WRITTEN HW - I

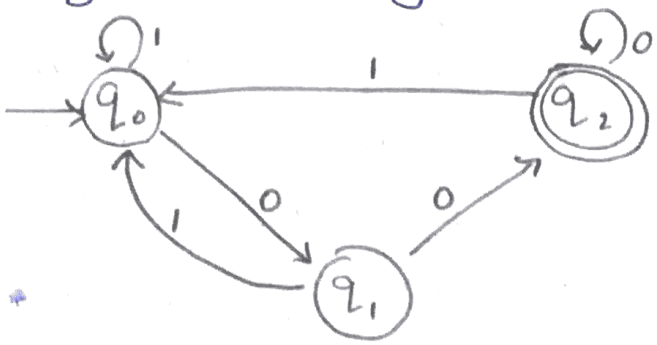
Foundations of Computer Science

Kiran Shettar

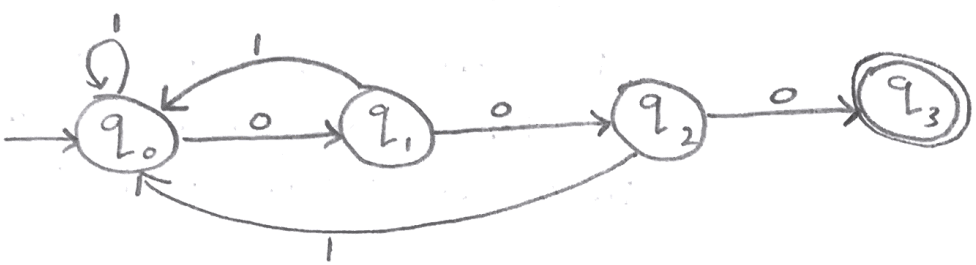
10/19/16

2.2.4:

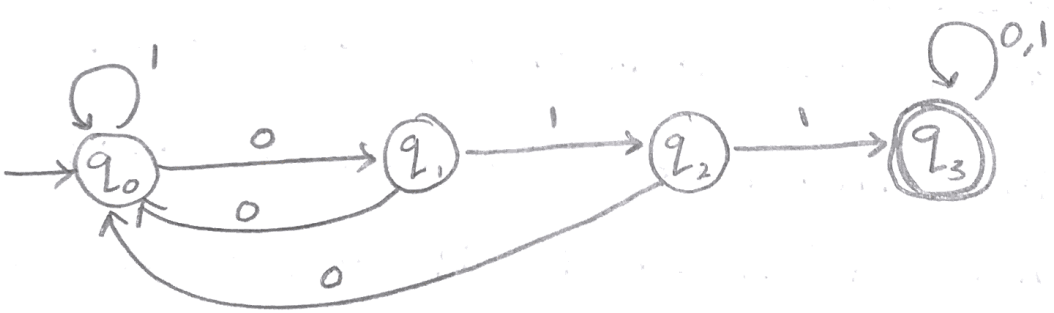
(a) Set of all strings ending in 00:



(b) Set of strings with three consecutive 0's:



(c) Set of strings with 011 as substring:



2.2.1 (a)

	A	B
→000 r	100 r	011 r
*000 a	100 r	011 r
*001 a	101 r	000 a
010 r	110 r	001 a
*010 a	110 r	001 a
011 r	111 r	010 a
100 r	010 r	111 r
100 a	010 r	111 r
101 r	011 r	100 a
101 a	011 r	100 a
110 r	000 a	101 a
110 a	000 a	101 a
111 r	001 a	110 a

* State corresponds to eight combinations of the switch Hence, there will be 16 possible combinations.

This also indicates if the previous input was accepted. Let '0' represent the position towards left & '1' represent position towards right.

Each ~~case~~ state can be represented by a sequence of three 0's and 1's, representing the direction of three switch. in order from left to right.

We follow these bits where 'a' is an accepting state & 'r' is a non-accepting state. Of the possible 16 states, it turns out that only 13 are accessible from the initial state, 000r. I've written the transition table above.

(b) when the input ends with a 1 it'll go to acceptance state. or if it has even number of 1's then also it'll go to acceptance state.

(c) See page 25

2.2.2 If $y = \epsilon$ then $\hat{\delta}(q,x) = \hat{\delta}(\hat{\delta}(q,x),\epsilon)$

We treat $\hat{\delta}(q,x)$ as if it was just a state 'p'.

Then we have to prove $p = \hat{\delta}(p,\epsilon)$

Induction: Assume the strings shorter than 'y', and break $y = za$, where 'a' is the last symbol of 'y'.

Below are the steps for converting $\hat{\delta}(\hat{\delta}(q,x),y)$ to $\hat{\delta}(q,xy)$

Step 1: Start

$\hat{\delta}(\hat{\delta}(q,x),y)$

Step 2: Assume $y = za$, then

$\hat{\delta}(\hat{\delta}(q,x),za)$

Step 3: Definition of $\hat{\delta}$ while treating $\hat{\delta}(q,x)$ as a state

$\delta(\hat{\delta}(\hat{\delta}(q,x),z),a)$

Step 4: Inductive hypothesis

$\delta(\hat{\delta}(q,xz),a)$

Step 5: $\hat{\delta}$ will be

$\hat{\delta}(q,xza)$

$\Rightarrow \hat{\delta}(q,xy)$ where $y = za$,

2.2.8

(a) Base case $n = 0$

$\hat{\delta}(q, \epsilon) = q$ by definition

Induction step:

Assume that $\hat{\delta}(q, a^k) = q$

$$\hat{\delta}(q, a^{k+1}) = \hat{\delta}(\hat{\delta}(q, a^k), a)$$
$$= \hat{\delta}(q, a)$$
$$= q$$

$\therefore$ By induction $\hat{\delta}(q, a^n) = q$ for all $n \geq 0$

(b) From (a), for all $n \geq 0$, $\hat{\delta}(q, a^n) = q$

This is true for the intial q_0.

For any input in $\{a^*\}$, we will end in state q_0.

Suppose our intial state q_0 is a final state, we accept every string in $\{a^*\}$.

$\therefore \{a^*\} \subseteq L(A)$

OR

Suppose q_0 is not a final state, then we reject every string in $\{a^*\}$, so $\{a^*\} \cap L(A) = \emptyset$

Hence, $\{a\}^* \cap L(A) = \emptyset$

2.2.3 This is the converse of 2.2.2

We have ~~$\hat{\delta}(q,ax)$~~ $\hat{\delta}(q, ax)$. For strings shorter than 'x', break $x = q_s \cdot r$ where 'r' is the last symbol of 'x'.

Step 1: Start

$\delta(q, ax)$

Step 2: $x = q_s \cdot r$ ∵ Assumption

Step 3: Def^n of $\hat{\delta}$ treating $\delta(q,a)$ as a state

$\hat{\delta}(\hat{\delta}(q,a), q_s r)$

Step 4: $x = q_s r$

$\hat{\delta}(\hat{\delta}(q,a), x)$

2.2.7

Base Case: Consider $\hat{\delta}(q, \epsilon)$

WKT $\hat{\delta}(q,\epsilon) = q$

Let us assume the case, upon a transition from $\hat{\delta}(q, wa) = q$

On the next transition

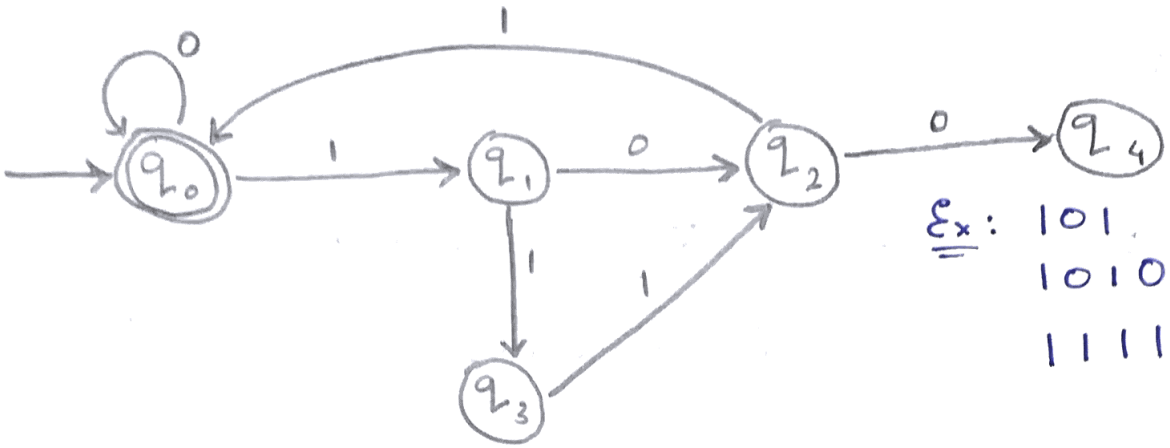
$\hat{\delta}(q, wa) = \hat{\delta}(q, w) + \hat{\delta}(q, a)$

$= q + q$

$= q$ ⇒ proved.

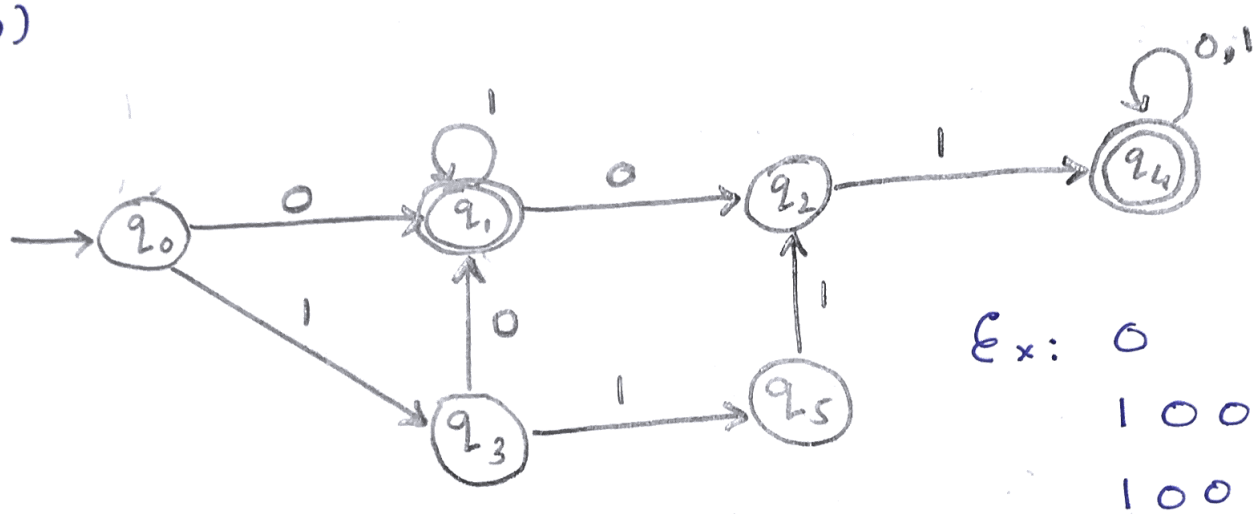
2.2.6

(a) Set of all strings beginning with 1, when interpreted as a binary integer is a multiple of 5.



Ex: 101
1010
1111

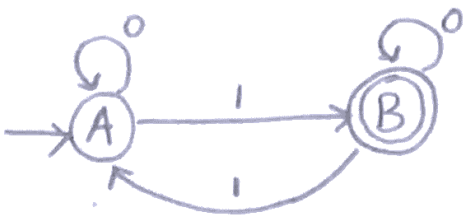
(b)



Ex: 0
100 11
100 1100
0101
1111
101

Set of all strings that, when interpreted in reverse as a binary integer, divisible by 5.

2.2.10



We claim that the language of the DFA is $\{w \mid w$ contains odd number of 1's$\}$

Proof by induction:

$$\delta(A, w) = \begin{cases} A & \text{If } w \text{ contains even no of 1's} \\ B & \text{If } w \text{ contains odd no of 1's} \end{cases}$$

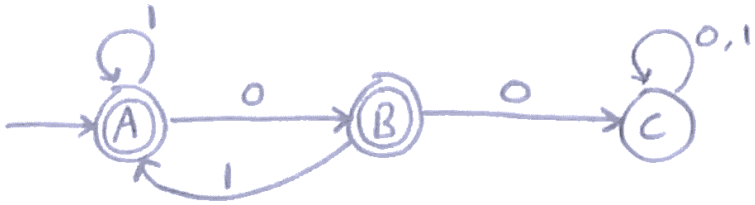
Base case: $w = \varepsilon$

$\hat{\delta}(A, \varepsilon) = A$, as ε has even number of 1's

$$\hat{\delta}(A, w) = \hat{\delta}(A, v1) = \delta(\hat{\delta}(A, v1), 1)$$

$$\Rightarrow \begin{cases} \delta(A, 1) & \text{if } v \text{ contains an even no of 1's} \\ \delta(B, 1) & \text{if } v \text{ contains odd no. of 1's.} \end{cases}$$

2.2.11



For the given transition table, is the above DFA

* The string will always end with one '0'.
* The string will end with many 1's.
* If there are more than one zero in the string, it'll not end.
* The string can have many 1's.

Proof by induction:

$$\delta(A, \omega) = \begin{cases} A, \text{If } \omega \text{ contains one or more 1's} \\ B, \text{If } \omega \text{ contains '0' as the last bit. and only 1 zero.} \\ C, \text{string not accepted.} \end{cases}$$

$\hat{\delta}(A, 0) = B$

$\hat{\delta}(B, 1) = A$

$\delta(\hat{\delta}(A, 0), 1) = A$

$\delta(\hat{\delta}(A, 01), 1) = A$

$\delta(\hat{\delta}(A, 011), 0) = B$

NOTE: For a string 0110

2.3.1 Convert to a DFA the following NFA

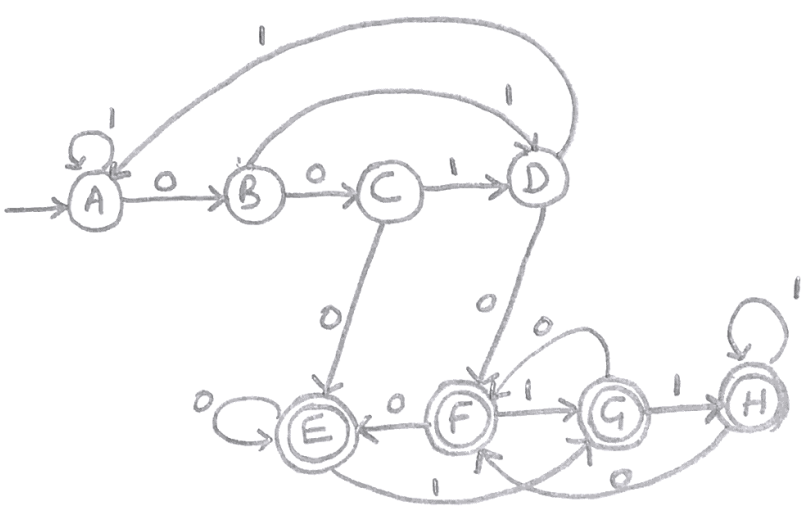
	0	1
→ p	{p, q}	{p}
q	{r}	{r}
r	{s}	{φ}
* s	{s}	{s}

Step 1: Subset construction

	0	1
→ P	{p, q}	{p}
{p, q}	{p, q, r}	{p, r}
{p, q, r}	{p, q, r, s}	{p, r}
{p, r}	{p, q, s}	{p}
* {p, q, r, s}	{p, q, r, s}	{p, r, s}
* {p, q, s}	{p, q, r, s}	{p, r, s}
* {p, r, s}	{p, q, s}	{p, s}
* {p, s}	{p, q, s}	{p, s}

Step 2: Renaming

	0	1
→ A	B	A
B	C	D
C	E	D
D	F	A
* E	E	G
* F	E	G
* G	F	H
* H	F	H



Step 3: DFA from NFA

2.3.2 DFA from NFA

Given

	0	1
→ p	{q, s}	{q}
* q	{r}	{q, r}
r	{s}	{p}
* s	{ϕ}	{p}

Subset.

	0	1
→ {p}	{q, s}	{q}
* {q, s}	{r}	{p, q, r}
* {q}	{r}	{q, r}
{r}	{s}	{p}
* {p, q, r}	{q, r, s}	{p, q, r}
* {q, r}	{r, s}	{p, q, r}
* {s}	ϕ	{p}
* {q, r, s}	{r, s}	{p, q, r}
* {r, s}	{s}	{p}

Renaming.

	0	1
→ A	B	C
* B	D	E
* C	D	F
D	G	A
* E	H	E
* F	I	E
* G	ϕ	A
* H	I	A
* I	G	A

2.3.3 NFA to DFA, informally describe language it accepts.

	0	1
→ p	{p,q}	{p}
q	{r,s}	{t}
r	{p,r}	{t}
* s	∅	∅
* t	∅	∅

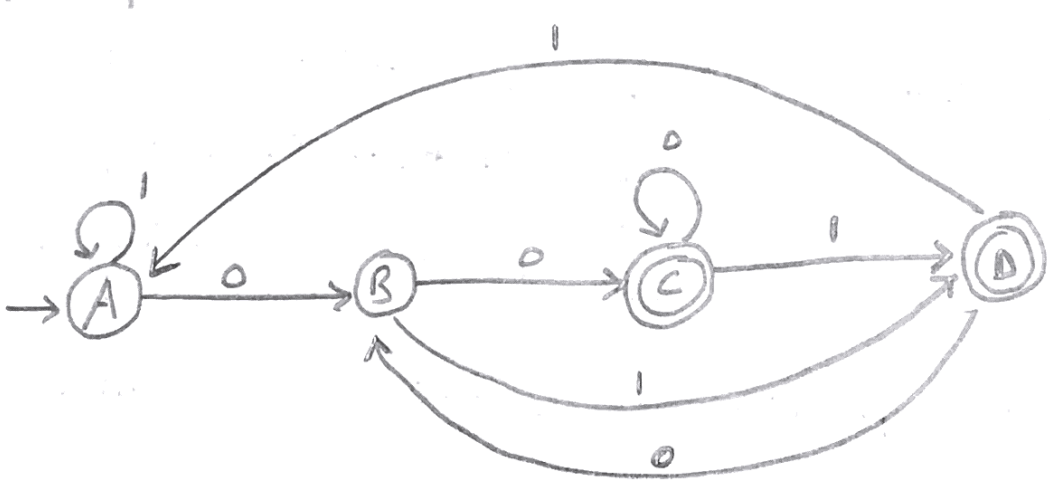
Subset construction

	0	1
→ p	{p,q}	{p}
{p,q}	{p,q,r,s}	{p,t}
*{p,q,r,s}	{p,q,r,s}	{p,t}
*{p,t}	{p,q}	{p}

Renaming

	0	1
→ A	B	A
B	C	D
* C	C	D
* D	B	A

→ The below DFA accepts the strings that end with 001 and the string that has all 0's.



→ This accepts strings end with last digit as 1 & the rest are 0's.

→ Accepts strings ending with 101.

2.3.5 Proof by induction:

Base Case:

$$|w| = 0 \rightarrow |w| = \epsilon$$

$$\hat{\delta}_D(q_0, \epsilon) = \{q_0\} = \hat{\delta}_N(q_0, \epsilon)$$

Induction:

Assume $\hat{\delta}_D(q_0, w) = \hat{\delta}_N(q_0, w)$

Let $w = xa$

where, $\hat{\delta}_N(\{q_0\}, x) = \hat{\delta}_N(\{q_0\}, x)$

By induction

$$\hat{\delta}_N(q_0, w) = \bigcup_{i=1}^{k} \delta_N(P_i, a) \quad \text{---- <i>}$$

By subset construction

$$\delta_D(\{P_1, P_2 \ldots, P_k\}, a) = \bigcup_{i=1}^{k} \delta_N(P_i, a) \quad \text{---- <ii>}$$

$$\delta_D(\{q_0\}, w) = \delta_D(\hat{\delta}_D(\{q_0\}, x), a)$$

$$= \delta_D(\{P_1, P_2 \ldots P_k\}, a)$$

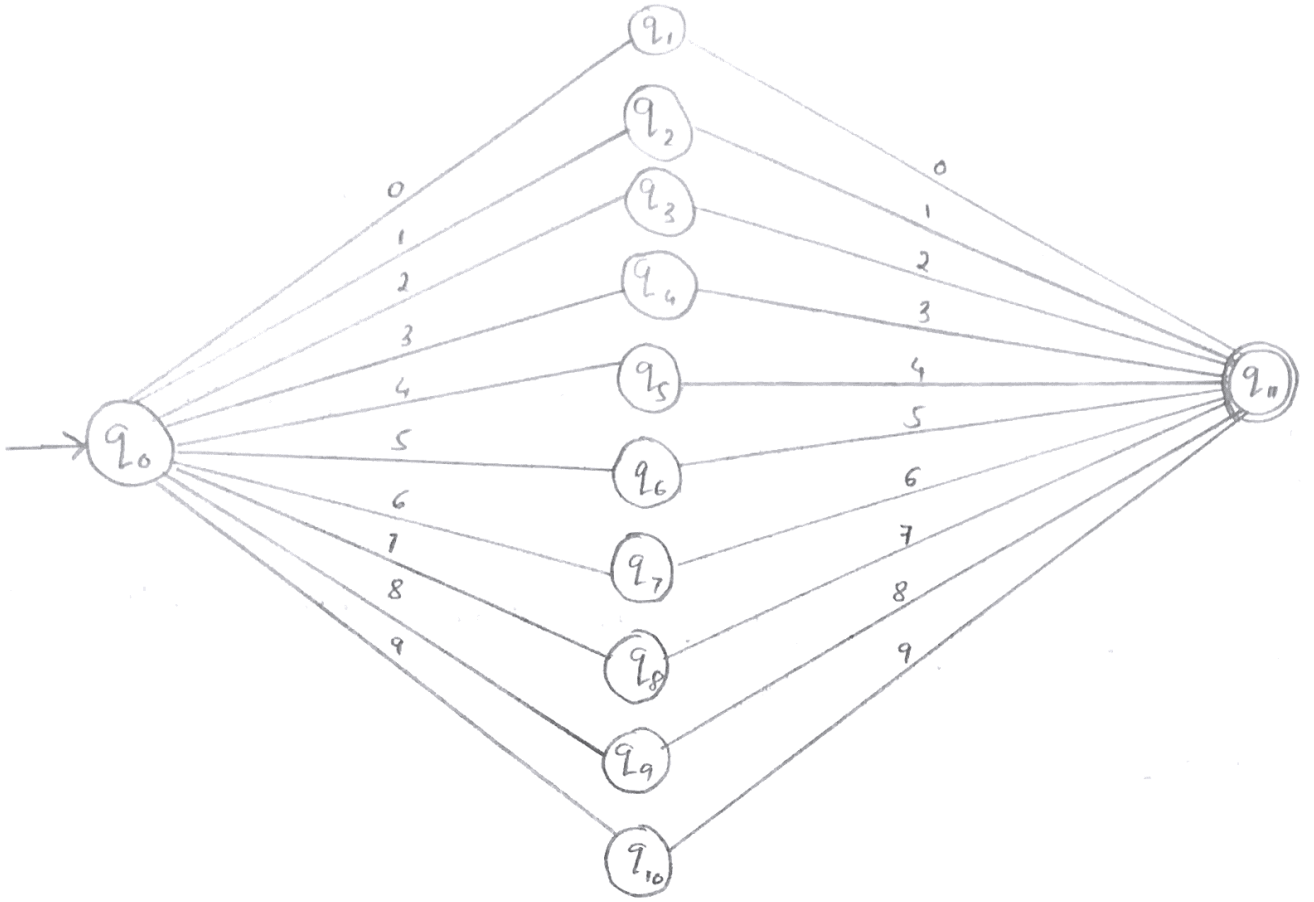
$$= \bigcup_{i=1}^{n} \delta_N(P_i, a) \quad \text{---- <iii>}$$

As equation <i> & <iii> are similar

Therefore $L(D) = L(N)$

2.3.4

(a)



(b)

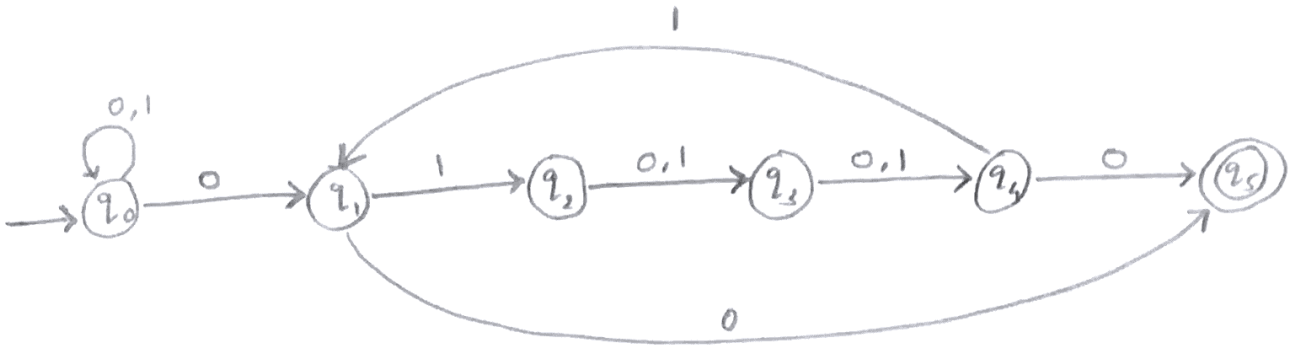
We can consider the above diagram & let the transitions be as below.

$\delta(q_0, \epsilon) = q_1$
$\delta(q_0, \epsilon) = q_2$
$\delta(q_0, \epsilon) = q_3$
⋮
$\delta(q_0, \epsilon) = q_{10}$

AND

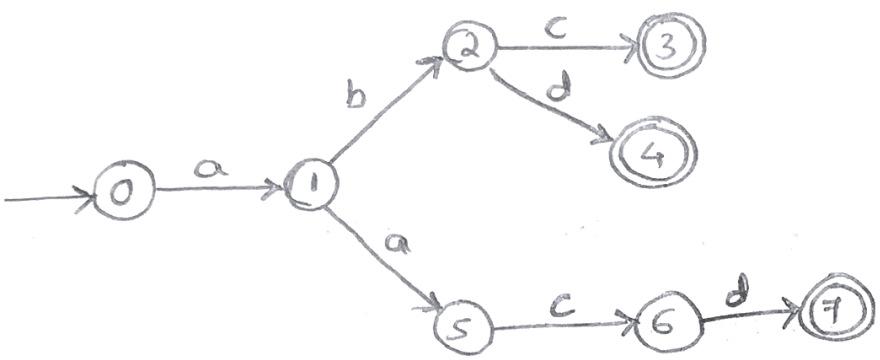
$\delta(q_1, 0) = q_{11}$
$\delta(q_2, 1) = q_{11}$
$\delta(q_3, 2) = q_{11}$
⋮
$\delta(q_{10}, 9) = q_{11}$

(c)

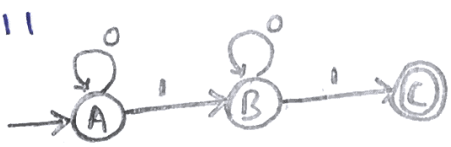


2.4.1 Design NFA to recognize following set of strings.

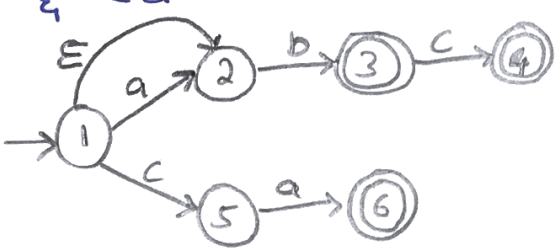
(a) abc, abd, & aacd. Assume the alphabet is {a, b, c, d}



(b) 0101, 101 & 011



(c) ab, bc & ca



2.2.9

(a) Base case: $|w| = 1$

$\hat{\delta}(q_0, w) = \delta(q_0, w)$ ∵ w is a single symbol

$\hat{\delta}$ agrees with δ on single symbol.

Induction:

Let $w = za$, so the inductive hypothesis applies to z. Then,

$$\hat{\delta}(q_0, w) = \delta(q_0, za) = \delta(\hat{\delta}(q_0, z), a)$$

$$= \delta(\hat{\delta}(q_0, z), a) = \hat{\delta}(q_0, za) = \hat{\delta}(q_0, w)$$

(b) WKT $\hat{\delta}(q_0, x) = q_f$

By part (a) WKT $\hat{\delta}(q_0, x) = q_f$

Now we have to do induction on 'k'.

& show that $\hat{\delta}(q_0, x^k) = q_f$

Base case: For $k = 1$

Induction: Assume the statement for $k-1$

i.e $\hat{\delta}(q_x, k > 0) \Rightarrow \hat{\delta}(q_x, k > k-1) = q_f$

Now,

$$\hat{\delta}(\hat{\delta}(q_0, x^{k-1}), x) = \hat{\delta}(q_0, x) = q_f$$

by inductive hypothesis.

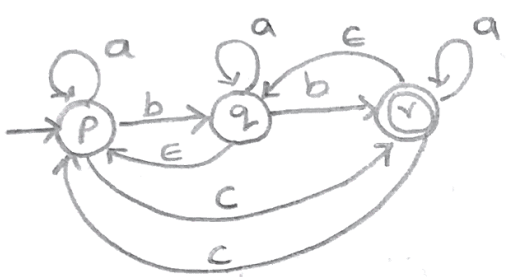
2.5.1

	ϵ	a	b	c
→ p	$\emptyset$	{p}	{q}	{r}
q	{p}	{q}	{r}	$\emptyset$
*r	{q}	{r}	$\emptyset$	{p}

(a) ϵ closure of each case

$ECLOSE(P) = \{P\}$
$ECLOSE(q) = \{P, q\}$
$ECLOSE(r) = \{P, q, r\}$

(b) All the strings of length three or less accepted by automaton.



* abb
* abc
* bb
* bba
* bc
* bca
* bab
* cab
* bbb
* ~~bc~~ c
* ~~bca~~ ac
* aac
* ccc
* ca
* caa

(C) ϵ-NFA to DFA

	a	b	c
→ P	{P}	{P,q}	{P,q,r}
{P, q}	{P,q}	{P,q,r}	{P,q,r}
{P, q r}	{P,q,r}	{P,q,r}	{P,q,r}

⇒

	a	b	c
→ A	A	B	C
B	B	C	C
*C	C	C	C

2.5.2

(a) $ECLOSE(p) = \{p, q, r\}$

$ECLOSE(q) = \{q\}$

$ECLOSE(r) = \{r\}$

(b) bc, bb, bbc, abc etc, ...

(c)

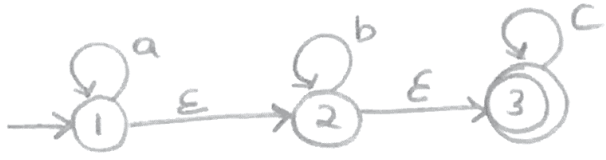
	a	b	c
→ p	{p, q, r}	{p, q}	{p, r}
* {p, q, r}	{p, q, r}	{q, r}	{q, r}
{p, q}	{p, q, r}	{q, r}	{q, r}
* {p, r}	{q, r}	{q, r}	{q, r}
* {q, r}	{p, q, r}	{r}	{q, r}
* {r}	∅	∅	∅
∅	∅	∅	∅

⇒

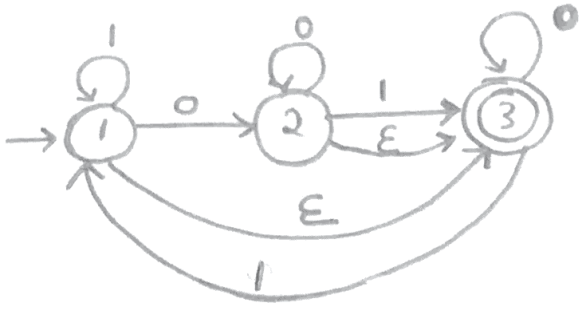
	a	b	c
A	B	C	D
B	B	E	E
C	B	E	E
D	E	E	E
E	B	F	E
F	G	G	G
G	G	G	G

2.5.3

(a)



(b)



(c)

	ϵ	0	1
$\rightarrow q_s$	$\{\emptyset\}$	$\{q_s\}$	$\{q_s, 1\}$
q_1	q_2	q_2	q_2
q_2	q_3	q_3	q_3
q_3	q_4	q_4	q_4
q_4	q_5	q_5	q_5
q_5	q_6	q_6	q_6
q_6	q_7	q_7	q_7
q_7	q_8	q_8	q_8
q_8	q_9	q_9	q_9
q_9	q_{10}	q_{10}	q_{10}
q_{10}	$\emptyset$	$\emptyset$	$\emptyset$

3.1.1 Regular expressions

(a) Atleast 1 a & atleast 1 b in {a, b, c}

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

(b) 10th symbol from the right end is 1.

$$(0+1)^*1(0+1)^9$$

(c) Set of 0's & 1's with atmost pair of consecuti -ve 1's.

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

3.1.2

(a) Every pair of adjacent 0's appears before any pair of adjacent 1's.

$$((0+1)+((00)(11))^*+(0+1))^*$$

(b) Set of 0's & 1's whose number of 0's is divisible by 5.

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

3.1.3

(a) Set of all strings 0's & 1's not containing "101"

(b) The set of all strings with an equal no of 0's & 1's such that no prefix has two ~~or~~ more 0's than 1's, nor two more 1's than 0's.

(c)

3.1.4

(a) $(1+\epsilon)(00^*1)^*0^*$

⇒ This has no adjacent 1's.
$(00^*1)^*$ says every 1 is preceded by atleast one 0. There's always a 1 in the beginning which is minimum size of the string.

(b) $(0^*1^*)^*000(0+1)^*$

The set of strings 0's & 1's where 000 is a sub string.

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

The set of string 0's & 1's where the 0's wont appear continuosly.

3.1.5

The language of the regular expression Σ' and it's closure is finite.

As Σ^* denotes → language of strings consisting of any number of empty strings.

3.2.01

(a) All the regular expressions $R_{ij}^{(0)}$ From 1 state to other state.

$R_{11}^{(0)} = \Sigma + 1$ $\quad R_{21}^{(0)} = 1$ $\quad R_{33}^{(0)} = \Sigma + 0$

$R_{12}^{(0)} = 0$ $\quad R_{22}^{(0)} = \Sigma$ $\quad R_{32}^{(0)} = 1$

$R_{13}^{(0)} = \phi$ $\quad R_{23}^{(0)} = 0$ $\quad R_{31}^{(0)} = \phi$

Same state $= \Sigma$

3.2.[redacted]1

(b) $R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)} (R_{11}^{(0)})^* R_{1j}^{(0)}$

$R_{11}^{(1)} = \varepsilon + 1 + \varepsilon + 1 (\varepsilon + 1)^* (\varepsilon + 1)$

$R_{11}^{(1)} = 1^*$

When we do this for all iterations, we get,

$R_{11}^{(1)} = 1^*$ $\quad R_{21}^{(1)} = 11^*$ $\quad R_{31}^{(1)} = \phi$

$R_{12}^{(1)} = 1^*0$ $\quad R_{22}^{(1)} = \varepsilon + 11^*0$ $\quad R_{32}^{(1)} = 1$

$R_{13}^{(1)} = \phi$ $\quad R_{23}^{(1)} = 0$ $\quad R_{33}^{(1)} = \varepsilon + 0$

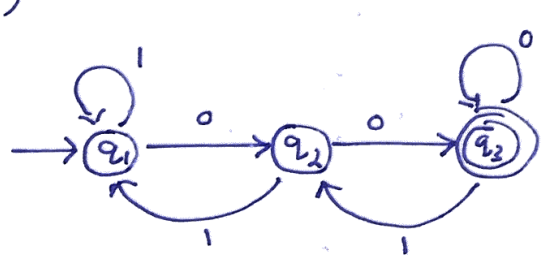
(c) $R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)} \cdot (R_{22}^{(1)})^* R_{2j}^{(1)}$

We have to follow similar steps as it's on the above example.

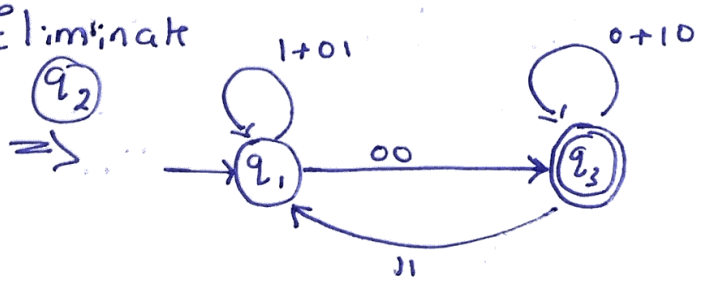
(d) Regular expression for the given automaton:

$(\varepsilon 1 + 01 + 00 (0 + 10)^* 11)^* 00 (0 + 10)^*$

(e)

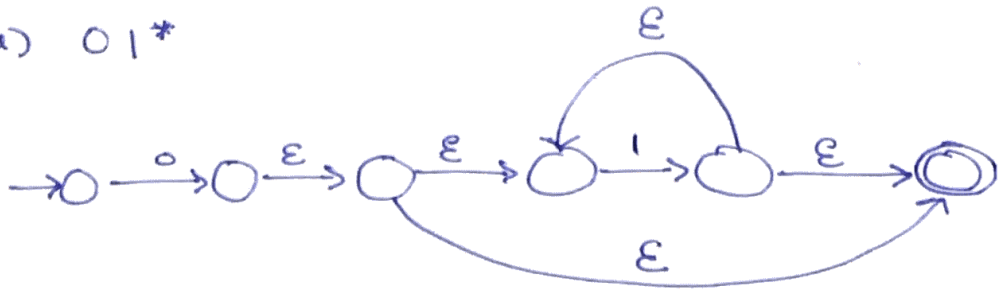


Eliminate q_2

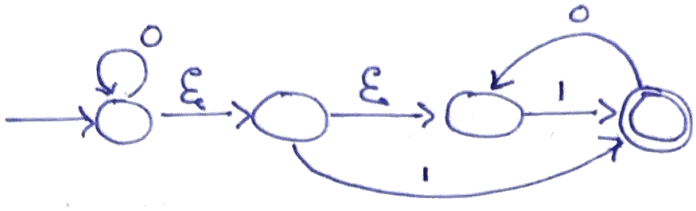


3.2.4

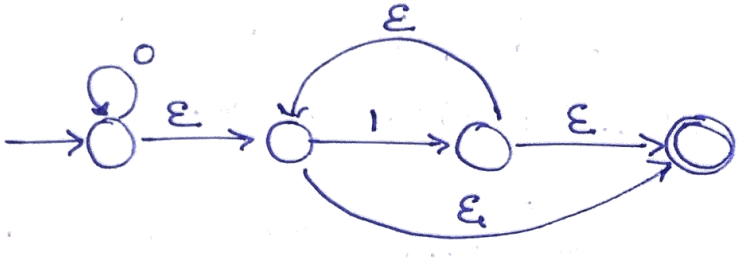
(a) 01^*



(b) $(0+1)01$

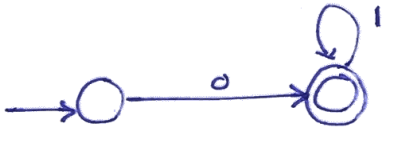


(c) $00(0+1)^*$

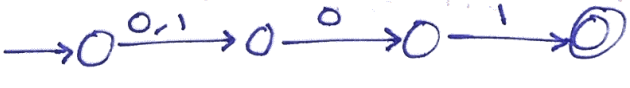


3.2.5

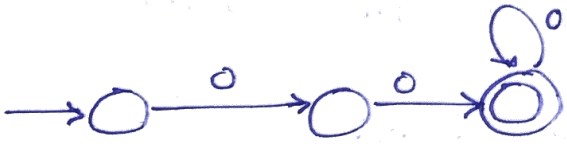
(a)



(b) $(0+1)01$



(c)



3.2.6 (a) LL^* or L^+

(b) The set of suffixes in string L

(c) The set of all states 's'.

(d) The set of all states 's' with suffixes of strings in 'L'.

3.2.8

Let $R^{(k)}_{ijm}$ be the number of paths from state i to state j of length 'm' & it doesn't go through a state higher than 'k'. We can compute these numbers for all states i & j, and for 'm' no greater than 'n'; by induction on k.

Basis: R^{0}_{ij1} is the no. of arcs from state i to j

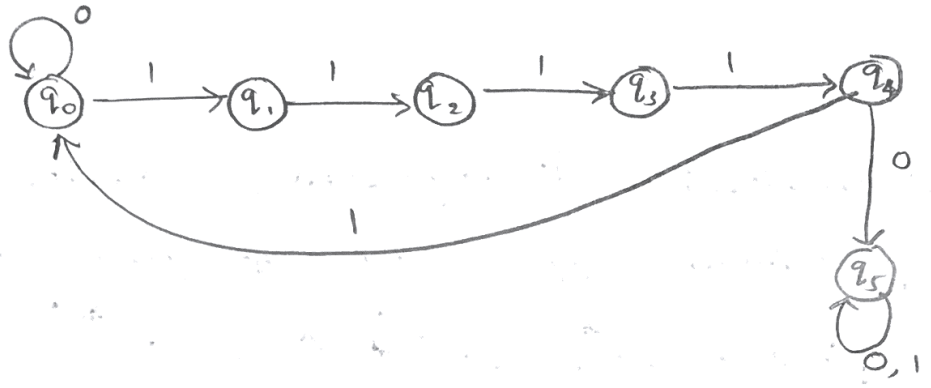
Induction: $R^{(k)}_{ijm}$ is the sum of $R^{(k-1)}_{ijm}$. Here 'r' must be atleast '2'.

The answer is $R^{(k)}_{1jn}$, where 'k' is the number of states, 1 is the start state & 'j' is any accepting state.

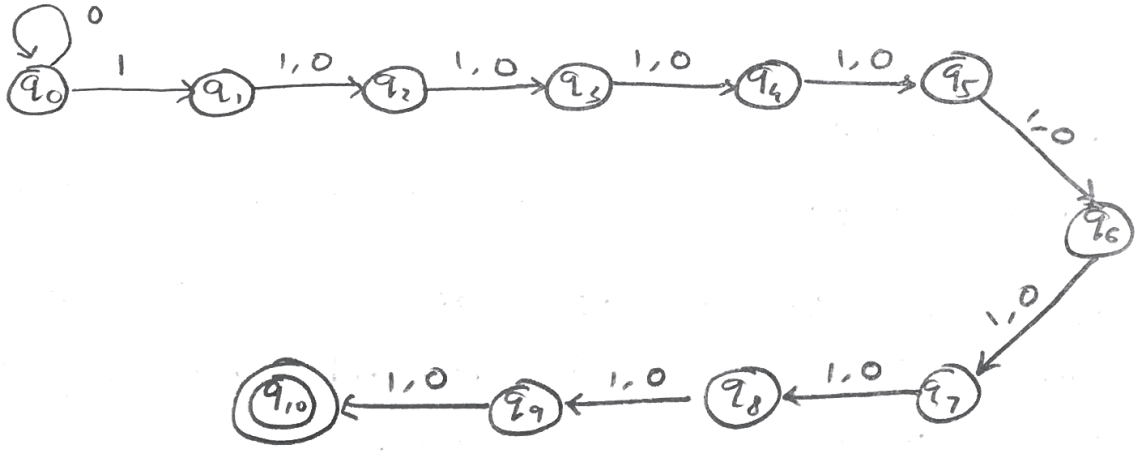
2.2.1 (c)

	A	B
000 r	110 a	001 a
110 a	010 r	111 a
001 a	111 a	010 a
010 r	100 r	011 a
111 a	011 r	100 r
010 a	100 r	011 a
100 r	000 r	101 a
011 a	101 r	000 r
011 r	101 r	000 r
101 a	001 r	110 a
101 r	001 r	110 a
001 r	111 a	010 a
110 a	010 r	111 a

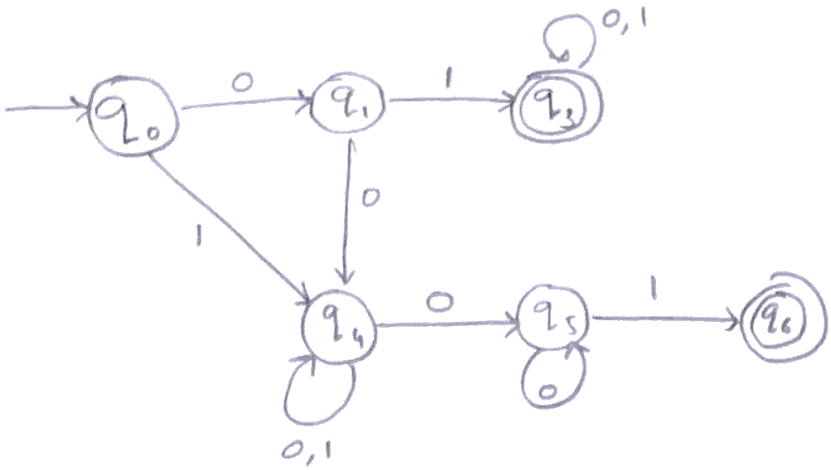
2.2.5 (a)



(b)



(c)



3.3.1

Regular expression to describe phone number.

"^\\+(?:[0-9] ?){6,14}[0-9]$";

3.3.2

Regular expression to represent salaries:
As they might appear in employment advertising:

"^\\$(?:[1-9]\d*) ([a-z][A-Z][])*"

3.4.1

(a) $R+S = S+R$

Now, replace R by {a} & S by {b}

$\{a\}+\{b\} = \{b\}+\{a\} \Rightarrow \{a,b\} = \{b,a\}$

⇒ Since, order is irrelevant in sets, both the languages are same.

3.4.1 (b) $(R+S)+T = R+(S+T)$

Let $R = \{a\}$, $S = \{b\}$ & $T = \{c\}$

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

$(a \cup b) \cup c = a \cup (b \cup c)$

$a \cup b \cup c = a \cup b \cup c$

As the order ~~of~~ ~~the~~ is irrelevant in sets, both the languages are same.

(c) $(RS)T = R(ST)$

Let $R = \{a\}$, $S = \{b\}$, $T = \{c\}$

$(ab)c = a(bc)$

$(a \cap b) \cap c = a \cap (b \cap c) \Rightarrow a \cap b \cap c = a \cap b \cap c$

$\therefore$ The above languages are same.

(d) $R(S+T) = RS + RT$

Let, $R = \{a\}$, $S = \{b\}$, $T = \{c\}$

Now we have to solve LHS

$R(S+T) \Rightarrow RS + RT$ ——(i) LHS

$RS + RT$ ——(ii) RHS.

$ab + ac = ab + ac$

Hence both the given languages are same

(e) $(R+S)^\dagger = R^\dagger + S^\dagger$

LHS $\longrightarrow$ $(R+S)^\dagger = R^\dagger + S^\dagger$ —— (i)

RHS $\longrightarrow$ $R^\dagger + S^\dagger$ —— (ii)

LHS = RHS.

(f) $(R^*)^* = R^*$

Let $R = \{a\}$

RHS will be $\{a\}^*$

where $\{a\}^* = \{\epsilon, a, aa \ldots\}$

LHS = $\{a^*\}^*$

The given string is the concatination of the a's.

Hence the given language is same

(g) $(\epsilon + R)^* = R^*$

As ϵ is a null transition it is (LHS) same as RHS.

(h) $(R^* S^*)^* = (R+S)^*$

consider $R = \{a\}$ & $S = \{b\}$

$(\{a\}^* \cap \{b\}^*)^* = (\{a\} \cup \{b\})^*$

Hence these cannot be same

The given languages isn't same.

3.4.2

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

Replace R by {a} & S by {b}

L.H.S → All the string of mixture a's & b's.

R.H.S → Only strings of a's & b's.

Hence the given regular expression is not same

(b) $(RS+R)^*R = R(SR+R)^*$

Replace R by {a} & S by {b}

L.H.S → $(ab+a)^*a$

$= abaa$

R.H.S → $a(ba+a)^* = abaa$

As the strings of a's & b's to the LHS & RHS match, the given regular expression is same.

(c) $(RS+R)^* RS = (RR^*S)^*$

Replace R by {a} & S by {b}

R.H.S → String is composed of 0 or more occurance of strings of the form a.... ab

L.H.S → Every string ends with an "ab".

Hence, the given regular expression isn't same

(d) $(R+S)^*S = (R^*S)^*$

Replace 'R' by {a} & 'S' by {b}

$(a+b)^*b \rightarrow$ LHS

Always ends with string 'b' & has atleast one 'b'.

$(a^*b)^* \rightarrow$ RHS

The string ends with 'b' but but it's not necessary that there should be atleast one 'b'.

(e) $S(RS+S)^*R = RR^*S(RR^*S)^*$

Replace 'R' by {a} & 'S' by {b}

$b(ab+b)^*a \rightarrow$ R.H.S

Here the string has 'a' & 'b' and it'll end only with a 'a'.

$aa^*b(aa^*b)^* \rightarrow$ R.H.S

Here the string ends with 'b' as it's a cumpulsion in the string.

As the below string fails to match,

bababa $\neq$ aab

The given regular expression is not same.

3.4.4

$\Rightarrow (L^* M^*)^* \subseteq (L+M)^*$ → To prove.

Replace 'L' & 'M' with a & b respectively.

L.H.S → $(a^* b^*)^*$

Has sub strings as

ab

R.H.S $(a+b)^*$ This also has strings of a & b and has multiple a's & b's.

Hence we can say that

$(L+M)^* = (L^* M^*)^*$

4.1.1

(c) $\{ 0^n 1 0^n \mid n \geq 1 \}$

Let 'n' be the pumping Lemma constant. ('n' is unrelated to n that is local variable).

Pick $w = 0^n 1 0^n$. Then we write $w = xyz$, we know that $|xy| <= n$, and therefore 'y' consists of only zeros. Then xz which must be in 'L' if 'L' is regular, consists of fewer than 'n' 0's, followed by a 1 and exactly n 0's. That string is not in 'L', so we contradict the assumption that L is ~~not~~ regular.

d) $\{0^n 1^m 2^n \mid n$ and m are arbitary integers $\}$

Assume that the language 'L' is regular, let P be the pumping lemma constant. Pick $\omega = 0^p 1 2^p$. Then we write $\omega = xyz$, wkt $|xy| \leq p$ & therefore 'y' consists of only 0's. Thus xz, which must be in L if L is regular, consists fewer than p 0's, followed by a 1 & exactly p 2's. That string is not in L, so we contradict the assumption that L is regular.

f) $\{0^n 1^{2n} \mid n \geq 1\}$

Assume the language 'L' is regular, let 'P be the pumping lemma constant. Pick $\omega = 0^p 1^{2p}$. Then we write $\omega = xyz$, wkt $|xy| \leq p$ & therefore 'y' consist of only 0's. Thus xz which must be in 'L' if 'L' is regular, consists of fewer than p 0's, followbe by exactly 2p 1's. That string is not in L, so we contradict the assumption that L is regular.

4.1.2 (a) $\{0^n \mid n$ is a perfect square$\}$

Let 'n' be a pumping lemma constant & pick $w = 0^{n^2}$, that's, n^2 0's. When we write $w = xyz$, we know that y consists of between 1 & n 0's. Thus, xyz has length between n^2+1 and n^2+n. Since the next perfect square after n^2 is $(n+1)^2 = n^2+2n+1$. We know that the length of xyz lies strictly between the consecutive perfect square n^2 and $(n+1)^2$. Thus the length of xyz cannot be a perfect square. But if the language were regular, then xyyz would be in the language, which contradicts the assumption that the language of strings of 0's, whose length is a perfect square is a regular language.

4.1.4 (a) The empty set : we cannot pick 'w' from the empty language.

(b) $\{00, 11\}$: If the adversary picks $n=3$, then we cannot pick a 'w' of length atleast 'n'.

(c) $(00+11)^*$

The adversary can pick an $n>0$, so we have to pick a non-empty w. Since 'w' must consist of pairs 00 & 11, the adversary can pick up 'y' to be one of those pairs. Then whatever i we pick, xy^iz will consist of pairs 00 & 11, and so belongs in the language.

4.2.1

(a) $h(0120) = aabbaa$

(b) $h(21120) = baababaa$

(c) $a(ab)^*ba$

(d) $a + abba$

(e) $h^{-1}(L) = \{110, 102, 022\}$

(f) $h^{-1}(L) = 02^*$

4.2.2

Let us start with a DFA 'A' for 'C'. Construct a new DFA 'B' that is exactly same as 'A', except that state 'q' is accepting state of B if & only if $\delta(q,a)$ is an accepting state of 'A'.

Then B accepts input string 'w' if and only if 'A' accepts 'wa'; that is $L(B) = L(a)$.

2.4.2 Convert each NFA to DFA from the above.

(a)

	a	b	c	d
A	B	ϕ	ϕ	ϕ
B	F	C	ϕ	ϕ
F	ϕ	ϕ	G	ϕ
C	ϕ	ϕ	D	E
G	ϕ	ϕ	ϕ	H
D	ϕ	ϕ	ϕ	ϕ
H	ϕ	ϕ	ϕ	ϕ

(b)

	0	1
A	B	F
B	ϕ	C
F	G	ϕ
C	D	I
G	ϕ	H
D	ϕ	E
I	ϕ	ϕ
H	ϕ	ϕ
E	ϕ	ϕ

(c)

	a	b	c
A	B	D	F
B	ϕ	C	ϕ
D	ϕ	ϕ	E
F	G	ϕ	ϕ
C	ϕ	ϕ	ϕ
E	ϕ	ϕ	ϕ
G	ϕ	ϕ	ϕ

3.1.3

(a) $((0+1)^* - (101))$

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

(c) $((00000)^* + (11)^*)^* + (0^* + (11)^* + (0000)^*) + ((00)^* + (11)^* + (000)^*)$

3.2.1

	0	1
$\rightarrow q_1$	q_2	q_3
q_2	q_3	q_1
$* q_3$	q_3	q_2

(a) $R_{ij}(0)$

$R_{11} = \epsilon + 1 \quad R_{12} = 0 \quad R_{13} = \phi$

$R_{21} = 1 \quad R_{22} = \epsilon \quad R_{23} = 0$

$R_{31} = \phi \quad R_{32} = 1 \quad R_{33} = \epsilon + 0$

(b) $R_{ij}(1)$

$R_{11} = 1^* \quad R_{12} = 1^*0 \quad R_{13} = \phi$

$R_{21} = 11^* \quad R_{22} = \epsilon + 11^*0 \quad R_{23} = 0$

$R_{31} = \phi \quad R_{32} = 1 \quad R_{33} = \epsilon + 0$

(c) $R_{ij}(2)$

$R_{11} = (\epsilon + 1) + 1^*0^*(\epsilon + 11^*0)11^*$

$R_{12} = 1^*0 + 1^*0(\epsilon + 11^*0).0$

$R_{21} = 11^* + (\epsilon + 11^*0)(\epsilon + 11^*0)11^*$

$R_{22} = (\epsilon + 11^*0) + (\epsilon + 11^*0)(\epsilon + 11^*0)(\epsilon + 11^*0)$

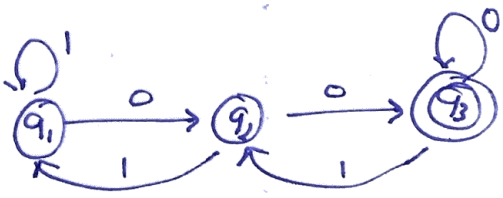
$R_{23} = 0 + (\epsilon + 11^*0)(\epsilon + 11^*0)(0)$

$R_{31} = \phi + 1(\epsilon + 11^*0)11^*$

$R_{32} = 1 + 1(\epsilon + 11^*0)(\epsilon + 11^*0)$

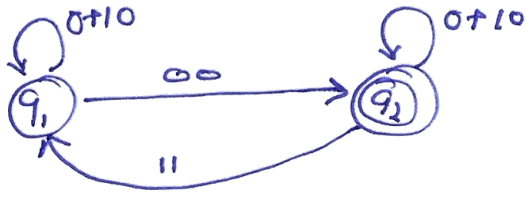
$R_{33} = (\epsilon + 0) + 1(\epsilon + 11^*0)0$

(d)



After eliminating q_2

(e)



3.2.2

(a) $R_{ij}(0)$

$R_{11} = 0$ $\quad R_{12} = \epsilon + 1$ $\quad R_{13} = \phi$

$R_{21} = \epsilon$ $\quad R_{22} = 0$ $\quad R_{23} = 1$

$R_{31} = 1$ $\quad R_{32} = \phi$ $\quad R_{33} = \epsilon + 0$

(b) $R_{ij}(1)$

$R_{11} = 1^*0$ $\quad R_{12} = 1^*$ $\quad R_{13} = \phi$

$R_{21} = 0$ $\quad R_{22} = 11^*$ $\quad R_{23} = \epsilon + 11^*0$

$R_{31} = 1$ $\quad R_{32} = \epsilon 0$ $\quad R_{33} = \phi$

(c) $R_{11} = 1^*0 + 1^*0\,(\epsilon + 11^*0).0$

$R_{12} = (\epsilon + 1) + 1^*0\,(\epsilon + 11^*0)\,11^*$

$R_{21} = (\epsilon + 11^*0) + (\epsilon + 11^*0)(\epsilon + 11^*0)(\epsilon + 11^*0)$

$R_{22} = 0 + (\epsilon + 11^*0)(\epsilon + 11^*0)(0)$

$R_{23} = 11^* + (\epsilon + 11^*0)(\epsilon + 11^*0)\,11^*$

$R_{31} = (\epsilon + 0) + 1\,(\epsilon + 11^*0)\,0$

$R_{32} = \phi + 1(\epsilon + 11^*0)\,11^*$

$R_{33} = 1 + 1(\epsilon + 11^*0)(\epsilon + 11^*0).$

3.4.3

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

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

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

3.4.5

Proof: proof is a structural induction on the expression.

Let $\epsilon = \epsilon$ or ϕ

when $\epsilon = \epsilon$ it's already concrete expression

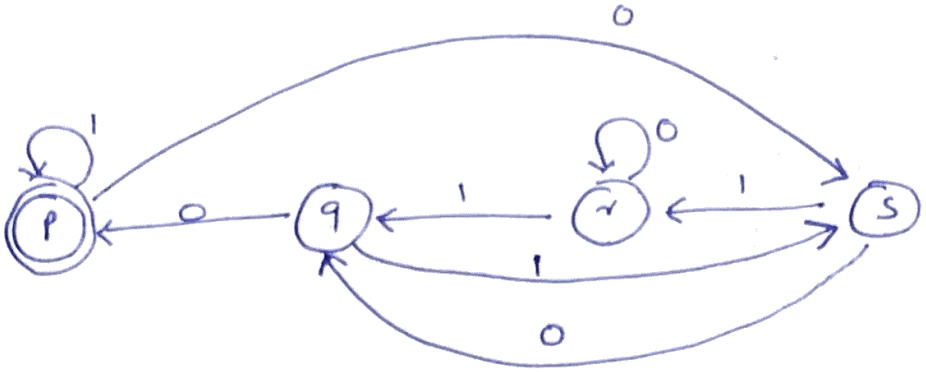
$\epsilon = \phi$ it's already concrete expression

Let $L(\epsilon) = L$, $L(c) = \{a\}$ for some symbol.

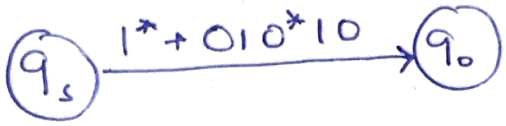
where $L(DD) = L(c)\,L(D)$ & F^* the concrete expression $L(c^*) = L(F^*)$

i.e $L(cD)$ it has to be in $L(c)$ & $L(D)$.

3.2.3



After state elimination method,



3.2.7

(i) Subsets are not important for union sets. Instead of making a new start state or accepting state, we can combine both the start states & it's transition. In the same way we can combine both the accepting states too.

(ii) For concatination, instead of combining the start state & accepting states of the first automaton with the second automaton. We can use an ϵ-transition from accepting state of the first to the second automaton.

(iii) for closure property we can actually combine the first state, i.e the accepting state of the first automaton with the start state of the second automaton

4.2.5

(b) We shall use D_a for the "derivative wrt to a". The key observation is that if epsilon is not in $L(R)$, then the derivative of RS will always remove an 'a' from the portion of a string that comes from R. If 'ϵ' is in $L(R)$, then the string might have nothing from R. & will remove 'a' from the beginning of a string in $L(S)$ (which is also a string in $L(RS)$. Thus the rule we want is:

If ϵ is not in $L(R)$, then $D_a(RS) = (D_a(R))S$

Otherwise $D_a(RS) = D_a(R)S + D_a(S)$

(e) L may have no strings that begin with '0'

4.2.8

Let A be a DFA for 'L'. We construct DFA B for half(L). The state of 'B' is of the form [q, S] where.

→ q is the state A.

→ S is the set of states of A.

To complete the construction of 'B', we have only to specify

→ The initial state is $[q_0, F]$

→ The accepting state of B are those states [q, S] such that q is in S.