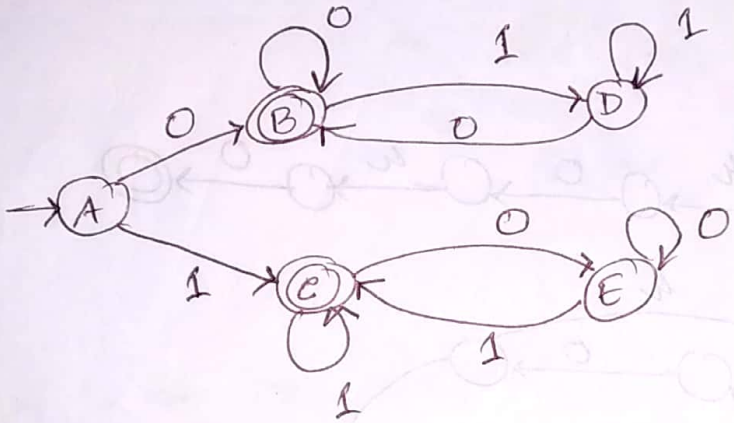
Nayeem Mahmood / 6AM / C161026

3c/ DFA that accepts $\Sigma^* 001 \Sigma^*$:



DFA that accepts $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$

[starts and ends with same symbol]



4/c/ Given,

$E \rightarrow I \mid E+E \mid E*E \mid (E)$

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

Parsing leftmost derivation tree for $a*(a+b00)$

$E \rightarrow E * E$

$\rightarrow I * E \quad [E \rightarrow I]$

$\rightarrow a * E \quad [I \rightarrow a]$

$\rightarrow a * (E) \quad [E \rightarrow (E)]$

$\rightarrow a * (E+E) \quad [E \rightarrow E+E]$

$\rightarrow a * (I+E) \quad [E \rightarrow I]$

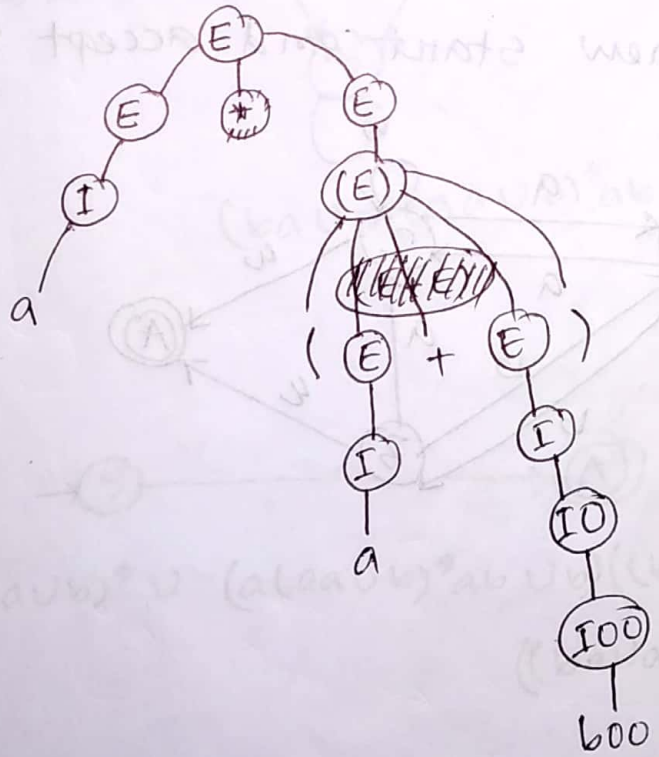
$\rightarrow a * (a+E) \quad [I \rightarrow a]$

$\rightarrow a * (a+I) \quad [E \rightarrow I]$

$\rightarrow a * (a+I0) \quad [I \rightarrow I0]$

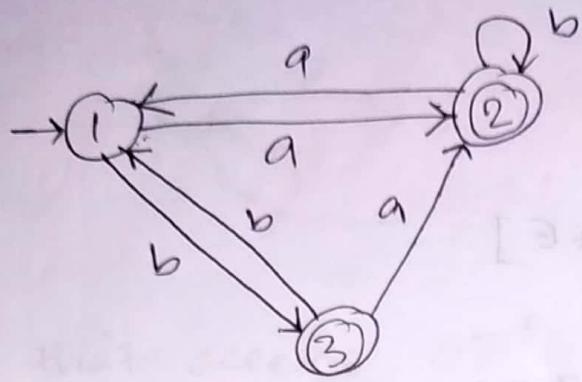
$\rightarrow a * (a+I00) \quad [I \rightarrow I0]$

$\rightarrow a * (a+b00) \quad [I \rightarrow b]$



Nayeem Mahmood / 6AM / C161026

4d/ Converting the DFA into Regex:



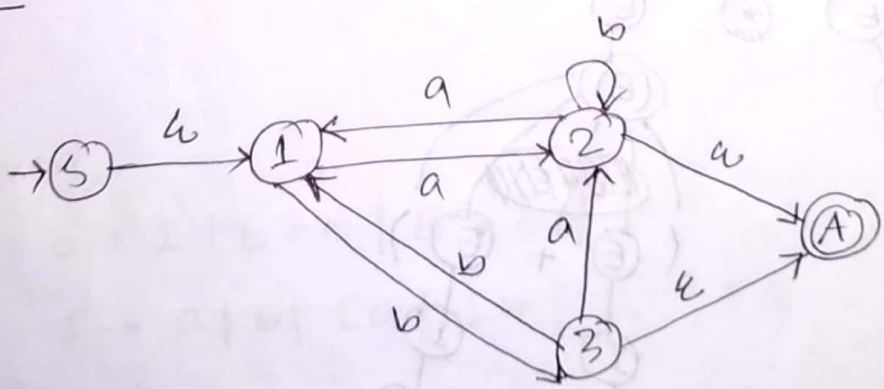
Incoming Transitions

(2) = 1a + 2b — (i)

(3) = 1b — (ii)

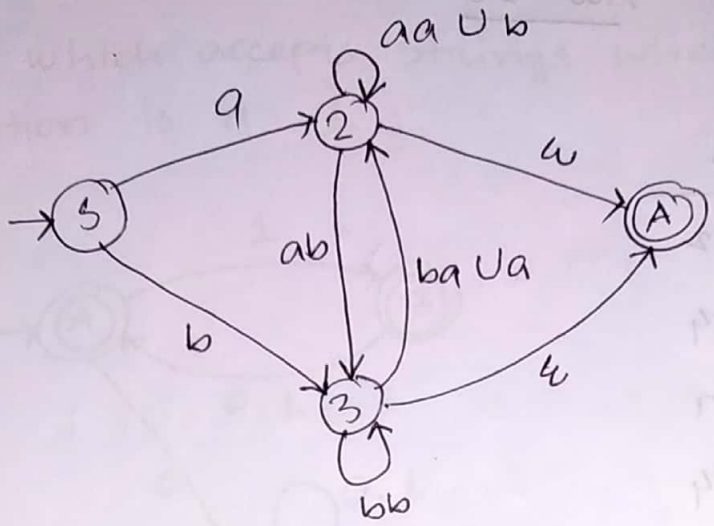
(1) = ε + 2a + 3b — (iii)

Step 1: Adding new start and accept states:

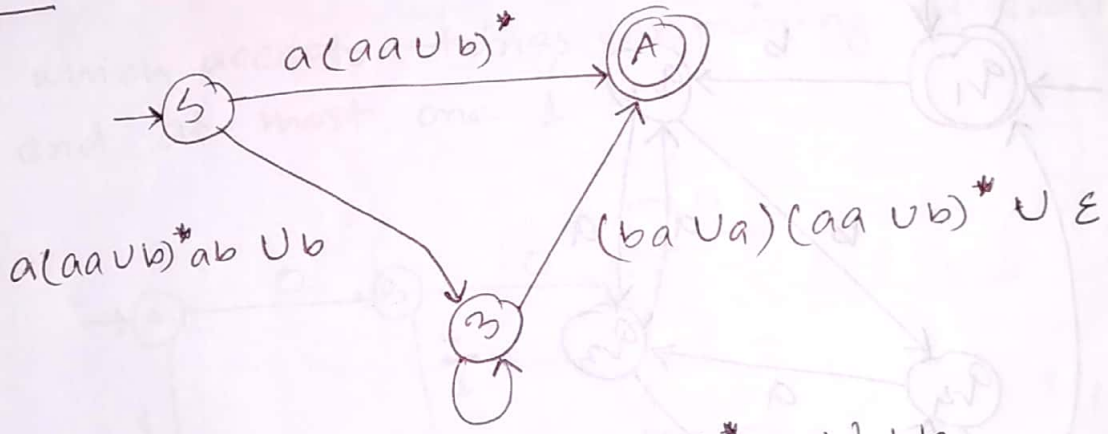


Nayeem Mahmood / 6AM / C161026

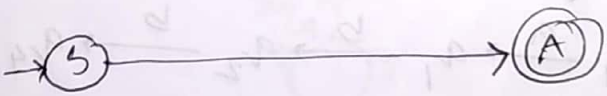
Step 2:



Step 3:



Step 4:



$a(aa \cup b)^* \cup (a(aa \cup b)^*ab \cup b)((ba \cup a)(aa \cup b)^*ab \cup bb)^* ((ba \cup a)(aa \cup b)^* \cup \varepsilon)$

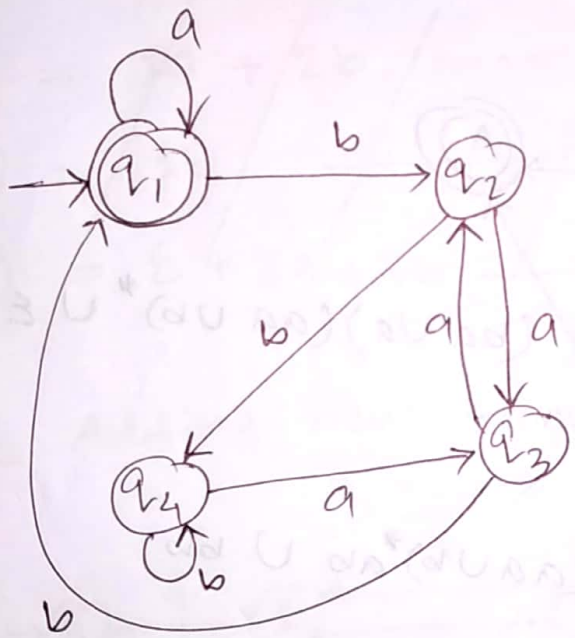
(Ans)

Nayeem Mahmood / 6AM / C161026

Aut 16

1a/

	a	b
→ q_1	q_1	q_2
q_2	q_3	q_4
q_3	q_2	q_1
q_4	q_3	q_4



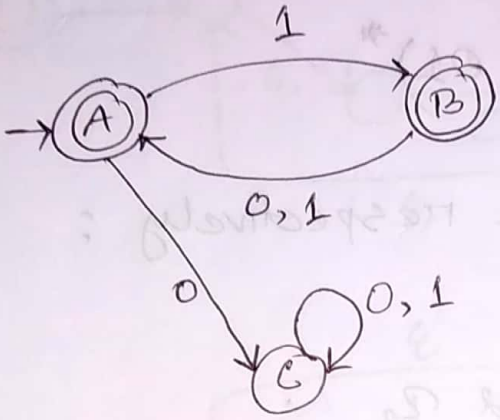
For the given string "aabb" ;

$$q_1 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{b} q_4$$

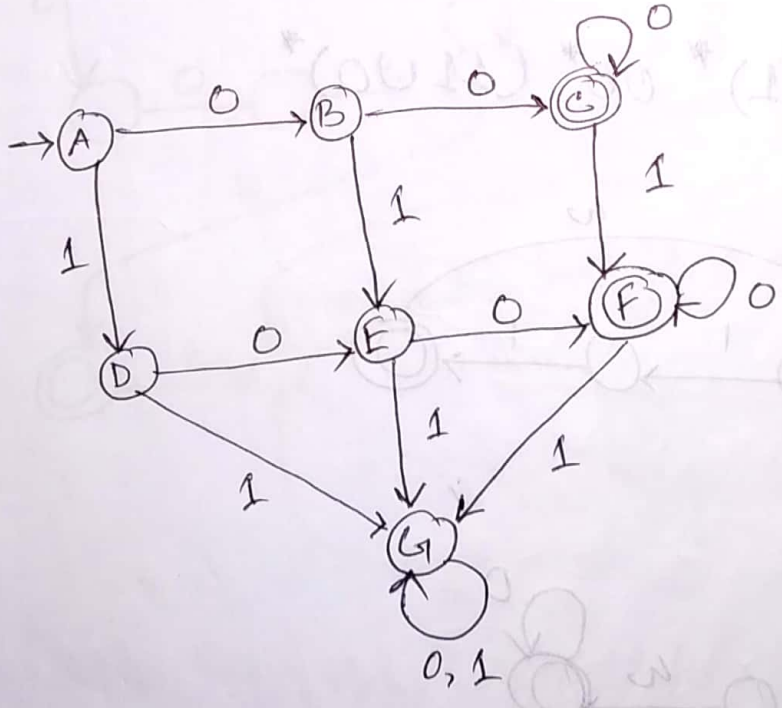
So, this machine rejects "aabb".

Nayeem Mahmood / 6AM / C161026

1b/ DFA which accepts strings where every odd position is a 1 :



DFA which accepts strings containing at least two 0's and at most one 1 :



Nayeem Mahmood / 6AM / C161026

3a/

$R_1 = (0 \cup 1)^* 000 (0 \cup 1)^*$

$R_2 = (((00)^* (11)) \cup 01)^*$

Members of R_1 & R_2 are respectively :

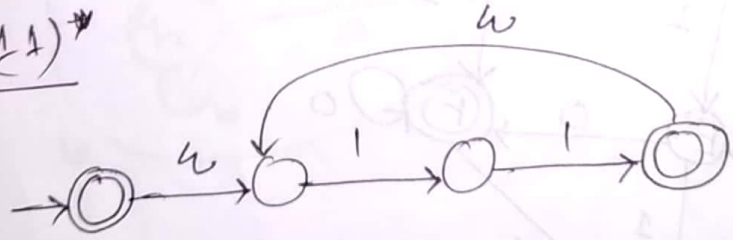
000 and 01

Non-member of R_1 and R_2 :

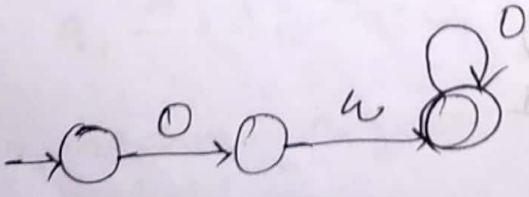
00 and 000

3b/ Given, $R = (11)^* 00^* (11 \cup 0)^*$

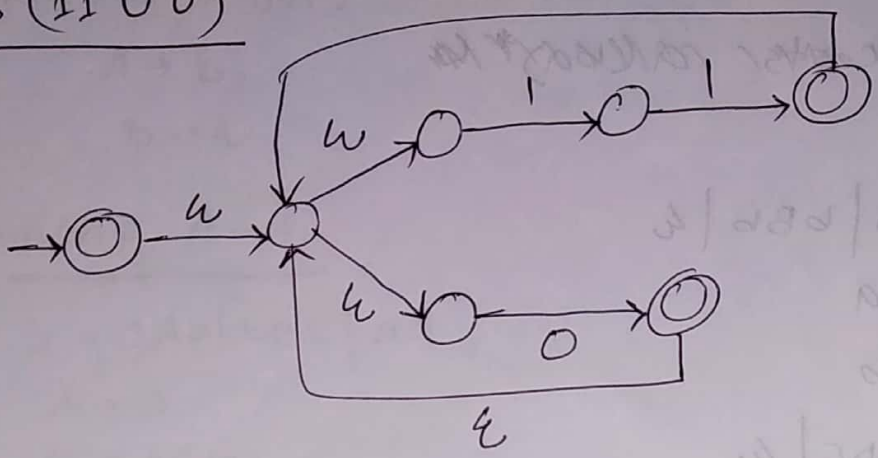
For $(11)^*$



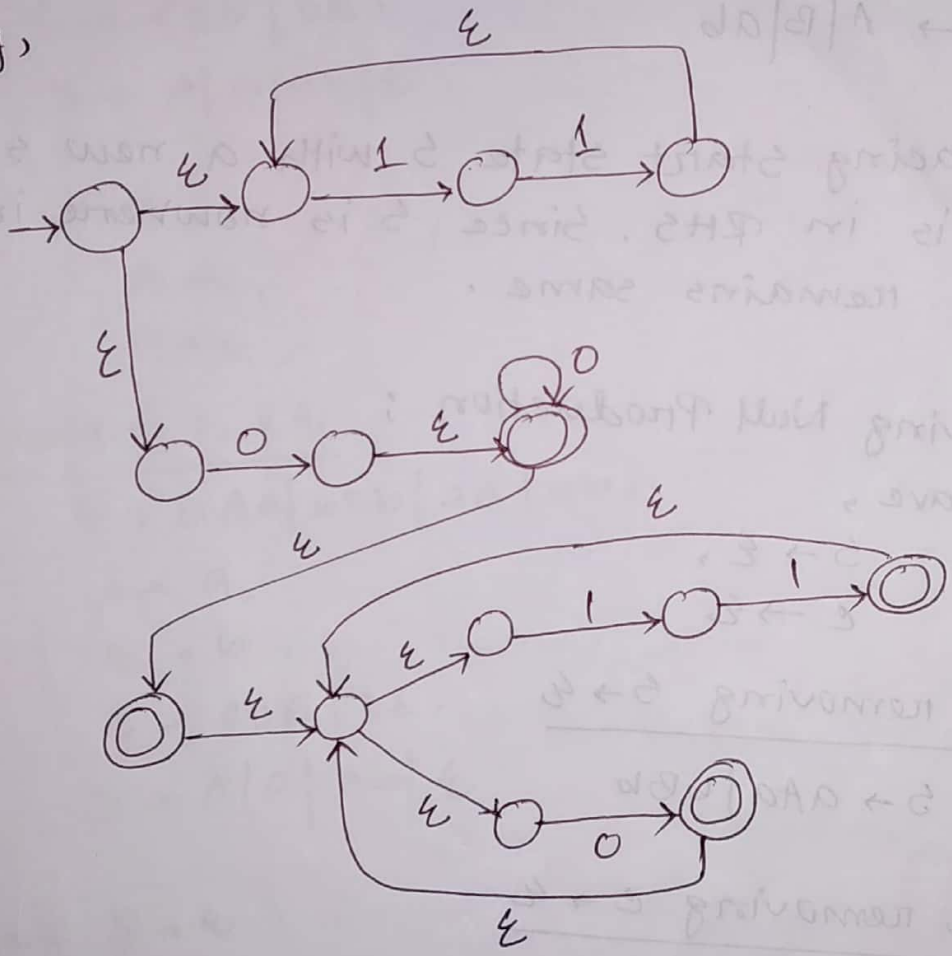
For 00^*



For $(11 \cup 0)^*$



Finally,



Nayeem Mahmood / 6AM / C161026

3c/ ~~DFA that accepts ca...~~

Given CFG,

$S \to aAa \mid bBb \mid \varepsilon$

$A \to C \mid a$

$B \to C \mid b$

$C \to CDE \mid \varepsilon$

$D \to A \mid B \mid ab$

Step 1: Replacing start state S with a new state where S is in RHS. Since S is nowhere in R.H.S., CFG remains same.

Step 2: Removing Null Production;

We have,

$S \to \varepsilon$,

$C \to \varepsilon$

After removing $S \to \varepsilon$

$S \to aAa \mid bBb$

After removing $C \to \varepsilon$

$A \to \varepsilon \mid a$

$B \to \varepsilon \mid b$

$C \to CDE \mid DE$

Again, we've got,

$A \to \varepsilon$,

$B \to \omega$

Removing $A \to \varepsilon$

$S \to aAa \mid bBb \mid aa$,

$A \to a$,

$B \to \omega \mid b$,

$C \to CDE \mid DE$,

$D \to A \mid B \mid ab \mid \varepsilon$,

Now we've,

$B \to \varepsilon$,

$D \to \omega$

Removing $B \to \omega$

$S \to aAa \mid bBb \mid aa \mid bb$,

$A \to a$,

$B \to b$,

$C \to CDE \mid DE$,

$D \to A \mid B \mid ab \mid \varepsilon$

Removing $D \to \omega$

$S \to aAa \mid bBb \mid aa \mid bb$,

$A \to a$

$B \to b$

$C \to CDE \mid DE \mid CE \mid E$

$D \to A \mid B \mid ab$

Nayeem Mahmood / 6AM / C161026

Step 3: Removing Unit Production :

We've Unit Production :

$C \to E$,

$D \to A$,

$D \to B$

Removing $D \to A$

$D \to a$ [since $A \to a$]

Removing $D \to B$

$D \to b$ [since $B \to b$]

Removing $C \to E$

$C \to$?

Step 4: Finding out Production Rules that have more than 2 Variables in R.H.S.

$S \to aAa \mid bBb \mid aa \mid bb$

$\to AAA \mid BBB \mid AA \mid BB$ [$\because A \to a, B \to b$]

$A \to a$, $B \to b$ $\Big| \to YA \mid ZB \mid AA \mid BB$ [Let $Y = AA$, $Z = BB$]

$C \to CDE \mid DE \mid CE$

$\to XE \mid DE \mid CE$ [let $X = CD$]

$D \to A \mid B \mid AB$

(Ans)

Nayeem Mahmood / 6AM / C161026