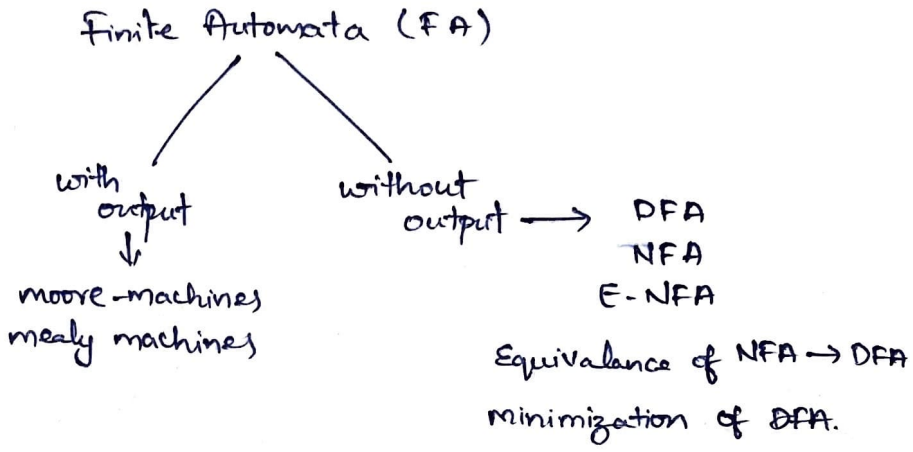
24/12/22



Deterministic Finite Automata (DFA)

Problem → understand / analyze

→ Generate (or) produce languages.

→ Design Machine.

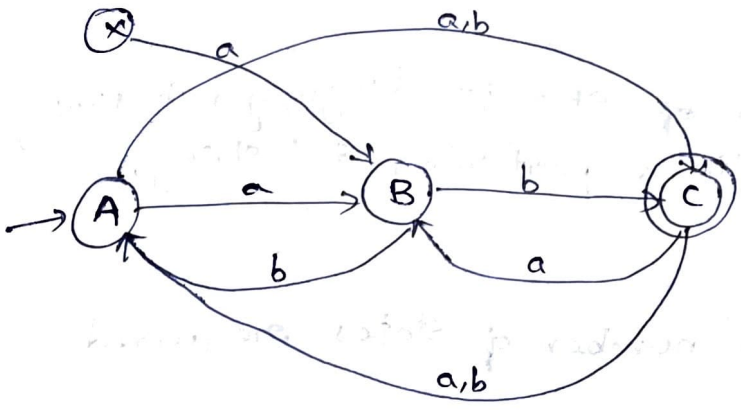
→ State Transition Table.

→ Transitions

5-tuples

DFA : $(Q, \Sigma, \delta, q_0, f)$

- Q : no. of states that are used in any machine.
- Σ : no. of inputs used to generate any language.
- δ : (no. of) transitions
- q_0 : Initial state
- f : Final state



abc

← NFA but not DFA

$Q = 4$

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

$\delta = ?$

$q_0 = A$

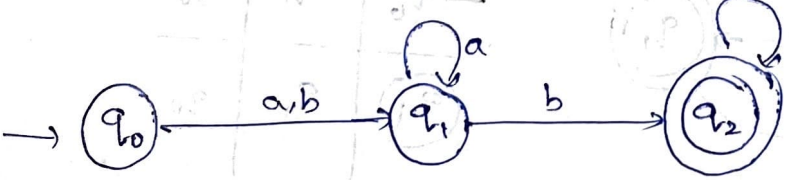
$f = C$

Note: * q_0 (initial state) is denoted based on the arrowmark such that there should not be any state before that arrow mark.

*f (final state) is defined using double circle symbol.

DFA example:

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



DFA rules :

→ Here, the construction of DFA for languages consisting of strings ending with a particular substring.

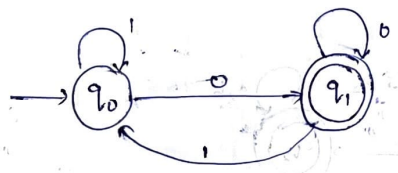
Rules:

* Determine the minimum number of states required in the DFA.
* Minimum number of states in DFA = $\boxed{n+1}$
* All strings ending with 'n' length substring will always require n+1 states.
* Always prefer to use the existing path.
* Create a new path only when there exists no path to go with.

Problems

→ Draw a DFA for the language accepting strings ending with '0' over input Alphabets $\Sigma = \{0,1\}$

Sol:



Q/Σ	0	1
→q_0	q_1	q_0
(q_1)	q_1	q_0

$\delta(q_0,0) = q_1$

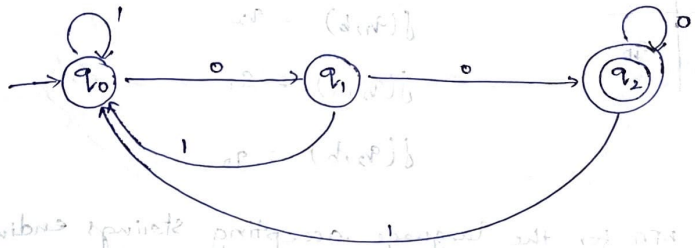
$\delta(q_0,1) = q_0$

$\delta(q_1,0) = q_1$

$\delta(q_1,1) = q_0$

→ Draw a DFA for the language accepting strings ending with '00'

Sol:- $\Sigma = \{0,1\}$



Q/Σ	0	1
→q_0	q_1	q_0
q_1	q_2	q_0
(q_2)	q_2	q_0

$\delta(q_0,0) = q_1$

$\delta(q_0,1) = q_0$

$\delta(q_1,0) = q_2$

$\delta(q_1,1) = q_0$

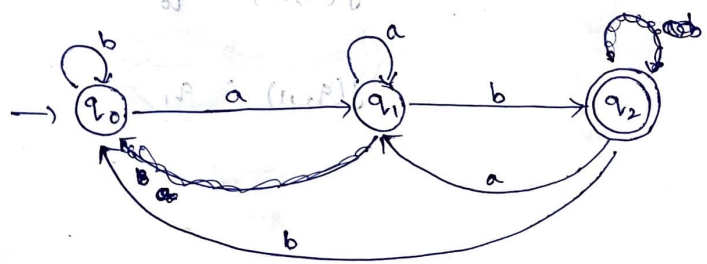
$\delta(q_2,0) = q_2$

$\delta(q_2,1) = q_0$

→ Draw a DFA for the language accepting strings ending with 'ab'

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

Sol:-



Q/Σ	a	b
→q_0	q_1	q_0
q_1	q_1	q_2
(*q_2)	q_1	q_0

$\delta(q_0,a) = q_1$

$\delta(q_0,b) = q_0$

$\delta(q_1,a) = q_1$

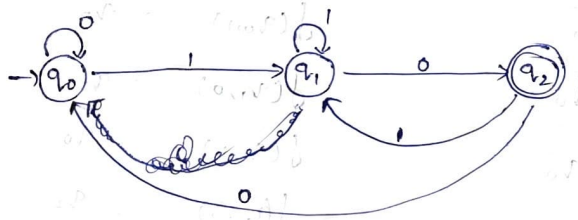
$\delta(q_1,b) = q_2$

$\delta(q_2,a) = q_1$

$\delta(q_2,b) = q_0$

→ Draw a DFA for the language accepting strings ending with '10'. over input $\Sigma = \{0,1\}$

Sol:



Q/Σ	0	1
→q_0	q_0	q_1
q_1	q_2	q_1
(*q_2)	q_0	q_1

$\delta(q_0,0) = q_0$

$\delta(q_0,1) = q_1$

$\delta(q_1,0) = q_2$

$\delta(q_1,1) = q_1$

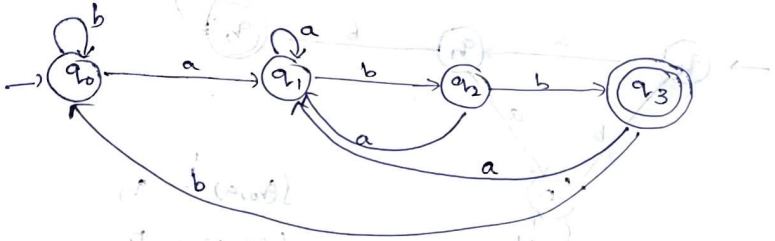
$\delta(q_2,0) = q_0$

$\delta(q_2,1) = q_1$

→ Draw a DFA for the language accepting strings ending with 'abb' over input $\Sigma = \{a,b\}$.

Sol:

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



Q/Σ	0	1
→q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
(*q_3)	q_1	q_0

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

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

$\delta(q_2,0) = q_1$, $\delta(q_2,1) = q_3$

$\delta(q_3,0) = q_1$, $\delta(q_3,1) = q_0$

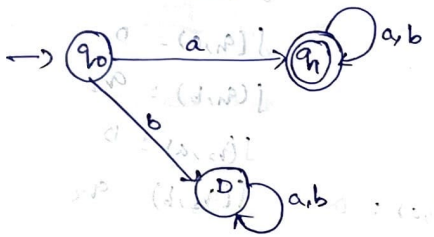
31/12/22

DFA "Starting-with" Problems

* Minimum number of states = n+2.

→ Draw a DFA for language accepting strings starting with 'a' over input $\Sigma = \{a,b\}$

Sol:



Q/Σ	a	b
→q_0	q_1	D
(*q_1)	q_1	q_1
D	D	D

$\delta(q_0,a) = q_1$

$\delta(q_0,b) = D$

$\delta(q_1,a) = q_1$

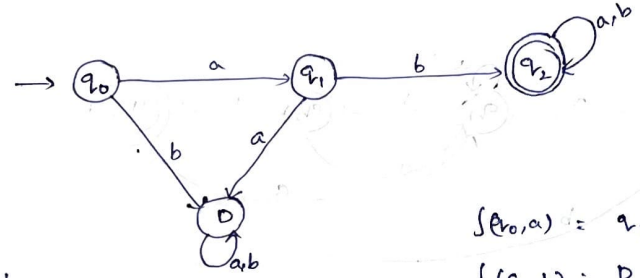
$\delta(q_1,b) = q_1$

$\delta(D,a) = D$

$\delta(D,b) = D$

→ Draw a DFA for the languages accepting some strings starting with 'ab' over input $\Sigma = \{a, b\}$

Sol:

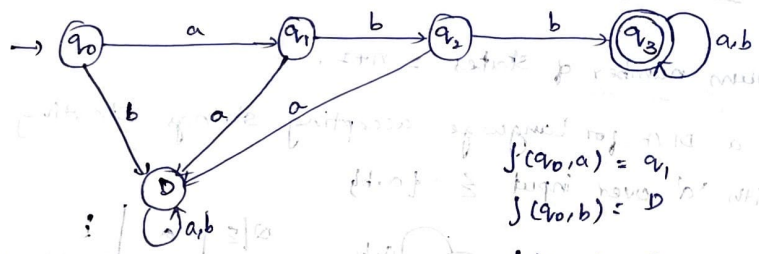


Q/Σ	a	b
→q_0	q_1	D
q_1	D	q_2
(q_2)	q_2	q_2
D	D	D

$\delta(q_0, a) = q_1$
$\delta(q_0, b) = D$
$\delta(q_1, a) = D$
$\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_2$
$\delta(q_2, b) = q_2$
$\delta(D, a) = D$
$\delta(D, b) = D$

→ Draw a DFA for the languages accepting some strings starting with 'abb' over input $\Sigma = \{a, b\}$

Sol:



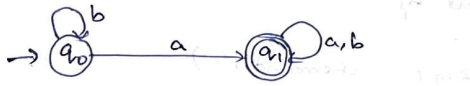
Q/Σ	a	b
→q_0	q_1	D
q_1	D	q_2
q_2	D	q_3
(q_3)	q_3	q_3
D	D	D

$\delta(q_0, a) = q_1$
$\delta(q_0, b) = D$
$\delta(q_1, a) = D$
$\delta(q_1, b) = q_2$
$\delta(q_2, a) = D$
$\delta(q_2, b) = q_3$
$\delta(q_3, a) = q_3$
$\delta(q_3, b) = q_3$
$\delta(D, a) = D$
$\delta(D, b) = D$

DFA "Contains" Problems

→ Draw a DFA for the languages accepting some strings contains 'a' over input $\Sigma = \{a, b\}$

Sol:

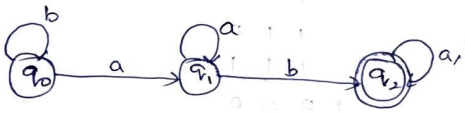


Q/Σ	a	b
→q_0	q_1	q_0
(q_1)	q_1	q_1

$\delta(q_0, a) = q_1$
$\delta(q_0, b) = q_0$
$\delta(q_1, a) = q_1$
$\delta(q_1, b) = q_1$

→ Draw a DFA for Languages accepting some strings containing 'ab' over input $\Sigma = \{a, b\}$

Sol:



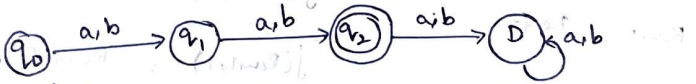
Q/Σ	a	b
→q_0	q_1	q_0
q_1	q_1	q_2
(q_2)	q_2	q_2

$\delta(q_0, a) = q_1$
$\delta(q_0, b) = q_0$
$\delta(q_1, a) = q_1$
$\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_2$
$\delta(q_2, b) = q_2$

DFA "Length" Problems

→ Draw a DFA for the languages accepting some strings ~~contains~~ having length = 2 over input $\Sigma = \{a, b\}$

Sol: $L = \{ab, ba, aa, bb\}$



Q/Σ	a	b
→q_0	q_1	q_1
q_1	q_2	q_2
(q_2)	D	D
D	D	D

$\delta(q_0, a) = q_1$
$\delta(q_0, b) = q_1$
$\delta(q_1, a) = q_2$
$\delta(q_1, b) = q_2$
$\delta(q_2, a) = D$
$\delta(q_2, b) = D$
$\delta(D, a) = D$
$\delta(D, b) = D$

DFA "divisibility" problems

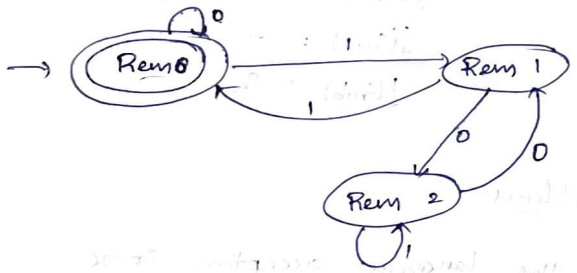
→ Draw a DFA for the languages accepting some strings divisible by '3'

Sol:- [0, 3, 6, 9] → Rem 0 ⇒ (Remainder '0')

[1, 4, 7, 10] → Rem 1

[2, 5, 8] → Rem 2

	Remainder	Binary Format
0	0	0
1	1	0 1
2	2	1 0
3	0	1 1
4	1	1 0 0
5	2	1 0 1
6	0	1 1 0
7	1	1 1 1
8	2	1 0 0 0
9	0	1 0 0 1
10	1	1 0 1 0



Q/Σ	0	1
Rem 0	Rem0	Rem1
Rem 1	Rem 2	Rem0
Rem 2	Rem1	Rem2

δ(Rem0, 0) = Rem 0

δ(Rem0, 1) = Rem 1

δ(Rem1, 0) = Rem2

δ(Rem1, 1) = Rem 0

δ(Rem2, 0) = Rem 1

δ(Rem2, 1) = Rem 2

NFA :- Non-deterministic Finite Automata

→ 5 tuples : $(Q, \Sigma, \delta, q_0, f)$

Abbreviations of these are same as the DFA.

DFA : $Q \times \Sigma \Rightarrow Q$

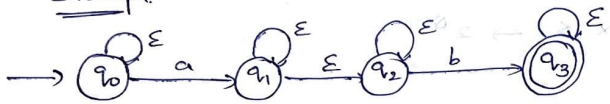
NFA : $Q \times \Sigma \Rightarrow 2^Q$

E-NFA : epsilon Non-deterministic Finite Automata

ε-NFA : "Epsilon" non-deterministic finite Automata

→ If you want to move from one state to another state without reading any value / variable / Element we use ε-NFA.

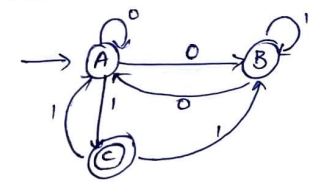
Example:



ε : Epsilon : This helps in moving from one state to another state.

NFA → DFA :

NFA:



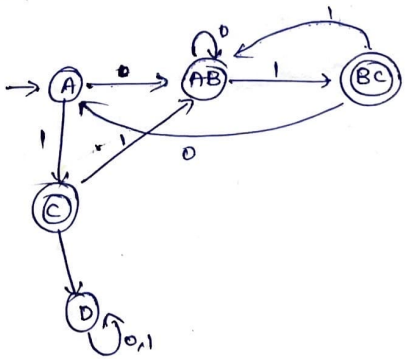
State Transition Table for NFA :

Q/Σ	0	1
→ A	A, B	C
B	A	B
C	— or ϕ	A, B

ϕ : denotes (or) represents empty.

STT for DFA using STT of NFA:

Q/Σ	0	1
A	AB	C
AB	AB	BC
BC	A	AB
C	ϕ	AB
ϕ	D	D



$\delta(A,0) = AB$ $\quad \delta(C,0) = D$ $\quad \delta(D,0) = D$

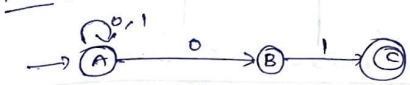
$\delta(A,1) = C$ $\quad \delta(C,1) = AB$ $\quad \delta(D,1) = D$

$\delta(AB,1) = BC$ $\quad \delta(BC,0) = A$

$\delta(AB,0) = AB$ $\quad \delta(BC,1) = AB$

Example-2:

NFA:

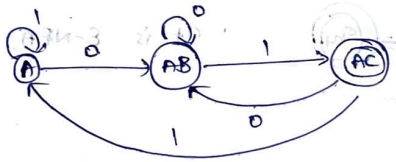


STT of NFA

Q/Σ	0	1
→ A	AB	A
B	ϕ	C
C	ϕ	ϕ

STT of DFA using NFA

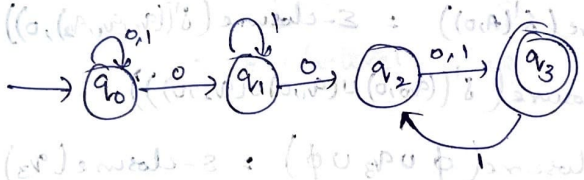
Q/Σ	0	1
→ A	AB	A
AB	AB	AC
AC	AB	A
B	ϕ	C
C	ϕ	ϕ



$\delta(A,0) = AB$ $\quad \delta(AB,0) = AB$ $\quad \delta(AC,0) = AB$

$\delta(A,1) = A$ $\quad \delta(AB,1) = AC$ $\quad \delta(AC,1) = A$

Example-3



STT of NFA:

Q/Σ	0	1
→ q_0	$q_0 q_1$	q_0
q_1	q_2	q_1
q_2	q_3	q_3
(q_3)	ϕ	q_3

STT of DFA using NFA

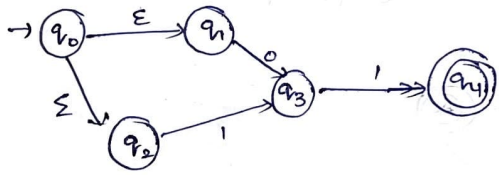
Q/Σ	0	1
→ q_0	$q_0 q_1$	q_0
$q_0 q_1$	$q_0 q_1 q_2$	$q_0 q_1$
$q_0 q_1 q_2$	$q_0 q_1 q_2 q_3$	

7/1/23

ε-closure Theorem

→ It is helpful for converting ε-NFA to DFA.

Ex:



: It is ε-NFA.

ε-closure (q_0) = $\{q_0, q_1, q_2\}$ [∵ q_0 has 2 epsilon transitions (q_1, q_2)] → A

ε-closure (q_1) = $\{q_1\}$

ε-closure (q_2) = $\{q_2\}$

ε-closure (q_3) = $\{q_3\}$

ε-closure (q_4) = $\{q_4\}$

$\delta'(A,0) = \varepsilon\text{-closure}(\delta'(A,0)) = \varepsilon\text{-closure}(\delta'((q_0,q_1,q_2),0))$

$= \varepsilon\text{-closure}(\delta'((q_0,0) \cup (q_1,0) \cup (q_2,0)))$

$= \varepsilon\text{-closure}(\phi \cup q_3 \cup \phi) = \varepsilon\text{-closure}(q_3)$

$= \boxed{q_3} \rightarrow B$

$\delta'(A,1) = \varepsilon\text{-closure}(\delta'(A,1))$

$= \varepsilon\text{-closure}(\delta'((q_0,q_1,q_2),1))$

$= \varepsilon\text{-closure}(\delta'((q_0,1) \cup (q_1,1) \cup (q_2,1)))$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup q_3)$

$= \varepsilon\text{-closure}(q_3) = \boxed{q_3}$

$\delta'(B,0) = \varepsilon\text{-closure}(\delta'(B,0))$

$= \varepsilon\text{-closure}(\delta'(q_3,0))$

$= \varepsilon\text{-closure}(\phi)$

$= \phi$

$\delta'(B,1) = \varepsilon\text{-closure}(\delta'(B,1))$

$= \varepsilon\text{-closure}(\delta'(q_3,1))$

$= \varepsilon\text{-closure}(q_4)$

$= \boxed{q_4} \rightarrow C$ (If the resultant state is new state then name that state, if the state is previously occured then leave it as it is because we previously named it).

$\delta'(C,0) = \varepsilon\text{-closure}(\delta'(C,0))$

$= \varepsilon\text{-closure}(\delta'(q_4,0))$

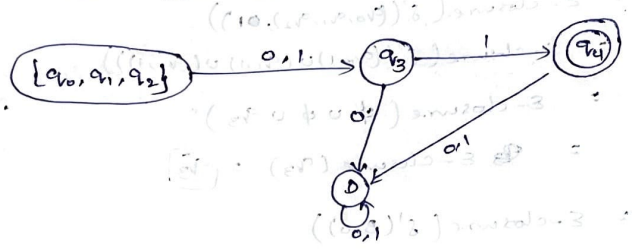
$= \varepsilon\text{-closure}(\phi) = \phi$

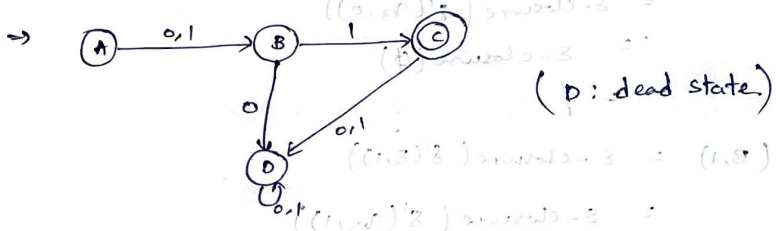
$\delta'(C,1) = \varepsilon\text{-closure}(\delta'(C,1))$

$= \varepsilon\text{-closure}(\delta'(q_4,1))$

$= \varepsilon\text{-closure}(\phi) = \phi$

Final required DFA Machine :-



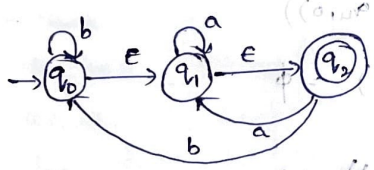


(D : dead state)

STT Table:

Q/Σ	0	1
→ A	B	B
B	D	C
©	D	D
D	D	D

Example-2 :



ε-closure (q_0) = $\{q_0, q_1, q_2\}$ → A

ε-closure (q_1) = $\{q_1, q_2\}$

ε-closure (q_2) = $\{q_2\}$

$\delta'(A,a)$ = ε-closure $(\delta'(q_0, a))$

= ε-closure (ϕ) = ϕ

$\delta'(A,b)$ = ε-closure $(\delta'(q_0, b))$

= ε-closure (q_0) = q_0

$\delta'(B,a)$ = ε-closure $(\delta'(q_1, a))$

= ε-closure $(\delta'(q_1))$

$\delta'(A,a)$ = ε-closure $(\delta'(q_0, a))$

= ε-closure $(\delta'((q_0,q_1,q_2), a))$

= ε-closure $(\delta'((q_0,a) \cup (q_1,a) \cup (q_2,a)))$

= ε-closure $(\phi \cup q_1 \cup q_1)$

= ε-closure (q_1)

= q_1, q_2 → B

~~$\delta'(B,a)$ = ε-closure $(\delta'($~~

$\delta'(A,b)$ = ε-closure $(\delta'(q_0, b))$

= ε-closure $(\delta'((q_0,q_1,q_2), b))$

= ε-closure $(\delta'((q_0,b) \cup (q_1,b) \cup (q_2,b)))$

= ε-closure $(q_0 \cup \phi \cup q_0)$

= q_0

$\delta'(B,a)$ = ε-closure $(\delta'(B,a))$

= ε-closure $(\delta'((q_1,q_2), a))$

= ε-closure $(\delta'((q_1,a) \cup (q_2,a)))$

= ε-closure $(q_1 \cup q_1)$ = ε-closure (q_1)

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

$\delta'(B,b) = \varepsilon\text{-closure}(\delta(q_1,b))$

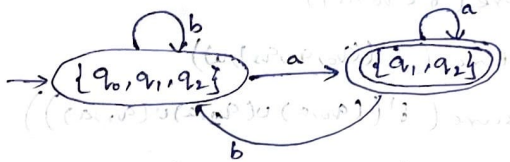
$= \varepsilon\text{-closure}(\delta'((q_1,q_2),b))$

$= \varepsilon\text{-closure}(\delta'(q_1,b) \cup (q_2,b))$

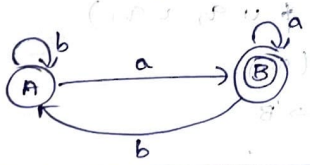
$= \varepsilon\text{-closure}(\phi \cup q_0)$

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

∴ Final required DFA :



⇒



Minimization of DFA :

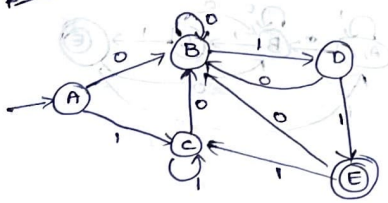
(1) Equivalence theorem.

(2) Table filling method (Myhill-Nirod theorem).

(1) Equivalence Theorem :

→ We have to search for un-reachable state and remove that state, remove all transitions from that state.

Problem :



0 - Equivalence :

{A, B, C, D} {E}

1- Equivalence :

{A, B, C} {D} {E}

2- Equivalence :

{A, C} {B} {D} {E}

3- Equivalence :

{A, C} {B} {D} {E}

3-Equivalence = 2-Equivalence

STT (for given problem) :

Q/Σ	0	1
→A	B	C
B	B	D
C	B	E
D	B	E
(E)	B	C

Comparision for 1 Equivalence

$\delta(A,0) = B$, $\delta(B,0) = B$ | $\delta(A,1) = C$, $\delta(B,1) = D$

A = B

$\delta(A,0) = B$, $\delta(C,0) = B$ | $\delta(A,1) = C$, $\delta(C,1) = C$

A = B = C

$\delta(A,0) = B$, $\delta(D,0) = B$ | $\delta(A,1) = C$, $\delta(D,1) = E$

∴ A ≠ D, B ≠ D, C ≠ D

$\delta(A,0) = B$, $\delta(B,0) = B$ | $\delta(A,1) = C$, $\delta(B,1) = D$

A ≠ B

$\delta(A,0) = B$, $\delta(C,0) = B$ | $\delta(A,1) = C$, $\delta(C,1) = C$

A = C

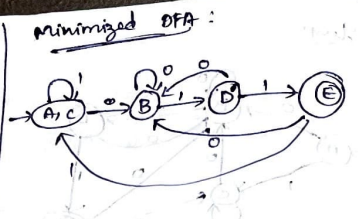
$\delta(A,0) = B$, $\delta(C,0) = B$ | $\delta(A,1) = C$, $\delta(C,1) = C$

A = C

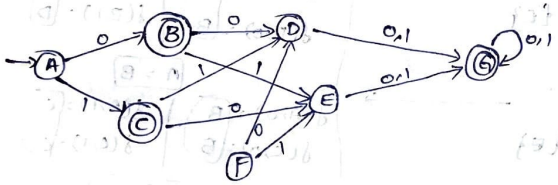
STT for (minimized dfa)

Q/Σ	0	1
AC	B	AC
B	B	D
D	B	E
E	B	AC

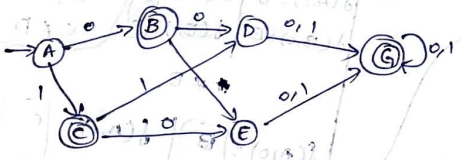
Minimized DFA :



Problem-2:



Sol: Here 'F' doesn't have any path from initial state (A) hence remove state F and all transitions of F.



STT Table

Q/Σ	0	1
A	B	C
(B)	D	E
(C)	E	D
D	G	G
E	G	G
(G)	G	G

0- Equivalence

{A, D, E} {B, C, G}

δ(A,0) = B | δ(A,1) = C
δ(B,0) = G | δ(B,1) = G

A ≠ D

δ(A,0) = B | δ(A,1) = C
δ(E,0) = G | δ(E,1) = G

A ≠ E ≠ D

δ(B,0) = D | δ(B,1) = E
δ(C,0) = E | δ(C,1) = D

B ≠ C

δ(B,0) = D | δ(B,1) = E
δ(G,0) = G | δ(G,1) = G

B ≠ G

∵ D, G are from different group and E, G are from different group

1 - Equivalence

{A, D, E} {B, C} {G}

δ(B,0) = D | δ(B,1) = E
δ(C,0) = E | δ(C,1) = D

B = C

δ(A,0) = B | δ(A,1) = C
δ(D,0) = G | δ(D,1) = G

A ≠ D

δ(A,0) = B | δ(A,1) = C
δ(E,0) = G | δ(E,1) = G

A ≠ E

But D = E.

2 - Equivalence

{A} {D, E} {B, C} {G}

δ(B,0) = D | δ(B,1) = E
δ(C,0) = E | δ(C,1) = D

B ≠ C

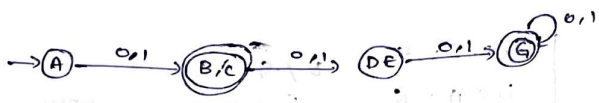
D = E

3 - Equivalence

{A} {D, E} {B, C} {G}

Finally 2-Equivalence = 3-Equivalence.

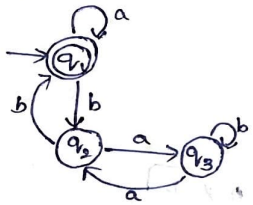
∴ Required minimized DFA :

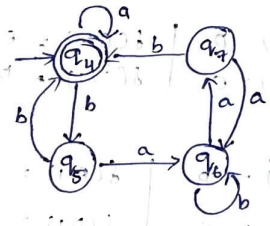


21/1/23

Equivalence of 2 FA

Ex-1 :





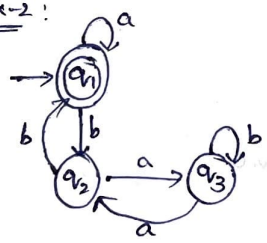
Fs : Final state

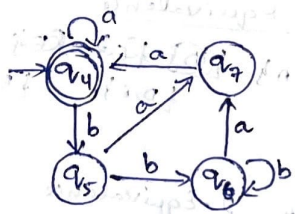
NFs : Non-Final state

pairs :	a	b
(q_1, q_4)	(q_1, q_4) Fs Fs	(q_2, q_5) NFS NFS
(q_2, q_5)	(q_3, q_6) NFs NFS	(q_1, q_4) Fs Fs
(q_3, q_6)	(q_2, q_7) NFS NFS	(q_3, q_6) NFS NFS
New state occured : (q_2, q_7)		
(q_2, q_7)	(q_3, q_6) NFS NFS	(q_1, q_4) Fs Fs

∴ The 2 Given Machines are Equal.

Ex-2 :





Ex-3 :

	a	b
pairs : (q_1, q_4)	(q_1, q_4) Fs Fs	(q_2, q_5) NFS NFS
(q_2, q_5)	(q_3, q_7) NFS NFS	(q_1, q_6) NFS NFS — Not equal
(q_3, q_6)	(q_2, q_7)	

∴ The 2 machines are not equal.

Regular Expressions

→ Introduction.
→ Operations.
→ Examples.

RE : Regular Expression.
FA : Finite Automata.

Conversion : RE → FA
FA → RE
by [1. State Elimination Method
2. Arden's theorem.]

Union : + | or

Concatenation : . o and

closure : *
↳ Kleen closure : *
zeros closure : +

Example :

$\Sigma = \{a\}$

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

$a^+ = \{a, aa, aaa, aaaa, \ldots\}$

ε : Empty string

Pb → Create a regular expression which is accepting some strings which are started with 'a' over $\Sigma = \{a, b\}$

Sol: $L = \{a, aa, aaa, ab, aab, aabb, aaab, aaabb, \ldots\}$

$RL = a(a+b)^*$ → starts with a

$(a+b)^* a$ → Ends with a

$a(a+b)^* a$ → starts with a & Ends with a.

Examples:

$a(a+b)^* b$ → starting & Ending must have different input.

$b(a+b)^* a$ →

$(a+b)^* a (a+b)^*$ → Contains a.

Pb → Create a regular expression which accepts some strings with inputs $\Sigma = \{a, b\}$ whose string length = 2.

Sol: $L = \{aa, ab, ba, bb\}$

$RL = aa + ab + ba + bb$

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

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

Pb → Create a regular expression which should accept some strings whose length is atleast 2 with input $\Sigma = \{a, b\}$.

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

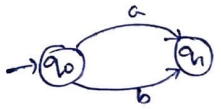
Note:

ϕ (empty) : → q_0

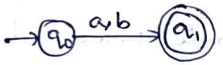
ϵ (epsilon) : → q_0 —ε→ q_1

(or) → q_0 with ε self-loop

Examples:

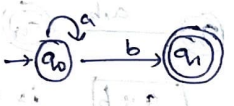


a+b

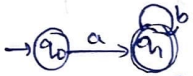


a^* :

a^*b :

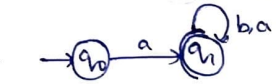


ab^* :



ab^*a :

ab^*a^* :



Conversion of Any Finite Automata into Regular Expressions

i, State Elimination Method

ii, Arden's Theorem.

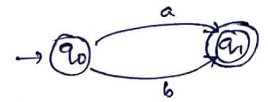
State Elimination Method:

→ Your initial state should not contain any incoming paths.

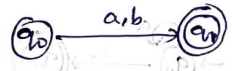
→ Your final state should not contain any outgoing paths.

→ Before writing Regular Expression you should maintain only one final state.

Ex: Given:

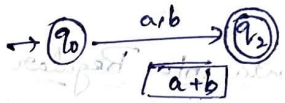


eliminate unwanted transitions!



∴ $\boxed{a+b}$ is the Regular Expression.

Ex:



Ex:

ab^*c (here no elimination is required).

Identities

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

Ex: Arden's Theorem:-

If P & Q are the 2 regular languages

$$R = Q + RP$$

$$= Q + (Q+RP)P = Q + QP + RP^2$$

$$= Q + QP + (Q+RP)P^2$$

$$= Q + QP + QP^2 + RP^3$$

$$= Q + QP + QP^2 + (Q+RP)P^3$$

$$= Q + QP + QP^2 + QP^3 + RP^4$$

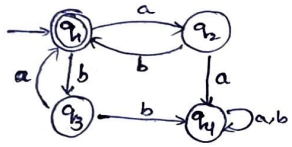
$$\vdots$$

$$= Q + QP + QP^2 + QP^3 + \dots + QP^n + RP^{n+1}$$

$$= Q(\varepsilon + P + P^2 + P^3 + \dots + P^n) = QP^*$$

∴ $\boxed{R = QP^*}$

Ex-2 :



Equations :

$q_1 = \varepsilon + q_3a + q_2b$ —①

$q_2 = q_1a$ —②

$q_3 = q_1b$ —③

$q_4 = q_2a + q_3b + q_4a + q_4b$ —④

Note : while writing the equations for the states, if the state is initial state then include epsilon and search for the incoming transitions to the state and write the equation.

Now, $q_1 = \varepsilon + (q_1b)a + (q_1a)b$ [sub ②, ③ in ①].

$q_1 = \varepsilon + q_1ba + q_1ab$

$q_1 = \varepsilon + q_1(ba + ab)$

($R = Q + RP$)

$\Rightarrow q_1 = \varepsilon(ba+ab)^*$

$q_1 = (ba+ab)^*$ [$\because \varepsilon R^* = R^*$].

Ex-3 :

Equations :

$q_1 = \varepsilon + q_10$ —①

$q_2 = q_11 + q_21$ —②

$q_3 = q_20 + q_30 + q_31$ —③

$q_1 = \varepsilon + q_10$

($R = Q + RP$) $\Rightarrow$ $R = QP^*$

$q_1 = \varepsilon \cdot 0^* \Rightarrow q_1 = 0^*$ —④

$q_2 = 0^*1 + q_21$

($R = Q + RP$)

$q_2 = 0^*1 \cdot 1^*$ —⑤

$q_1 + q_2 = 0^* + 0^*1 \cdot 1^*$

$= 0^*(\varepsilon + 11^*)$

$= 0^* \cdot 1^*$ [$\because \varepsilon + RR^* = R^*$]

25/1/23

Regular Grammar:

Naam Chomsky is the founder of this Grammar.

Type 0 : UN-Restricted Grammar — Recursive (or) Enumerable Languages → TM

Type 1 : → CSG → CSL → Linear Bounded Automata

Type 2 : → CFG → CFL → PA CFG: Context Free Grammar.

Type 3 : → RG → RL → FSA

TM : Turing Machine

CSG : Context Sensitive Grammar

PA : Push down Automata.

Regular Grammer Contains 4-tuples.

$G = \{V, T, P, S\}$

V : Variables (or) non-terminals → Upper case letters.

T : Terminals → Lower case letters.

P : Production Rules.

S : Start Symbol.

Derivations from a grammer:

$G1 = (\{S, A\}, \{a, b\}, S)$ S : starting Symbol.

Production rules: $\{S \to aAb,\ aA \to aaAb,\ A \to \epsilon\}$

$S \to aAb$ (by Pr-1)

$S \to ab$ (by Pr-3)

$S \to aAb$ (by Pr-1)

$S \to aaAbb$ (by Pr-2)

$S \to aabb$ (by Pr-3)

Pr : production Rule.

$G2 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB,\ A \to a,\ B \to b\})$

$S \to AB$ (by pr-1)	$S \to AB$ (by pr-1)
$S \to aB$ (by pr-2)	$S \to Ab$ (by pr-3)
$S \to ab$ (by pr-3)	$S \to ab$ (by pr-2)

$G3 = (\{S, A, B\}, \{a, b\}, \{S \to AB,\ A \to aA/a,\ B \to bB/b\})$

Note:
$A \to aA/a$
$\Rightarrow$
$A \to aA$ (or) $A \to a$

$S \to AB$ (by pr-1)	$S \to AB$ (by pr-1)
$S \to aB$ (by pr-2)	$S \to aAB$ (by pr-2)
$S \to ab$ (by pr-3)	$S \to aaB$ (by pr-2)
	$S \to aab$ (by pr-3)
$S \to AB$ (by pr-1)	$S \to AB$ (by pr-1)
$S \to aB$ (by pr-2)	$S \to aAB$ (by pr-2)
$S \to abB$ (by pr-3)	$S \to aAbB$ (by pr-3)
$S \to abb$ (by pr-3)	$S \to aabB$ (by pr-2)
	$S \to aabb$ (by pr-3)

Pumping Lemma for Regular Languages

→ Assume that the given language is regular.

→ It has to have a pumping length; say 'p'.

→ All strings longer than p.

→ Now, find a string 's' in given language such that length of $s \geq p$.

→ Now, divide 's' into 3 parts, : x, y, z.

→ Show that xy^iz is not belongs to 'a'.

Pb: Using pumping lemma prove that the given language $L = \{a^n b^n / n \geq 0\}$ is not a regular language.

Sol:

Assume A is regular.

pumping length is p (say). => $s = a^p b^p$

Let p = 7. => $L = \{a^7 b^7\}$

s = aaaaaaabbbbbbb

Case-1: Assume that y part contains only 'a' part:

s = aa aaaa abbbbbbb
(x) (y) (z)

xy^2z : aa aaaaaaaa abbbbbbb

Case-2: Assume that y part contains only 'b' part:

s = aaaaaaabb bbbb b
(x) (y) (z)

Case-3: Assume 'y' part contain both 'a' & 'b'.

s = aaaaaa aabb bbbbb
(x) (y) (z)

xy^2z : aaaaaa aabbaabb bbbbb

Case-1 ✓

Case-2 ✗

Case-3 ✗

CO-2

Context Free Grammer (CFG):

RG	CFG
$A \to a$	$A \to \phi$
$(V \cup T)$	$(V \cup T)^*$

Ex: For generating a language that generates equal no. of a's and equal no. of b's in the form $a^n b^n$, the context free grammer will be defined like this:

Sol:

$G = \{(S, A), (a, b), S \to aAb, A \to aAb / \epsilon\}$

$S \to aAb$ (by pr-1) | $S \to aAb$ (by pr-1)

$S \to ab$ (by pr-2) | $S \to aaAbb$ (by pr-2)

$S \to aaaAbbb$ (by pr-2)

$S \to aaabbb$ (by pr-2)

Derivation Tree's / parse Tree's:

(G)

Variables
Terminals
Production rules
Start

Root vertex
Variables
Terminals

Ex: $G = \{V, T, P, S\}$ (0 : zero)

where $S \to 0B$

$A \to 1AA / \epsilon$

$B \to 0AA$

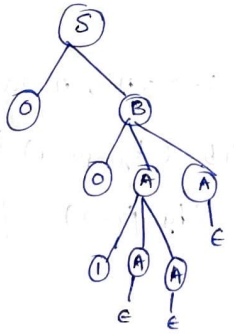
Sol:

Derivation Tree:

$V = \{S, A, B\}$

$T = \{0, 1\}$

$S \Rightarrow$



Derivation Tree's are classified into two types:

① Left Most Derivation Tree.

② Right Most Derivation Tree.

Ex: aabaa → required language.

From the given context free grammar:

$S \to aAs / ass / \epsilon$

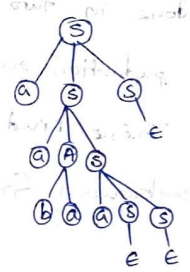
$A \to sbA / ba$

Sol:

→ Identify the no. of variables:

$V = \{S, A\}$

→ Terminals $T = \{a, b\}$



H/W

Ambiguious Grammar

→ $G = (\{S\}, \{a+b\}, +, *\}, S, P)$

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

Required: $a+a*b$

$S \to S+S$ (from pr-1)

$S \to a+S$ (from pr-3)

$S \to a+S*S$ (from pr-2)

=> S ⟶ a+a*S (from pr-3)

=> S ⟶ a+a*b (from pr-4)

S ⟶ S*S (from pr-2)

S ⟶ S*b (from pr-4)

S ⟶ S+S*b (from pr-1)

S ⟶ a+S*b (from pr-3)

S ⟶ a+a*b (from pr-3)

Here the same problem is done in two ways by changing the order of production rules but the final answer is same. These kind of grammar is known as Ambiguous Grammar.

Simplification of CFG

↳ 3 ways

1) direct method (or) Reduction CFG (or) Removing useless symbols.
2) Removable of NULL productions.
3) Removable of unit productions.

Direct Method:

→ Find a reduced grammar equivalent to one grammar having some production rules like:

P: S ⟶ AC/B
A ⟶ a
C ⟶ c/BC
E ⟶ aA/e

Sir

V = {S, A, B, C, E}

T = {a, c, e}

After deleting E, we have only 3 production rules.

S ⟶ AC/B
A ⟶ a
C ⟶ c/BC

Here we deleted 'E' because it is not available at the RHS.

S ⟶ AC | S ⟶ B
S ⟶ aC
S ⟶ ac

Here we can't generate any language further with 'B'. Hence remove 'B' wherever it was available.

→ S ⟶ AC
A ⟶ a
C ⟶ c

Reduction CFG

Pb → G P: S ⟶ AC/B
A ⟶ a
C ⟶ c/BC
E ⟶ aA/e

phase 1

→ Firstly find all the terminals.

T = {a, c, e}

W_1 = {A, C, E}

→ In w_1 we didn't include 'S' because it is not generating any terminals.

$$w_2 = \{A, C, E, S\}$$

Before writing w_2 we need to add all the elements present in w_1. S is added to w_2 because A is occured with the help of S, E. Since E is already available, S is additionally added.

$$w_3 = \{A, C, E, S\}$$

Now: $w_2 = w_3$

$$G' = \{(A, C, E, S), (a, c, e), S, P;$$

$$(S \to AC, A \to a, C \to c, E \to aA \mid e)\}$$

phase - 2:

$$y_1 = \{S\}$$

$$y_2 = \{S, A, C\}$$

Here A, C are added because $S \to AC$, hence we need to include A & C.

$$y_3 = \{S, A, C, a, c\}$$

$$y_4 = \{S, A, C, a, c\}$$

$$y_3 = y_4$$

Now:

$$G'' = \{(S, A, C), (a, c), S;$$

$$(S \to AC, A \to a, C \to c)\}$$

Removal of Unit productions:

Pb: Remove unit productions from the grammar whose production rules are given by:

P:
$S \to XY$
$X \to a$
$Y \to Z \mid b$
$Z \to M$
$M \to N$
$N \to a$

Sol: From the given grammar

$$V = \{S, X, Y, Z, M, N\}$$

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

All the unit productions in the given grammar: (one non-terminal goes to another non-terminal is called as unit production),

$Y \to Z$
$Z \to M$
$M \to N$

P: $S \to XY$, $X \to a$, $Y \to Z \mid b$, $Z \to M$, $M \to N$, $N \to a$

P: $S \to XY$, $X \to a$, $Y \to Z \mid b$, $Z \to M$, $M \to a$, $N \to a$

P: $S \to XY$, $X \to a$, $Y \to Z \mid b$, $Z \to a$, $M \to a$, $N \to a$

P: $S \to XY$, $X \to a$, $Y \to a \mid b$, $Z \to a$, $M \to a$, $N \to a$

Now:

$S \to XY$	$S \to XY$
$S \to aY$	$S \to aY$
$S \to aa$	$S \to ab$

∴ M, N, Z are not used in S, we call them unreachable states & remove them.

=> P ⟶ S → XY, X → a, Y → a/b.

Removal of NULL productions:

Pb → Remove all the NULL productions from the grammar

$S \longrightarrow ABAC$

$A \longrightarrow aA \mid \epsilon$

$B \longrightarrow bB \mid \epsilon$

$C \longrightarrow c$

Sol:

$V = \{S, A, B, C\}$

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

$\begin{cases} A \longrightarrow aA \mid \epsilon \\ B \longrightarrow bB \mid \epsilon \end{cases}$

$A \longrightarrow \epsilon$

$B \longrightarrow \epsilon$

eliminate $A \longrightarrow \epsilon$

$S \longrightarrow ABAC$ $\qquad$ $A \longrightarrow aA \mid \epsilon$

$A \longrightarrow aA$

$A \longrightarrow a$

=> $S \longrightarrow BAC \mid ABC \mid BC$ (occurred by subs ϵ on A)

=> $S \longrightarrow ABAC \mid BAC \mid ABC \mid BC$ $\qquad$ => $A \longrightarrow aA \mid a$

Now to eliminate $B \longrightarrow \epsilon$

$S \longrightarrow ABAC$ ∴

$S \longrightarrow AAC$

=> $S \longrightarrow ABAC \mid AAC$

$B \longrightarrow bB \mid \epsilon$

$B \longrightarrow bB$

$B \longrightarrow b$

∴ $B \longrightarrow bB \mid b$

Chomsky Normal Form:

Conversion of CFG to CNF:

Pb → The given grammar is:

$S \longrightarrow ASA \mid aB$

$A \longrightarrow B \mid s$

$B \longrightarrow b \mid \epsilon$

Step-1: Find the starting symbol.

$S' \longrightarrow S$ (∵ S is occurring 2 times in RHS)

Step-2: Remove all NULL productions:

$B \longrightarrow b \mid \epsilon$

=> $\boxed{B \longrightarrow \epsilon}$

To eliminate $B \longrightarrow \epsilon$

$S \longrightarrow ASA \mid aB$

=> $S \longrightarrow ASA \mid a$

∴ $\boxed{S \longrightarrow ASA \mid aB \mid a}$

$A \longrightarrow B \mid s$

$A \longrightarrow \epsilon \mid s$

=> $\boxed{A \longrightarrow B \mid s \mid \epsilon}$

$B \longrightarrow b$

$S' \longrightarrow S$

To eliminate $A \longrightarrow \epsilon$

=> $S \longrightarrow ASA \mid aB$

$S \longrightarrow SA \mid AS \mid s \mid aB$

=> $\boxed{S \longrightarrow ASA \mid aB \mid SA \mid AS \mid s}$

$A \longrightarrow B \mid s$

$B \longrightarrow b$

$S' \longrightarrow S$

Step-3: Remove all unit productions:

$S \longrightarrow S$

$A \longrightarrow B$

$A \longrightarrow S$

$S' \longrightarrow S$

first remove $S \longrightarrow S$

$\therefore S \longrightarrow ASA \mid aB \mid SA \mid AS$ (Here $S \to S$ should not be included because we should remove it).

$A \longrightarrow B \mid s$

$B \longrightarrow b$

$S' \longrightarrow S$

Second remove $S' \longrightarrow S$

$\Rightarrow S \longrightarrow ASA \mid aB \mid SA \mid AS$

$A \longrightarrow B \mid S$

$B \longrightarrow b$

$S' \longrightarrow ASA \mid aB \mid SA \mid AS$

Third remove $A \longrightarrow B$

$\Rightarrow S \longrightarrow ASA \mid aB \mid SA \mid AS$

$S' \longrightarrow ASA \mid aB \mid SA \mid AS$

$B \longrightarrow b$

$A \longrightarrow b \mid S$

Finally remove $A \longrightarrow S$

$\Rightarrow S \longrightarrow ASA \mid aB \mid SA \mid AS$

$S' \longrightarrow ASA \mid aB \mid SA \mid AS$

$B \longrightarrow b$

$A \longrightarrow b \mid ASA \mid aB \mid SA \mid AS.$

Step-4

Select the variables where 3 non-terminals exist and reduce them.

$\Rightarrow X = SA$

$S \longrightarrow Ax \mid aB \mid SA \mid AS$

$A \longrightarrow b \mid Ax \mid aB \mid SA \mid AS$

$B \longrightarrow b$

$S' \longrightarrow Ax \mid aB \mid SA \mid AS$

Step-5

Since there is a condition where one non-terminal and one terminal exist together change them by:

$a = Y$

$S \longrightarrow Ax \mid YB \mid SA \mid AS$

$A \longrightarrow b \mid Ax \mid YB \mid SA \mid AS$

$B \longrightarrow b$

$S' \longrightarrow Ax \mid YB \mid SA \mid AS.$

Greebach Normal Forms (GNF):

Ex: the given grammar is:

$S \longrightarrow CA \mid BB$

$B \longrightarrow b \mid SB$

$C \longrightarrow b$

$A \longrightarrow a$

Sol: Name all the non-terminals:

$S \longrightarrow A_1$

$C \longrightarrow A_2$

$A \longrightarrow A_3$

$B \longrightarrow A_4$

=> $A_1 \longrightarrow A_2A_3 \mid A_4A_4$

$BA_4 \longrightarrow b \mid A_1A_4$

R

$A_2 \longrightarrow b$

Pb

$A_3 \longrightarrow a$

$A_1 \longrightarrow A_2A_3$ (i, j)

$i < j$

$1 < 2$ true

$A_1 \longrightarrow A_4A_4$ (i, j)

$i < j$

true

$A_4 \longrightarrow b$ | $A_4 \longrightarrow A_1A_4$ (i, j)

$i < j$ (false)

Now,

$A_4 \longrightarrow b$ | $A_4 \longrightarrow A_2A_3A_4$ (i, j) $i < j$ false | $A_4 \longrightarrow bA_3A_4$ | $A_4 \longrightarrow A_4A_4A_4$ (i, j) $4 < 4$ left recursion ($\because 4 \leq 4$)

=> $A_1 \longrightarrow A_2A_3 \mid A_4A_4$

$A_4 \longrightarrow b \mid bA_3A_4 \mid A_4A_4A_4$

$A_2 \longrightarrow b$

$A_3 \longrightarrow a$

Let $Z \longrightarrow A_4A_4 \mid A_4A_4Z$

=> $A_4 \longrightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

Now, check whether the grammar follows GNF (or) Not.

$\therefore$ $A_1 \longrightarrow bA_3 \mid bA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_3A_4ZA_4$

$A_2 \longrightarrow b$

$A_3 \longrightarrow a$

$A_4 \longrightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

$Z \longrightarrow bA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_3A_4ZA_4 \mid bA_4Z \mid bA_3A_4A_4Z \mid bZA_4Z \mid bA_3A_4ZA_4Z$.