flat

Basic Concepts (or) mathematical notation in Automata theory

- Alphabet: finite set of symbol

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

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

  $\Sigma = \{A, B, \ldots Z\}$

- String: Finite collection of symbol over alphabet Σ.

  String is denoted by w (or) known as word

1. Length of a String: $|w|$

   ex:- $w = abc$, then $|w| = 3$

   (3 char is the length of the string)

2. Empty String: (string with '0' symbol) is denoted by Epsilon (ϵ)

   ↳ length is zero. $|w| = 0$

3. Prefix:- Any no of leading symbol (↳ means starting)

   ex:- $w = abc$

   Prefix are ϵ, a, ab, abc

4. Proper Prefix:- we can't take Epsilon (ϵ) and the string

   ex:- $w = abc$

   a, ab

5. Suffix of a String:- trailor symbols

   ex:- $w = abc$ (←)

   ϵ, c, bc, abc

6. Proper Suffix: expet epsilon & the string are proper suffix

   ex:-

   $w = abc$

   c, bc

7. Substring:- It is a contiguous sequence of charater within a string

   ex:- $w = abc$

   $\epsilon, a, ab, abc, c, bc, b$

- If the length of string is n then the no. of prefix/suffix are $n+1$

LANGUAGE

- It is a collection of strings generated from Σ and denoted by 'L'

  A language over the alphabet Σ is a subset of Σ^* i.e $L \subseteq \Sigma^*$

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

  $L = \{0, 1, 00, 01, 10, 11, 000, 111, 001, 100, \ldots\}$

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

L = Any no. of 0's $L = \{\epsilon, 0, 00, 000, \ldots\}$

L = Any no. of 0's & 1's then $L = \{\epsilon, 0, 1, 01, 00, 10, 11, \ldots\}$

Operations of Language

1. Kleene closure (or) star closure :- It is denoted by Σ^* (↳ contains string with any length)

   Ex:- $\Sigma = \{0\}$ $\Sigma^* = \{\epsilon, 0, 00, 000, \ldots\}$

Σ = {ε, 0, 00, 000 ....} with any length

2. Positive closure :- denoted by Σ^+ it doesn't contain ε (epsilon)

Ex:- $\Sigma = \{0\}$ $\Sigma^+ = \{0, 00, 000, 0000, \ldots\}$

Strings

- Concatenation :- If w_1 & w_2 are two strings then concatenation of w_1 & w_2 is represented by $w_1 \cdot w_2$.

ex: let $w_1 = abc$, $w_2 = def$ then $w_1 \cdot w_2 = abc.def$ so, $|w_1 \cdot w_2| = |w_1| + |w_2|$

- Power of Alphabet :- If Σ is an Alphabet, power of alphabet Σ is denoted by Σ^k which contains set of strings of length k ($k \geq 1$)

ex:- let $\Sigma = \{a, b\}$, $\Sigma^1 = \{a, b\}$ so $|\Sigma^1| = 2^1 = 2$, $|\Sigma^2| = \{aa, ab, ba, bb\}$ (contains strings with length 2) so $|\Sigma^2| = 4$

$\boxed{|\Sigma^k| = n^k}$ where n is no. of symbol of alphabet

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots$ All strings including empty string

↓ Kleene closure (or) Star closure

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$ All strings excluding empty string

↓ positive closure

- Transpose (Reverse) :- Let $w = abc$ then $w^T = cba$
- Palindrome :- Reverse of a string is equal to given string ex:- madam

Operations on Language :

1. Union of 2 languages :- It is denoted by $L_1 \cup L_2$ or $L_1 + L_2$ we get the strings from both languages.

ex: L_1 = {good, boy} L_2 = {good, girl} then $L_1 \cup L_2$ = {good, boy, good girl}

2. Intersection of 2 Languages :- It is denoted by $L_1 \cap L_2$ we get common strings from both languages $L_1 \cap L_2$ = {good}

3. Concatenation :- It is denoted by $L_1 \cdot L_2$ we combine the strings from both languages. ex:- $L_1 \cdot L_2$ = {goodgood, goodgirl, boygood, boygril}

4. Difference :- $L_1 - L_2$ contains the string which are in L_1 but not in L_2

Ex:- $L_1 - L_2 = \{boy\}$ or $\{girl\}$

5. Kleene closure (L^*) :- contains the strings of any length including null string

$L^* = L^0 \cup L^1 \cup L^2 \ldots$

$L^* = \in \cup L^1 \cup L^2 \ldots$

Ex:- $\Sigma = \{a, b\}$

$L^* = \{\in, a, b, aa, ab, \ldots\}$

6. Positive closure (or) Kleene plus :- Denoted by L^+ ; contains the strings of any length excluding null string.

$L^+ = L^* - \in$.

$L^+ = L^1 \cup L^2 \cup L^3 \ldots$

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

$L^+ = \{a, b, ab, aa \ldots\}$

Sets

- Set is a collection of elements

Ex:- Set of 10 integers

- finite Set :- Countable no. of elements

Ex:- $O = \{1, 3, 5, 7, 9\}$

- Infinite Set :- Uncountable

$O = \{1, 3, 5, \ldots\}$

Member or Element : $\in$ Symbol is used to denote wheather the element belong to the set or not

$S = \{1, 2, 3, 4\}$

$1 \in S$ (1 belongs to set 'S'), $2 \in S$, $3 \in S$, $4 \in S$

$7 \notin S$ (7 doesnot belong to set 'S')

- Singleton Set :- contains only one element

ex:- $S = \{1\}$

Set Representation (or) Set Notations

1. Roster Notation : the elements must be in curly braces and separate by ,

ex:- $S = \{1, 2, 3, 4 \ldots\}$

2. Set builder Notation

↓

all the elements must have same property.

Ex:- Set of odd number . $S = \{1, 3, 5, 7, \ldots\}$ then $S = \{n \mid P(n)\}$

$S = \{n \mid n$ is a odd no$\}$

Set Therminology :

Cardinality : it specifies the no. of elements present in the set

ex:- $S = \{1, 2, 3, 4\}$

$|S| = 4$

$S = \{1, \{2, 3\}, \{3, 4, 5\}\}$

$|S| = 3$

Subset or Set Inclusion : $A \subseteq B$ all the elements present in A must be present in B then A is subset B

Eu: $A = \{1, 2, 3\}$ $B = \{1, 2, 3, 4, 5\}$

Equality of sets : $A \subseteq B$ and $B \subseteq A$

Eu: $A = \{1, 2, 3\}$ $B = \{1, 2, 3\}$ then $A = B$

proper subset :- It is denoted by '$\subset$'. $A \subset B$ if $A \subseteq B$ & $A \neq B$

these are the condition for proper subset

Eu: $A = \{1, 2, 3\}$ $B = \{1, 2, 3, 4, 5\}$

Universal set : U or E

Eu :- $A = \{1, 2\}$ $B = \{3, 4\}$, $C = \{a, b\}$

$U = \{1, 2, 3, 4, a, b\}$

Empty set or Null set : ϕ

$\phi = \{\ \}$

Power set : set of all subsets → Eu :- $S = \{a, b\}$ $P(S) = \{\phi, \{a\}, \{b\}, \{a, b\}\}$

$P(A)$ $n \rightarrow 2^n$ → no. of elements in powerset

no. of elements in set

power set

SET OPERATIONS

1. Intersection :- denoted by $\cap$, $\wedge$. common elements of both the sets.

$A \cap B = \{x \mid (x \in A) \wedge (x \in B)\}$ Eu: $A = \{1, 2, 3, 4, 5\}$ $B = \{3, 5, 6, 7, \}$

$A \cap B = \{3, 5\}$

2. Disjoint :- Two sets are said to be disjoint if $A \cap B = \phi$

Eu: $A = \{1, 2\}$ $B = \{3, 4\}$ then $A \cap B = \phi$

3. Union :- Denoted by $\cup$ or $\vee$. $A \cup B = \{x \mid (x \in A) \vee (x \in B)\}$

Eu: $A = \{1, 2, 3, 4\}$ $B = \{3, 4, 5, 6\}$ $A \cup B = \{1, 2, 3, 4, 5, 6\}$

4. Difference :- $A - B = \{x \mid (x \in A) \wedge (x \notin B)\}$

Eu :- $A = \{1, 2, 3, 4, 5\}$ $B = \{3, 5, 6, 7\}$ $A - B = \{1, 2, 4\}$; $B - A = \{6, 7\}$

5. Complement :- denoted by A^C or $\sim A$ or $\overline{A}$. U and A

$A^C = U - A = \{x \mid (x \in U) \wedge (x \notin A)\}$ Eu: $U = \{1, \overline{2}, 3, 4, \overline{5}\}$ $A = \{2, 5\}$

6- Symmetric difference :— denoted by Δ

$A^c = \{1, 3, 4\}$

$A \Delta B = (A - B) \cup (B - A)$

ex:- $A = \{1,2,3,4,5\}$ $B = \{3,4,5\}$ so $A \Delta B = \{1,2\}$

$A - B = \{1,2\}$ $B - A = \emptyset$

7. Cartesian Product :— collection of ordered pairs . denoted by '×'

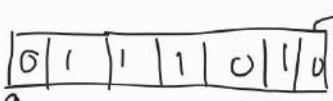
$A \times B = \{(x, y) \mid (x \in A) \cap (y \in B)\}$

ex:- $A = \{1,2\}$ $B = \{a, b, c\}$

$A \times B = \{(1,a), (1,b), (1,c), (2,a), (2,b), (2,c)\}$

$A \times B \neq B \times A$

finite Automata Model & formal Definition of F.A



→ these are known as cells which contain one symbol

Input tape

↑ Read Head moves from left to right

↑

finite control — all the states information will be maintained

'm' means mission $m = (Q, \Sigma, \delta, q_0, F)$

- Q : represents set of states
- Σ : input alphabet (contains (0,1) or ABC)
- δ : Transition function
- q_0 : initial state (only 1 initial state)
- F : set of final states (we can have multiple final states)

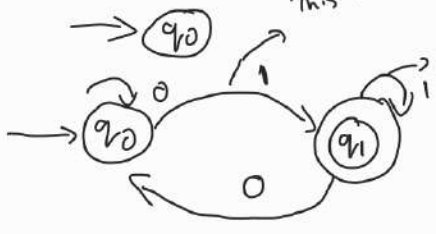
Representation of finite Automata

1. Transition Diagram :- Pictorial representation of finite automata.

L Initial state — L final state

no need of arrow symbol

ex:



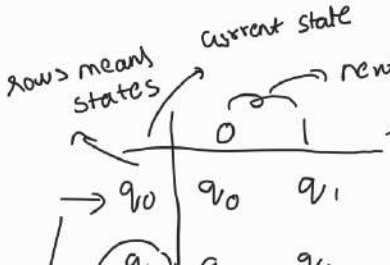
$m = (Q, \Sigma, \delta, q_0, F)$

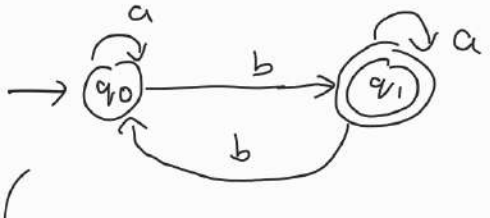
$m = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\})$

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

(current state → next state)

Transition Table





arrow represents initial state

→ final state is represent in a circle

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

DFA (Deterministic finite Automata)

'm' means mission $m = (Q, \Sigma, \delta, q_0, F)$

- Q → represents set of states
- Σ → input alphabet (contains (0,1) or ABC)
- δ → Transition function → $Q \times \Sigma = Q_1$ (another state)
- q_0 → initial state (only 1 initial state)
- F → set of final states (we can have multiple final states)

- In DFA if we apply an input symbol on a particular state there will be only one transition / one state.

Acceptability of a string by finite Automata

(or) Language Accepted by finite Automata

Let 'm' be the given Automata & 'w' be the given string. To check whether the input string is accepted by finite automata or not use the following

$\delta(q_0, w) \vdash^* q_F$ (zero or more transitions)

is $q_F \in F$, string is accepted

is $q_F \notin F$, string is rejected

Q] DFA for strings starts with "a"

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

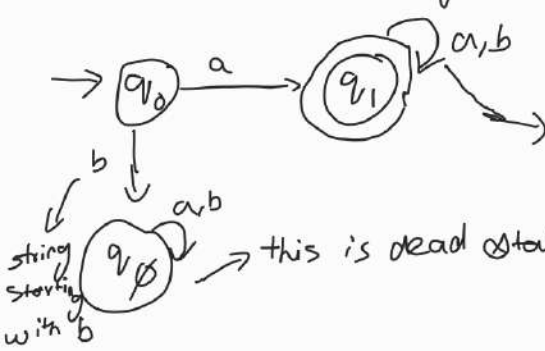
$L = \{a, aa, ab, aaabb, \ldots\}$ ← string (we can take any string from)

$\delta(q_0, \underline{a}abb) \vdash \delta(q_1, \underline{a}bb) \vdash \delta(q_1, \underline{b}b)$

↳ apply a on q_1

$\vdash (q_1, b) \vdash q_1 \in F$

↳ means input string is accepted



DFA Examples

1Q) Construct a DFA that accepts set of all strings over {a,b}∘ of string starting with 'a'

$\Sigma = \{a, b\}$; $L = \{a, aa, ab, aab, \ldots\}$ ($n = 1$)

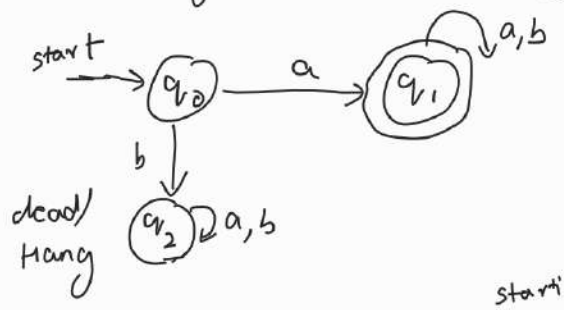
↳ $n+1 = 2$ states

2Q) starting with "ab"

$L = \{ab, aba, abb, abab \ldots\}$

↳ if min possible string length is 2 then n

$n+1 \Rightarrow$ states



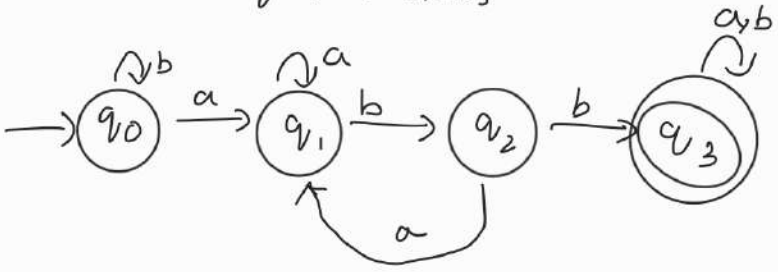
because the condition is that the string is starting with 'ab'

Q] Construct DFA that accepts set of all strings over {a, b} of strings containing "ab" as substring.

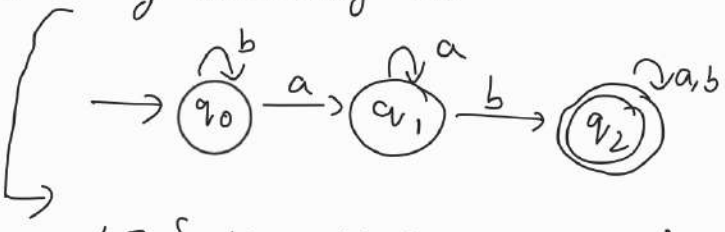
$\Sigma = \{a, b\}$

$L = \{ab, aab, aabb, bab, \ldots\}$

↓ we require 3 states



P] Having substring "abb"



$L = \{abb, aabb, babb, \ldots\}$

↓ 4 states

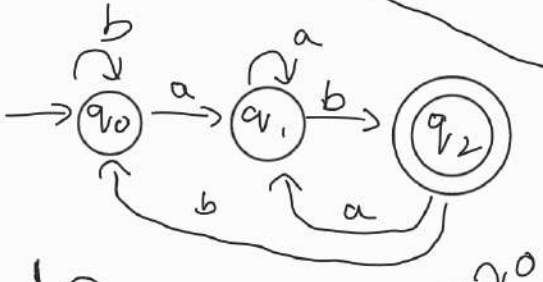
Q] Construct DFA that accepts set of all strings over {a, b} of strings ends with "ab".

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

$L = \{ab, aab, bab, aaab, baab \ldots\}$

↓ 3 states

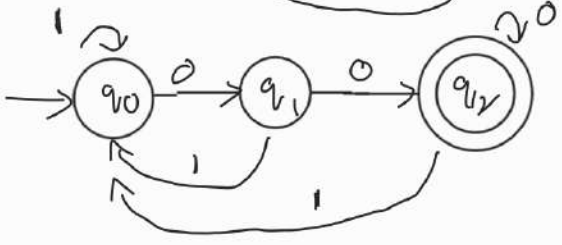
Q) ends with "00"



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

$L = \{00, 100, 000, 1000, 1100 \ldots\}$

↓ 3 states

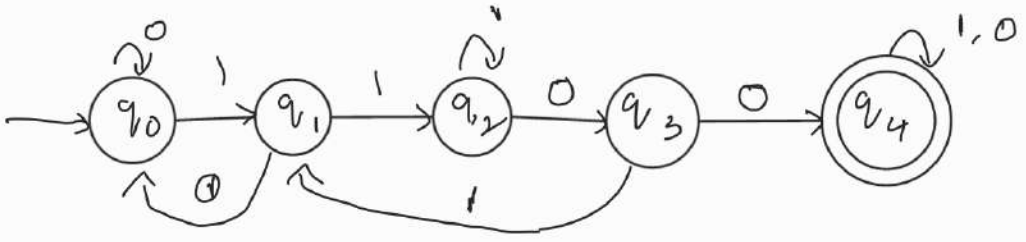


Q] Design DFA which Accepts set of all strings containing 1100 as substring.

$\Sigma = \{0, 1\}$ $L = \{1100, 01100, 011001, 00110011, \ldots\}$

↓ 5 states

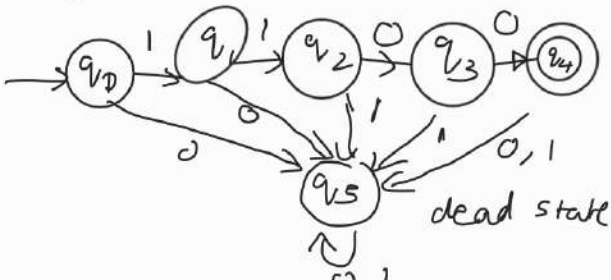
01101100



Q] Design DFA which Accepts the string 1100 only.

$\Sigma = \{0, 1\}$ $L = \{1100\}$

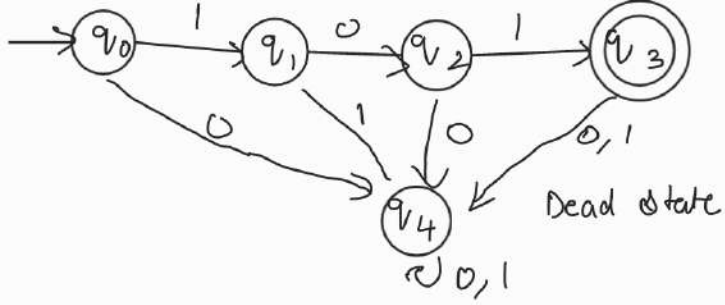
↓ 5 states



Q) for string 101 only

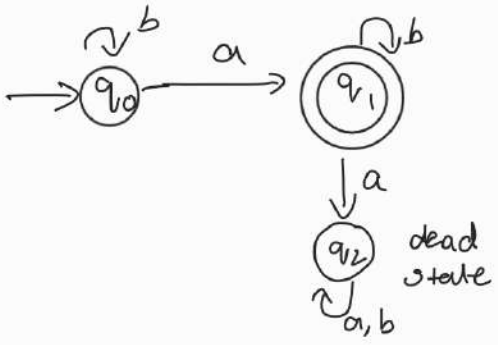
$\Sigma = \{0, 1\}$ $L = \{101\}$

↓ 4 states

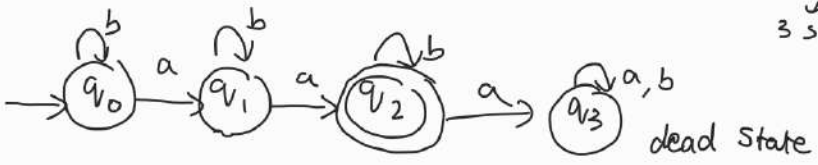


Q) Design DFA which Accepts set of all strings over $\Sigma = \{a, b\}$ with exactly one 'a'

$\Sigma = \{a, b\}$ $L = \{a, ab, abb, ba, bba, \ldots\}$ ↳ 2 states

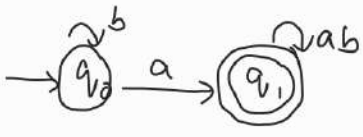


Q) Exactly two a's $\Sigma = \{a, b\}$, $L = \{aa, aab, baa \ldots\}$ ↳ 3 states

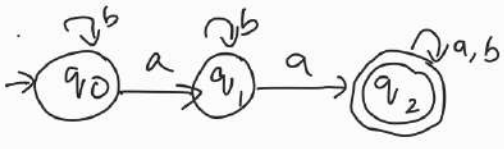


Q) Design DFA which Accepts set of all strings over $\Sigma = \{a, b\}$ with Atleast one 'a'

$\Sigma = \{a, b\}$ $L = \{a, ab, aa, baa, bbaa, \ldots\}$ ↳ 2 states



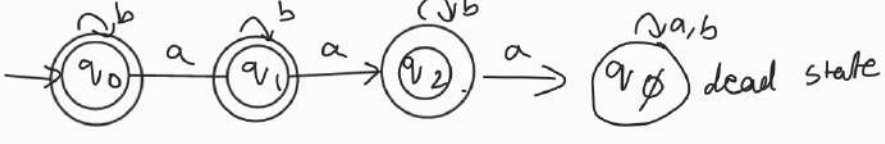
Q) Atleast two a's $\Sigma = \{a, b\}$ $L = \{aa, aba, baa, bbaa \ldots\}$ ↳ 3 states



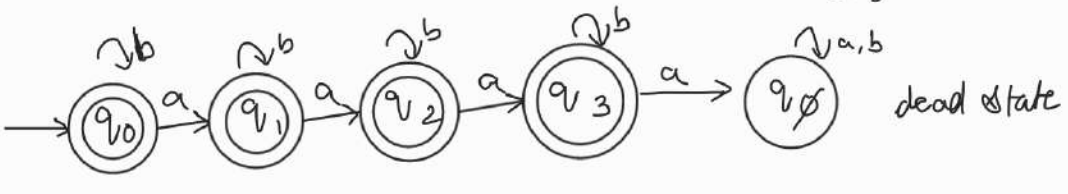
Q) DFA to accept set of string with ① Atmost 2 a's (or) not more than 2a's

② Atmost 3a's over alphabet $\{a, b\}$

$\Sigma = \{a, b\}$ $L = \{\epsilon, ab, aa, baa, baab, baba, bbaa \ldots\}$ ↳ 3 states

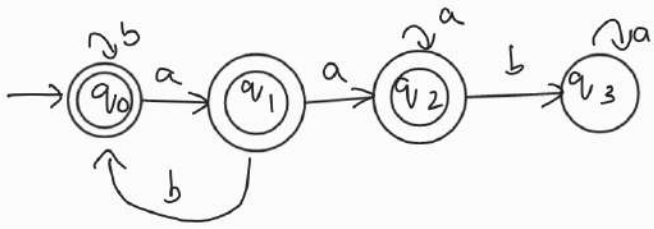


$\Sigma = \{a, b\}$ $L = \{\epsilon, a, aa, aaa, baa, bbaaa, baaabb \ldots\}$ ↳ 4 states



Q) Construct DFA which accepts set of strings over $\{a, b\}$ except be substring aab.

$\Sigma = \{a, b\}$ $L = \{a, b, ab, bab, bba, babb \ldots\}$



} because q_3 is not a final state even if the string aab will not be accepted.

Q] Construct DFA which accepts set of strings over {a, b} with

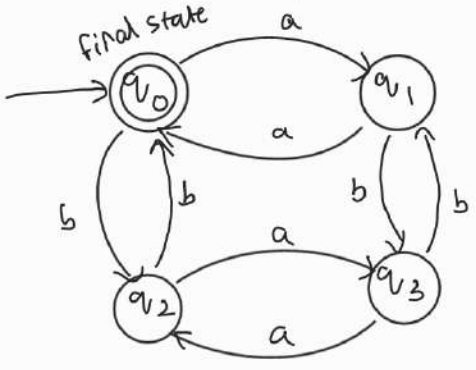
q_0 a) Even number of a's and even number of b's → L = { aabb, abab, aaaabb .... }

q_2 b) Even number of a's and odd number of b's

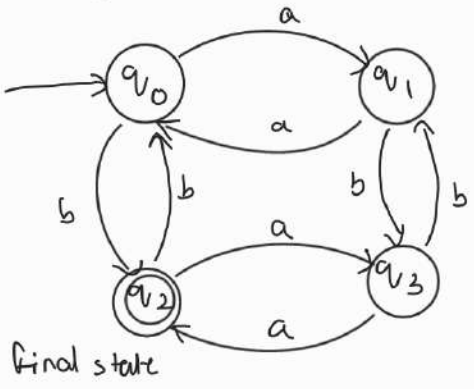
q_1 c) odd number of a's and even number of b's

q_3 d) odd number of a's and odd number of b's

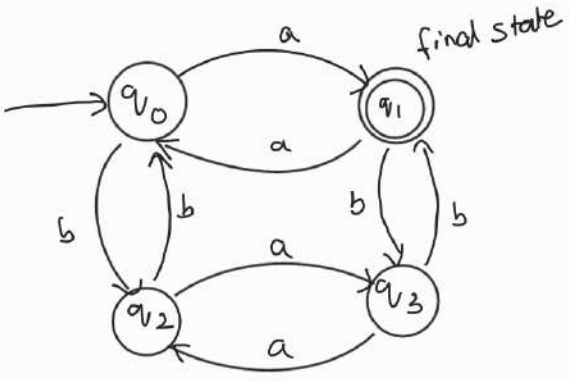
a)

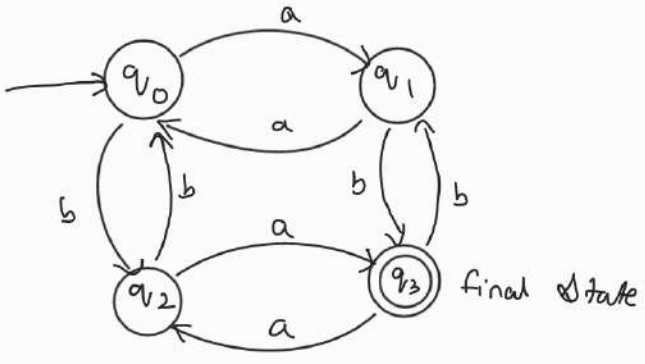


b)



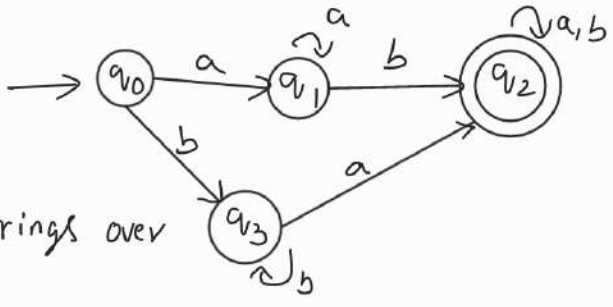
c)





Q] Construct DFA which accepts set of all strings over the Alphabet {a, b} in which L = { w | w contains the Substring ab or ba }

L = { ab, ba, aabb, bbaa, ... }
↳ 3 states

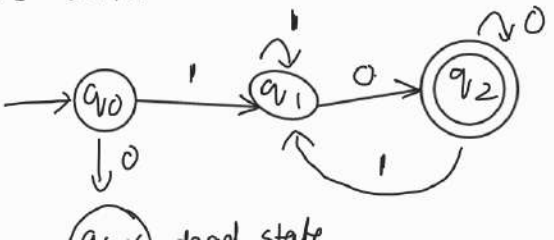


Q] Construct DFA which Accepts set of all strings over the Alphabet {0, 1} in which

a) Every string starts with 1 & Ends with 0 $\quad \Sigma = \{0, 1\}$

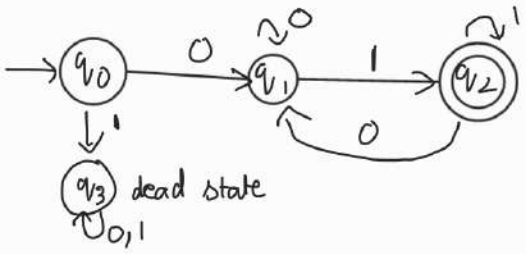
b) Every string starts with 0 & Ends with 1

L = { 10, 1010, 1000, ... }
↳ 3 states



L = { 01, 0011, 01001, ... }

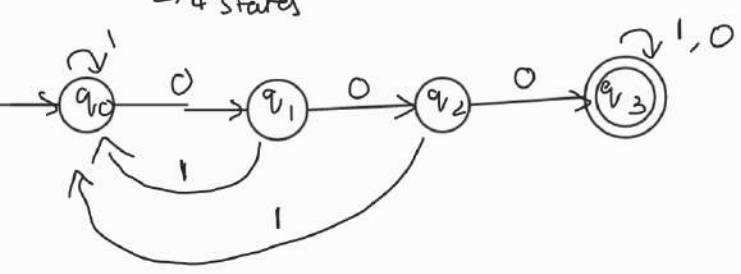
↳ 3 states

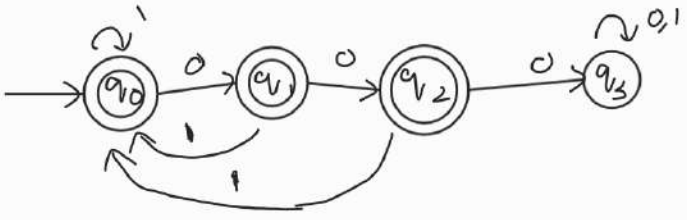


Q) DFA for strings with 3 consecutive 0's & without 3 consecutive 0's

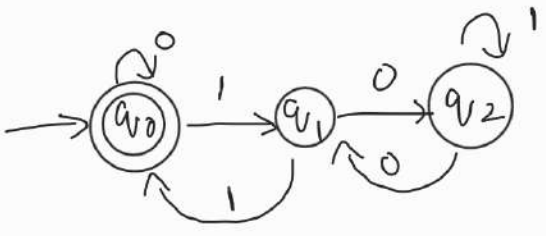
$L = \{000, 1000, 10001, 1000101 \ldots\}$

↳ 4 states





Q) Design DFA to accept all binary string which are divisible by 3

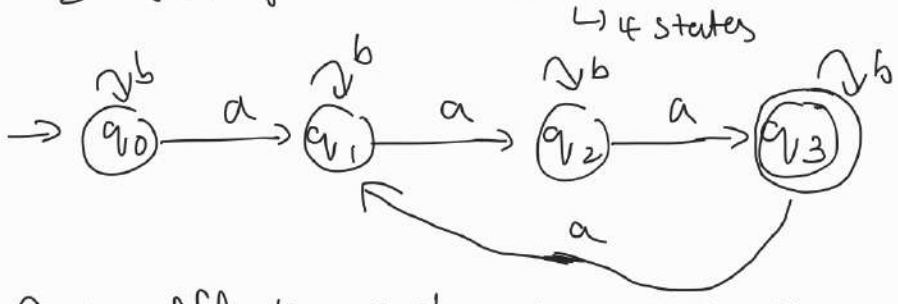


dn	b/n	rem
0	000	$0 - q_0$
1	001	$1 - q_1$
2	010	$2 - q_2$
3	011	$0 - q_0$
4	100	$1 - q_1$
5	101	$2 - q_2$
6	110	$2 - q_0$

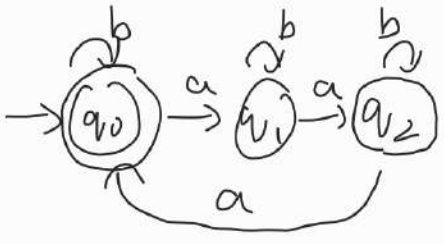
Q] Design DFA for number of a's are divisible by 3 over alphabet {a, b} (or) a's will always be tripled.

$\Sigma = \{a, b\}$ $L = \{aaa, baaba, abbababa, \ldots\}$

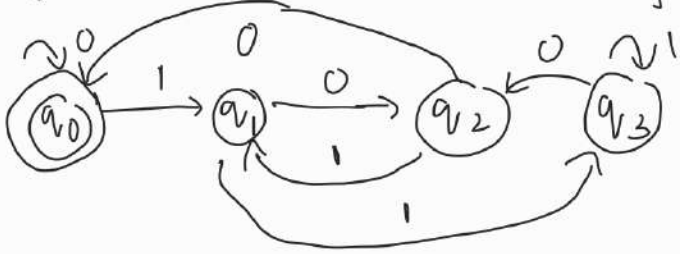
↳ 4 states



(or)



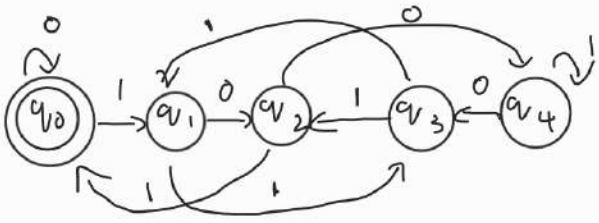
Q) Design DFA to accept set of all binary strings which are divisible by 4 $L = \{w | w \bmod 4 = 0\}$



dn	b/n	rem	dn	b/n	rem
0	000	0 q_0	5	101	1 q_1
1	001	1 q_1	6	110	2 q_2
2	010	2 q_2	7	111	3 q_3
3	011	3 q_3	8	1000	0 q_0
4	100	0 q_0			

Q) Design DFA to accept set of all binary strings divisible by 5

$L = \{w / w \bmod 5 = 0\}$



Transition Table

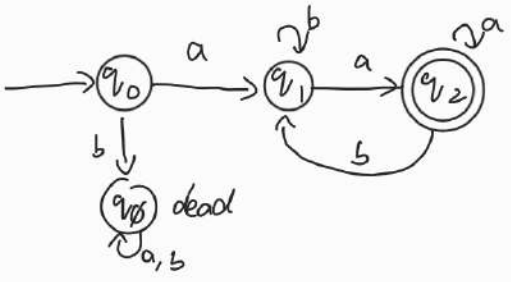
	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

dn	bn	Vem		5	101	0 q_0
0	000	0	q_0	6	110	1 q_1
1	001	1	q_1			
2	010	2	q_2	7	111	2 q_2
3	011	3	q_3	8	1000	3 q_3
4	100	4	q_4	9	1001	4 q_4
				10	1010	0 q_0

Q) Design DFA to accept the language $L = \{ \underset{\text{starts}}{a} w \underset{\text{ends}}{a} / \text{where } w \in (a+b)^* \}$

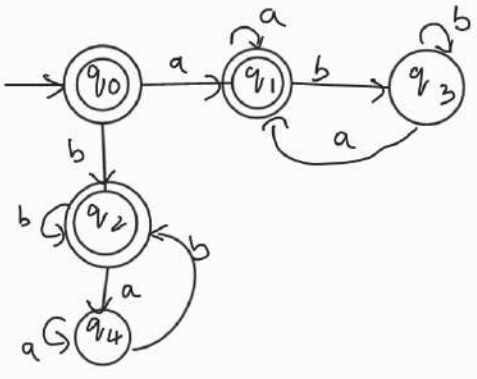
$(a+b)^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aba, \ldots\}$ $\quad \Sigma = \{a, b\}$

$L = \{aa, aaa, aba, ababa, \ldots\}$



Q] DFA which accepts strings starting & ending with same symbol over $\{a, b\}$

$\Sigma = \{a, b\}$ $\quad L = \{\epsilon, a, b, aa, bb, abaa, babbab, \ldots\}$



NFA – Non Deterministic Finite Automata

5 tuples – $m = (Q, \Sigma, \delta, q_0, f)$

- Q: finite states
- Σ: alphabets
- q_0: initial state
- f: final state

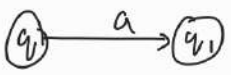
$\delta = Q \times \Sigma \to 2^Q$

transition function maps

Differences Between DFA and NFA

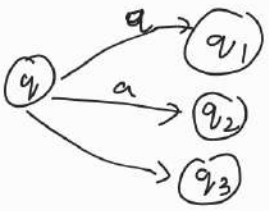
DFA	NFA
• It is defined using 5 Tuples $m = (Q, \Sigma, \delta, q_0 f)$	• It is defined using 5 Tuples $m = (Q, \Sigma, \delta, q_0 f)$
$\delta : Q \times \Sigma \to Q$	$\delta : Q \times \Sigma \to 2^Q$

- In case of DFA, every transition generates single state with a particular I/p symbol



- Every state must contain one transition with every I/p symbol in given alphabet
- It is difficult to construct.
- Epsilon (ϵ) transitions are not allowed

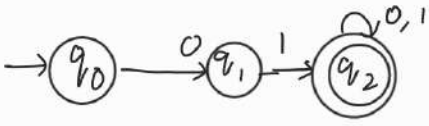
- In case of NFA, transition generates more than one outcome with a particular I/p symbol.



- No need to maintain transition with every I/p symbol
- It is easy to construct
- ϵ -transitions are allowed

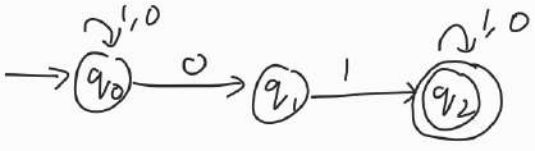
NFA Examples

Q] Start with '01' $\Sigma = \{0,1\}$; $L = \{01, 010, 011, 0101, 0110, \ldots\}$ — 3 states

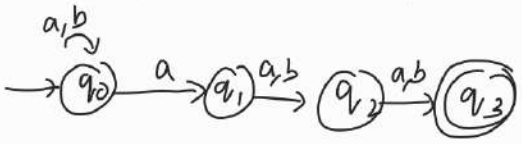


Q) Ending with '01' $\Sigma = \{0,1\}$; $L = \{01, 101, 001, 1001, 1101, \ldots\}$ — 3 states

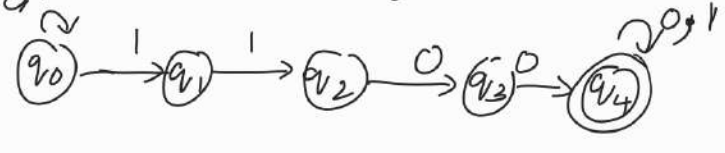
Q) Substring '01' $\Sigma = \{0,1\}$, $L = \{01, 101, 001, 0010, 1011, \ldots\}$ — 3 states



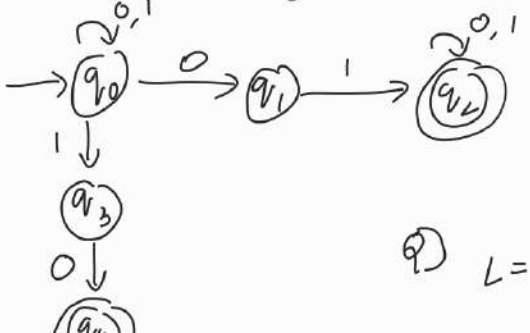
Q) 3rd symbol from right end is 'a'



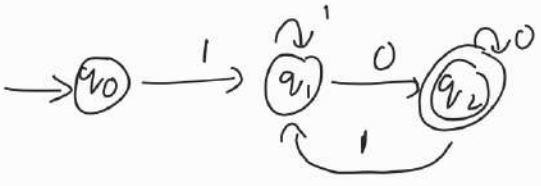
Q] '1100' min string



Q) String containing either '01' or '10'



Q] Starts with 1 & ends with '0'

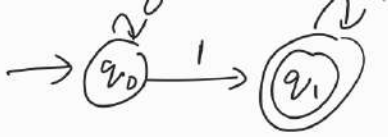


Q) $L = \{0^m 1^n \mid m \geq 0 \text{ and } n \geq 1\}$

$m=0, n=1 \Rightarrow 1$

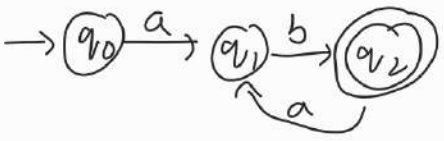
$m=1, n=2 \Rightarrow 011$

$m=2, n=3 \Rightarrow 00111$



Q) $(ab)^n$ with $n \geq 1$

$n=1 \Rightarrow ab$; $n=2 \Rightarrow abab$

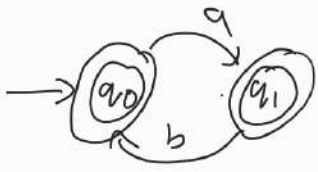


Q) $(ab)^n$ with $n \geq 0$

$n=0 \Rightarrow \phi$

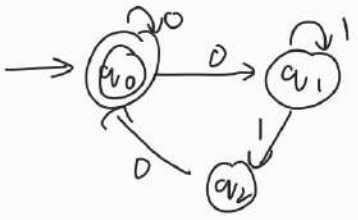
$n=1 \Rightarrow ab$

$n=2 \Rightarrow abab$



Converting NFA To DFA

Q) Design DFA from one given NFA



NFA

transition table (NFA)

	0	1
→ q_0	$\{q_0, q_1\}$	ϕ
q_1	ϕ	$\{q_1, q_2\}$
(q_2)	q_0	ϕ

DFA transition table

	0	1
→ q_0	$\{q_0, q_1\}$	ϕ (q_{dead})
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	q_0	$\{q_1, q_2\}$

$\delta(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0, q_1\} \cup \phi$

$\delta(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$

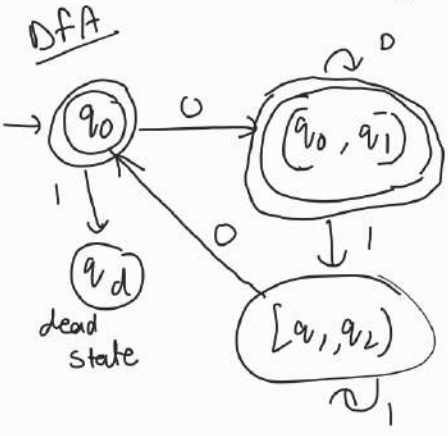
$= \phi \cup \{q_1, q_2\}$

$\delta(\{q_1, q_2\}, 0) = \delta(q_1, 0) \cup \delta(q_2, 0) = \phi \cup q_0$

$\delta(\{q_1, q_2\}, 1) = \delta(q_1, 1) \cup \delta(q_2, 1)$

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

DFA



Q) NFA to DFA

NFA Transition table

	a	b
→ q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
(q_2)	ϕ	ϕ

DFA Transition Table

	a	b
→ q_0	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	q_0

$\delta(\{q_0, q_1\}, a) = \delta(q_0, a) \cup \delta(q_1, a)$

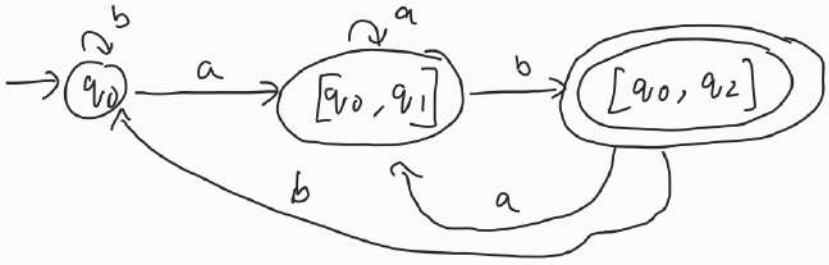
$= \{q_0, q_1\} \cup \phi$

$\delta(\{q_0, q_1\}, b) = \delta(q_0, b) \cup \delta(q_1, b)$

$= q_0 \cup q_2$

$\delta(\{q_0, q_2\}, a) = \delta(q_0, a) \cup \delta(q_2, a) = \{q_0, q_1\} \cup \phi$

$\delta(\{q_0, q_2\}, b) = \delta(q_0, b) \cup \delta(q_2, b) = q_0 \cup \phi$



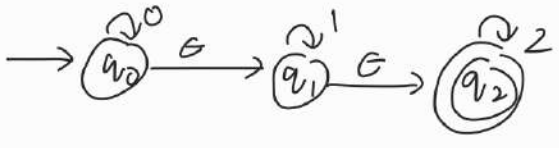
Converting NFA with ϵ-moves to NFA without ϵ-moves

- ϵ (Epsilon) moves from 1 state to another state without any I/P symbol.

Eg:

ϵ-closure(q) :- set of states that are reachable from states q with ϵ-transitions

1. Calculate ϵ-closure of all states
2. Calculate extended transition function $\delta'(q_0, 0)$ → delta dash



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

ϵ-closure(q_1) = $\{q_1, q_2\} = \epsilon(q_1)$

ϵ-closure(q_2) = $\{q_2\} = \epsilon(q_2)$

$\delta'(q_0, 0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), 0))$

$= \epsilon(\delta(\{q_0, q_1, q_2\}), 0))$

$= \epsilon(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$

$= \epsilon(q_0 \cup \phi \cup \phi)$

$= \epsilon(q_0)$

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

$\delta'(q_0, 0) = \{q_0, q_1, q_2\}$

$\delta'(q_0, 1) = \epsilon(\delta(\epsilon(q_0), 1))$

$= \epsilon[\delta(\{q_0, q_1, q_2\}), 1))$

$= \epsilon(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$

$= \epsilon(\phi \cup q_1 \cup \phi)$

$= \epsilon(q_1) = \{q_1, q_2\}$

$\delta'(q_0, 1) = \{q_1, q_2\}$

$\delta'(q_0, 2) = \epsilon(\delta(\epsilon(q_0), 2)) = \epsilon(\delta(\{q_0, q_1, q_2\}, 2)) = \epsilon(\delta(q_0, 2) \cup \delta(q_1, 2) \cup \delta(q_2, 2))$

$= \epsilon(\phi \cup \phi \cup q_2) = \epsilon(q_2) = \{q_2\}$

$\delta'(q_1, 0) = \epsilon(\delta(\epsilon(q_1), 0)) = \epsilon(\delta(\{q_1, q_2\}, 0)) = \epsilon(\delta(q_1, 0) \cup \delta(q_2, 0))$

$= \epsilon(\phi \cup \phi) = \phi$

$\delta'(q_1, 1) = \epsilon(\delta(\epsilon(q_1), 1)) = \epsilon(\delta(\{q_1, q_2\}, 1)) = \epsilon(\delta(q_1, 1) \cup \delta(q_2, 1))$

$= \epsilon(q_1 \cup \phi) = \epsilon(q_1) = \{q_1, q_2\}$

$\delta'(q_1, 2) = \epsilon(\delta(\epsilon(q_1), 2)) = \epsilon(\delta(\{q_1, q_2\}, 2)) = \epsilon(\delta(q_1, 2) \cup \delta(q_2, 2))$

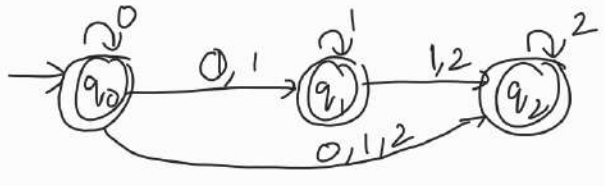
$= \epsilon(\phi \cup q_2) = \epsilon(q_2) = \{q_2\}$

$\delta'(q_2, 0) = \epsilon(\delta(\epsilon(q_2), 0)) = \epsilon(\delta(\{q_2\}, 0)) = \epsilon(\phi) = \phi$

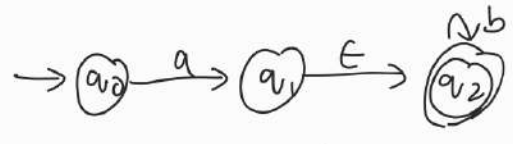
$\delta'(q_2, 1) = \epsilon(\delta(\epsilon(q_2), 1)) = \epsilon(\delta(q_2, 1)) = \epsilon(\phi) = \phi$

$\delta'(q_2, 2) = \epsilon(\delta(\epsilon(q_2), 2)) = \epsilon(\delta(q_2, 2)) = \epsilon(q_2) = \{q_2\}$

NFA



Q)



step 1:- find ϵ-closures

step 2:- find extended transition function.

$\delta'(q_0, a) = \{q_1, q_2\}$

ϵ-closure $(q_0) = \{q_0\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

$\delta'(q_0, a) = \epsilon(\delta(\epsilon(q_0), a)) = \epsilon(\delta(q_0, a)) = \epsilon(q_1) = \{q_1, q_2\}$

$\delta'(q_0, b) = \epsilon(\delta(\epsilon(q_0), b)) = \epsilon(\delta(q_0, b)) = \epsilon(\phi) = \phi$

$\delta'(q_1, a) = \epsilon(\delta(\epsilon(q_1), a)) = \epsilon(\delta(\{q_1, q_2\}, a)) = \epsilon(\delta(q_1, a) \cup \delta(q_2, a))$

$= \epsilon(\phi \cup \phi) = \phi$

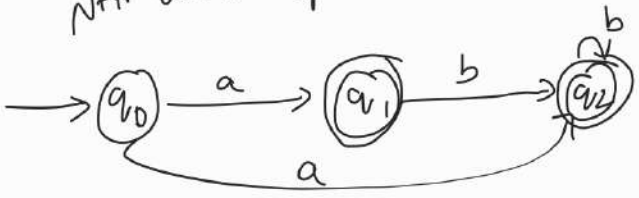
$\delta'(q_1, b) = \epsilon(\delta(\epsilon(q_1), b)) = \epsilon(\delta(\{q_1, q_2\}, b)) = \epsilon(\delta(q_1, b) \cup \delta(q_2, b))$

$= \epsilon(\phi \cup q_2) = \epsilon(q_2) = \{q_2\}$

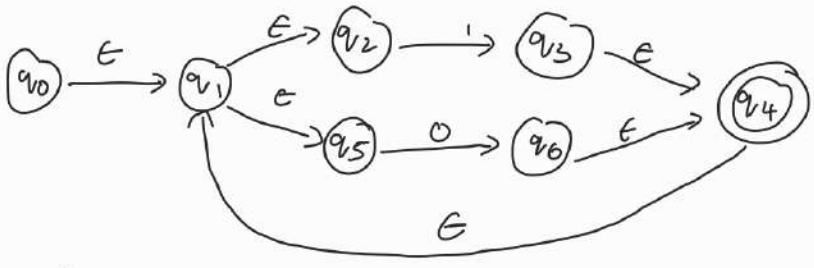
$\delta'(q_2, a) = \epsilon(\delta(\epsilon(q_2), a)) = \epsilon(\delta(q_2, a)) = \epsilon(\phi) = \phi$

$\delta'(q_2, b) = \epsilon(\delta(\epsilon(q_2), b)) = \epsilon(\delta(q_2, b)) = \epsilon(q_2) = \{q_2\}$

NFA without epsilon



Q] converting NFA with ϵ-transitions to DFA



$\epsilon(q_0) = \{q_0, q_1, q_2, q_5\}$

$= A$ (assume)

$\delta'(A, 0) = \epsilon(\delta(\epsilon(A, 0)) = \epsilon(\delta(\{q_0, q_1, q_2, q_5), 0)) = \epsilon(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0) \cup \delta(q_5, 0))$

$= \epsilon(\phi \cup \phi \cup \phi \cup q_6) = \epsilon(q_6) = \{q_6, q_4, q_1, q_2, q_5\} = B$

$\delta'(A, 1) = \epsilon(\delta(\epsilon(A, 1)) = \epsilon(\delta(\{q_0, q_1, q_2, q_5), 1)) = \epsilon(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1) \cup \delta(q_5, 1))$

$= \epsilon(\phi \cup \phi \cup q_3 \cup \phi) = \epsilon(q_3) = \{q_3, q_4, q_1, q_2, q_5\} = C$

$\delta'(B, 0) = \epsilon(\delta(\epsilon(B, 0)) = \epsilon(\delta(\{q_6, q_4, q_1, q_2, q_5\}, 0)) = \epsilon(\delta(q_6, 0) \cup \delta(q_4, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0) \cup \delta(q_5, 0)) = \epsilon(\phi \cup \phi \cup \phi \cup \phi \cup q_6)$

$= \epsilon(q_6) = \{q_6, q_4, q_1, q_2, q_5\} = B$

$\delta'(B, 1) = \epsilon(\delta(\epsilon(B, 1)) = \epsilon(\delta(\{q_6, q_4, q_1, q_2, q_5\}, 1)) = \epsilon(\delta(q_6, 1) \cup \delta(q_4, 1)$

$\cup \delta(q_1,1) \cup \delta(q_2,1) \cup \delta(q_5,1)) = \epsilon(\phi \cup \phi \cup \phi \cup q_3)$

$= \epsilon(q_3) = \{q_3, q_4, q_1, q_2, q_5\} = C$

$\delta'(C,0) = \epsilon(\delta(\epsilon(C,0)) = \epsilon(\delta(\{q_3,q_4,q_1,q_2,q_5\},0)) = \epsilon(\delta(q_3,0) \cup \delta(q_4,0)$

$\cup \delta(q_1,0) \cup \delta(q_2,0) \cup \delta(q_5,0)) = \epsilon(\phi \cup \phi \cup \phi \cup \phi \cup q_6) = \epsilon(q_6)$

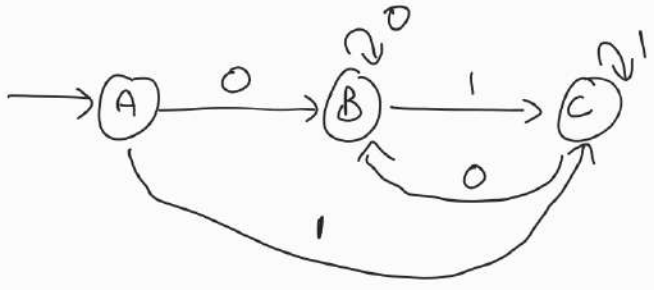
$= B$

$\delta'(C,1) = \epsilon(\delta(\epsilon(C,1)) = \epsilon(\delta(\{q_3,q_4,q_1,q_2,q_5\},1)) = \epsilon(\delta(q_3,1) \cup \delta(q_4,1)$

$\cup \delta(q_1,1) \cup \delta(q_2,1) \cup \delta(q_5,1)) = \epsilon(\phi \cup \phi \cup \phi \cup q_3 \cup \phi) = \epsilon(q_3)$

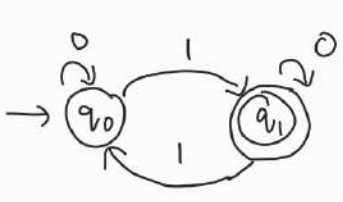
$= C$

DFA

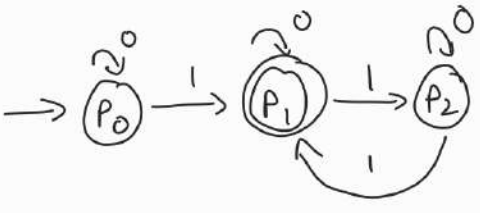


Equivalence of two finite state machines

Q)



m_1



m_2

intermediate states

	0	1
$\{q_0, P_0\}$	$\{q_0, P_0\}$	$\{q_1, P_1\}$
$\{q_1, P_1\}$	$\{q_1, P_1\}$	$\{q_0, P_2\}$
$\{q_0, P_2\}$	$\{q_0, P_2\}$	$\{q_1, P_1\}$

Both final states

∴ The m_1 and m_2 are equal

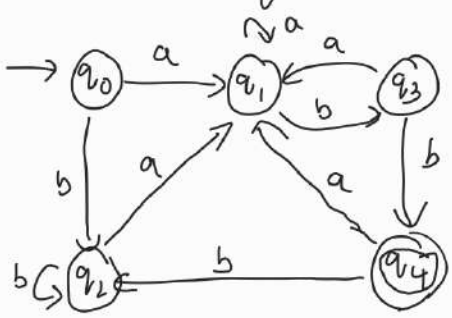
If we get a pair like $\{q_0, P_1\}$ then m_1 and m_2 are not equal

(q_0: Intermediate, P_1: final)

Minimization of DFA

1. Equivalence method (or) Partition method

2. Table filling method (or) myhill-nerode Theorem



Transition table

	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
q_4	q_1	q_2

0- Equivalence

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

1 - Equivalence

They are present in same set

(q_0, q_1)

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

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

$q_0 = q_1$

(q_0, q_2)

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

$\delta(q_0, b) = q_2$, $\delta(q_2, b) = q_3$

$q_0 = q_2$

(q_0, q_3)

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

$\delta(q_0, b) = q_2$, $\delta(q_2, b) = q_4$ ✗

$q_0 \neq q_3$

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

Compare with 1-equivalence

2-Equivalence

(q_0, q_1)

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

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

$q_0 \neq q_1$

(q_0, q_2)

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

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

$q_0 = q_2$

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

calculate based on 2-equivalence

3 - Equivalence

(q_0, q_2)

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

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

$q_0 = q_2$

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

DFA

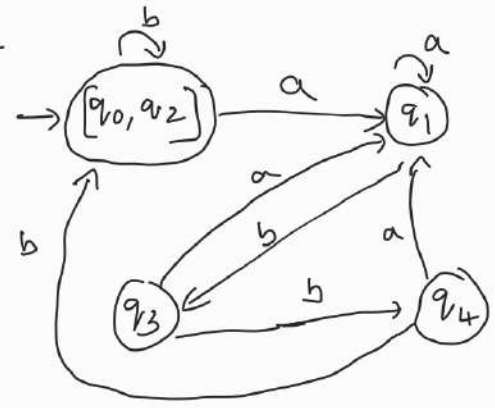
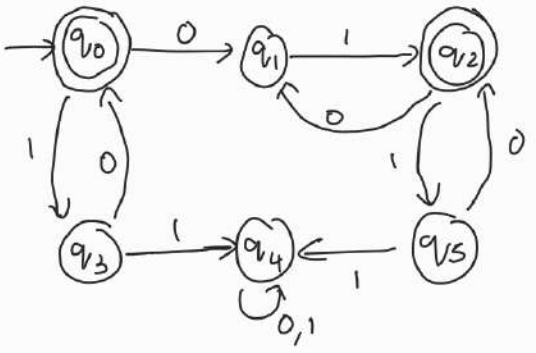


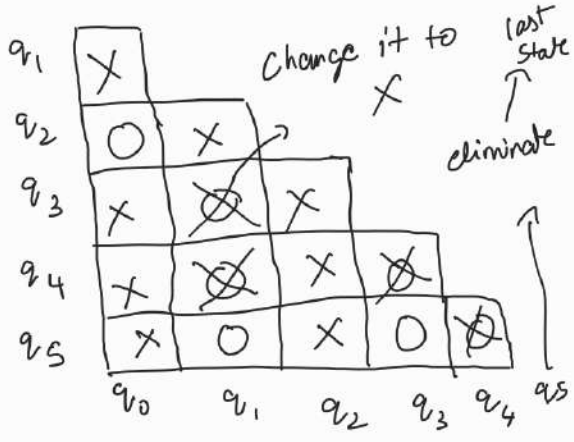
Table filling method (or) myhill - Nerode Theorem



Transition Table

	0	1
→ q_0	q_1	q_3
q_1	q_4	q_2
q_2	q_1	q_5
q_3	q_0	q_4
q_4	q_4	q_4
q_5	q_2	q_4

q_0 → eliminate first state



(f.s, n.fs)

Final state — non-final state

(q_2, q_0)

$\delta(q_2, 0) = q_1$ $\delta(q_2, 1) = q_5$

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

(q_3, q_1)

$\delta(q_3, 0) = q_0$ $\delta(q_3, 1) = q_4$

$\delta(q_1, 0) = q_4$ $\delta(q_1, 1) = q_4$

(q_5, q_1)

$\delta(q_5, 0) = q_2$

$\delta(q_1, 0) = q_4$

(q_5, q_3)

$\delta(q_5, 0) = q_2$

$\delta(q_3, 0) = q_0$

(q_4, q_1)

$\delta(q_4, 0) = q_4$ $\delta(q_4, 1) = q_4$

$\delta(q_1, 0) = q_4$ $\delta(q_1, 1) = q_2$

(q_4, q_3)

$\delta(q_4, 0) = q_4$ $\delta(q_4, 1) = q_4$

$\delta(q_3, 0) = q_0$ $\delta(q_3, 1) = q_4$

$\delta(q_5, 1) = q_4$

$\delta(q_3, 1) = q_4$

(q_5, q_4)

$\delta(q_5, 0) = q_2$

$\delta(q_4, 0) = q_4$

unmarked

(q_0, q_2)

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

$\delta(q_2, 1) = q_1$ $\delta(q_2, 1) = q_5$

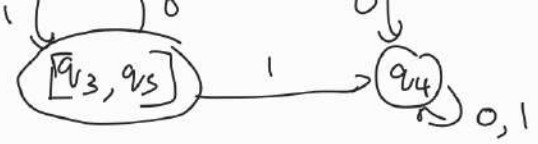
(q_3, q_5)

$\delta(q_3, 0) = q_0$ $\delta(q_3, 1) = q_4$

$\delta(q_5, 0) = q_2$ $\delta(q_5, 1) = q_4$

(q_0, q_1) q_1 (q_3, q_5) q_4

4 states



Finite Automata with Output

1. Moore machine

6 tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

Q – finite set of states

Σ – Input Alphabet

Δ – output Alphabet

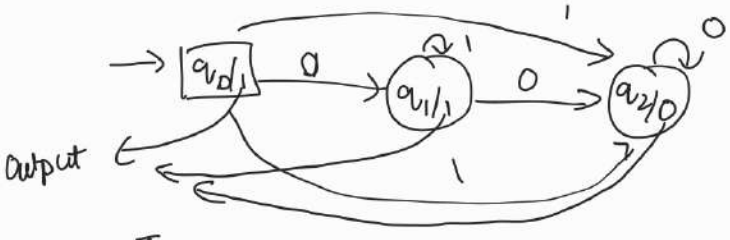
δ – transition function

$Q \times \Sigma \to Q$

λ – output function

$Q \to \Delta$

$q_0 \to$ Initial State



Transition table

Current State	Next state		Output
	0	1	
q_0	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

- moore machine & mealy machine doesn't contain any final State

Input string = n

then output string = n+1

Input = 0110 → ε is applied

$\delta(q_0, 0110) \to 1$ → output of q_0

$\delta(q_0, \underline{0}110) \to 1$

$\delta(q_1, \underline{1}10) \to 1$

$\delta(q_1, \underline{1}0) \to 1$

$\delta(q_1, 0) \to 1$

→ output string = 11111

q_2

2. Mealy machine

6 tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

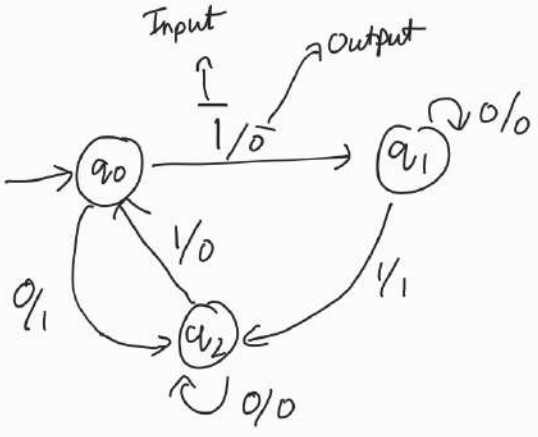
Q – finite set of states

Σ – Input Alphabet

Δ – output Alphabet

δ – transition function

$Q \times \Sigma \to Q$



Current State	0		1	
	ns	o/p	ns	o/p
q_0	q_0	1	q_1	0

λ — output function

$Q \times \Sigma \rightarrow \Delta$

$q_0 \rightarrow$ Initial state

Input string = n

then output string = n

q_0	q_2	1	q_1	0
q_1	q_1	0	q_2	1
q_2	q_2	0	q_0	0

ex:- 1001 = input

$\delta(q_0, 1001) \rightarrow 0$

$\delta(q_1, 001) \rightarrow 0$

$\delta(q_1, 01) \rightarrow 0$

$\delta(q_1, 1) \rightarrow 1$

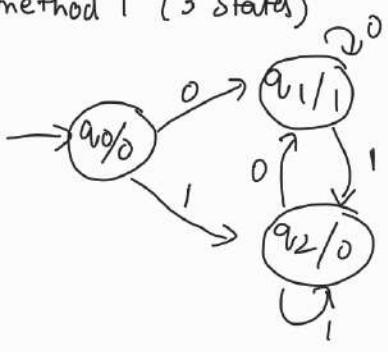
$q_2 \rightarrow$ ends at q_2

} → output is 0001

Q) Design a moore machine to find 1's complement of a given binary number.

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

method 1 (3 states)



ex: string 1011

$q_0\ q_2\ q_1\ q_2\ q_2$

0 0 1 0 0 → output

neglect

method 2 (2 state)

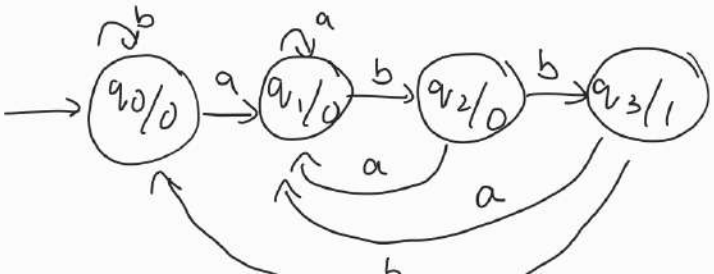
Ex: 1 0 1 1

$q_0\ q_0\ q_1\ q_0\ q_0$

0 0 1 0 0 → output

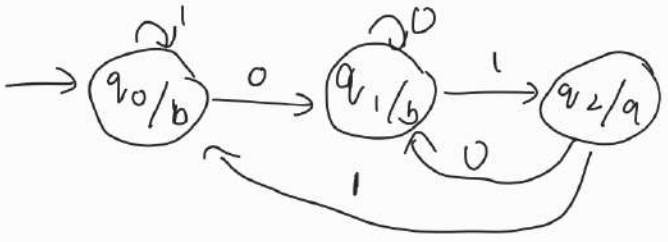
Q) Design a moore machine that counts number of occurences of 'abb'

$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$



Q) Design a moore machine that prints 'a' when even the sequence '01' is encountered in any input string

$\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$

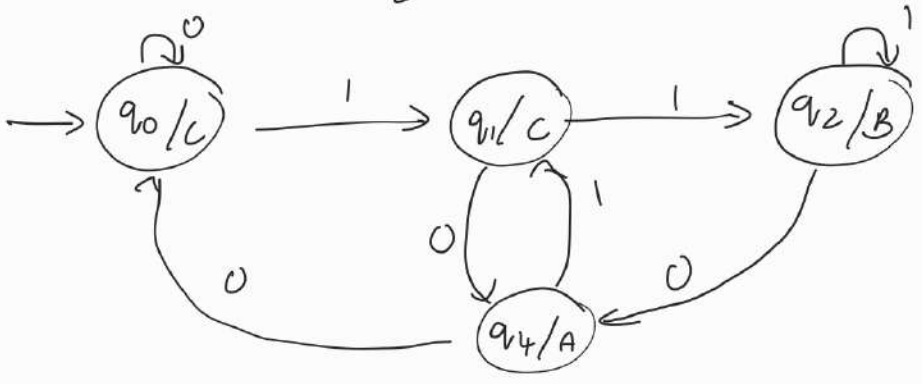


Ex:- 1 0 1 0 0 0 1
q_0 q_0 q_1 q_2 q_1 q_2 q_1 q_2
2 times

Q) Design a moore machine that takes set of all string over $\{0, 1\}$ and produces 'A' as output if Input ends with '10' or produces 'B' as O/P is I/P ends with '11' otherwise produces 'C' as O/P

$\Sigma = \{0, 1\}$ $\Delta = \{A, B, C\}$

A → 10
B → 11
C → other

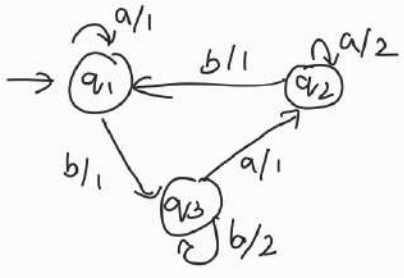


Ex:-
0 0 1 0
q_0 q_0 q_0 q_1 q_4
C C C C A

0 1 0 1 0 1 1
q_0 q_0 q_1 q_4 q_1 q_4 q_1 q_2

Converting mealy machine to moore machine

Q)



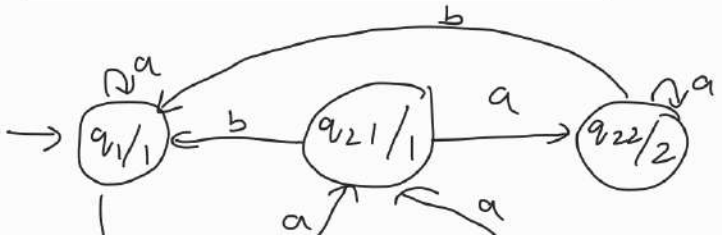
Transition table

current state	a		b	
	ns	o/p	ns	o/p
→ q_1	q_1	1	q_3	1
q_2	q_2	2	q_1	1
q_3	q_2	1	q_3	2

$q_1 \rightarrow 1$
$q_2 \rightarrow q_2\ 1$
$\rightarrow q_2\ 2$
$q_3 \rightarrow q_3\ 1$
$\rightarrow q_3\ 2$

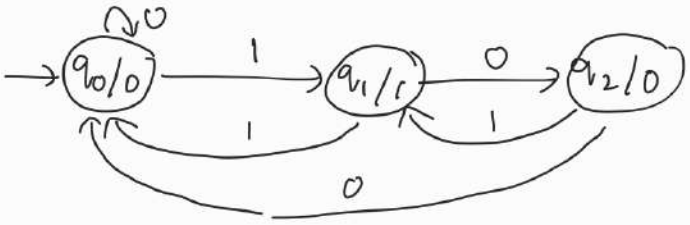
moore machine T T

CS	ns		o/p
	a	b	
q_{11}	q_{11}	q_{31}	1
q_{21}	q_{22}	q_{11}	1
q_{22}	q_{22}	q_{11}	2
q_{31}	q_{21}	q_{31}	2



q_{31}	q_{21}	q_{32}	1
q_{32}	q_{21}	q_{32}	2

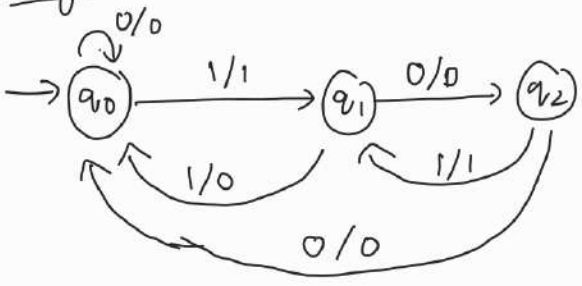
Conversion of moore machine to Mealy Machine



Transition table

Current state	next state		output
	0	1	
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_0	q_1	0

mealy machine



mealy machine Transition Table

Current state	0		1	
	ns	op	ns	op
q_0	q_0	0	q_1	1
q_1	q_2	0	q_0	0
q_2	q_0	0	q_1	1

UNIT-2 Regular Expressions

- Regular language means language accepted by the finite automata.
- Regular expression means a patter an algebraic expression.

Let Σ be one input alphabet then Regular expression (RE) can be defined as

1. ϕ is a RE which denotes empty set $\{\}$
2. $\in$ is a RE which denotes null string $\{\in\}$
3. let 'a' be an I/P symbol in Σ, then 'a' is a RE
4. if R_1 is a RE then R_1^* is also a RE
5. if R_1 is a RE then R_1^+ is also a RE

6. If R_1 & R_2 are RE then $R_1 + R_2$ is also a RE.
union

7. If R_1 & R_2 are RE then $R_1 \cdot R_2$ is also a RE.
concatenate

Q) Design a RE to accept all possible combination of a's & b's over $\Sigma = \{a, b\}$

$L = \{ \epsilon, a, b, ab, aa, bb, aba, \ldots \}$ $\quad RE = (a+b)^*$

Q) Ends with 00 over $\Sigma = \{0, 1\}$

$L = \{ 00, 100, 1100, 1000, 0100 \ldots \}$ $\quad RE = (0+1)^* 00$

Q) Starts with 1 & end with 0

$L = \{ 10, 1000, 1010, \ldots \}$ $\quad RE = 1 (0+1)^* 0$

Q) Any no. of a's followed by any no. of b's followed by Any no. of c's

$L = \{ \epsilon, abc, aabbcc, aabc, \ldots \}$ $\quad RE = a^* b^* c$ → because of G

Q) Atleast 1a followed by atleast one B followed by atleast one C.

$L = \{ abc, aabbcc \ldots \}$ $\quad RE = a^+ b^+ c^+$ → because no G

Q) Begins (or) ends with either 00 or 11

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

Q) 3rd character from right end is 'a' over $\Sigma = \{a, b\}$

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

Q) Atleast 2 b's over $\Sigma = \{a, b\}$

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

Q) Exactly 2b's.

$RE = a^* b a^* b a^*$

Q) even length string over $\Sigma = \{0,1\}$

$RE = (00)^*$

Q) odd length string over $\Sigma = \{0,1\}$

$RE = 1(11)^*$

Q) No. of a's are divisible by 3

$RE = (b^*a\,b^*\,a\,b^*\,a\,b^*)^*$

Q) atmost 2 a's

$RE = 0a \to b^*$

$1a \to b^*\,a\,b^*$

$2a \to b^*\,a\,b^*\,a\,b^*$

Algebraic laws of Regular Expression

1. $\phi + R = R$
2. $\phi \cdot R = R \cdot \phi = \phi$
3. $\epsilon \cdot R = R \cdot \epsilon = R$
4. $\epsilon^* = \epsilon$ & $\phi^* = \epsilon$
5. $R + R = R$
6. $R^* \cdot R^* = R^*$
7. $R \cdot R^* = R^* \cdot R = R^+$
8. $(R^*)^* = R^*$
9. $\epsilon + RR^* = \epsilon + R^*R = R^*$
10. $(PQ)^*P = P(QP)^*$
11. $(P+Q)^* = (P^* \cdot Q^*)^* = (P^* + Q^*)^*$
12. $(P+Q) \cdot R = P \cdot R + Q \cdot R$

Simplification of Regular Expression

Q] $(1 + 00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1) = 0^*1(0+10^*1)^*$

$(1+00^*1)\left(\epsilon + \underbrace{(0+10^*1)^*}_{R^*}\underbrace{(0+10^*1)}_{R}\right)$

$\epsilon +$

$(1+00^*1)(0+10^*1)^*$

$1(\underbrace{\epsilon + 00^*}_{RR^*})(0+10^*1)^* \Rightarrow 0^*1(0+10^*1)^*$

Q] $\epsilon + \underbrace{1^*(011)^*}_{R}\underbrace{(1^*(011)^*)^*}_{R^*} = (1+011)^*$

$(P^*Q^*)^*$

$= (P+Q)^* \leftarrow (\underset{P}{1^*}\underset{Q}{(011)^*})^* \Rightarrow (1+011)^*$

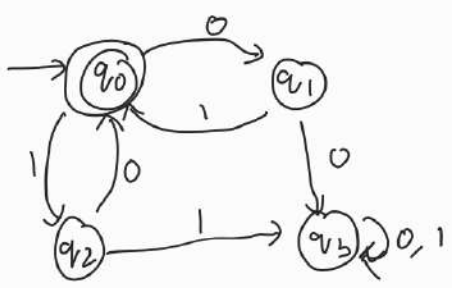
Conversion of finite Automata to Regular expression

Converting

Steps:

1. Construct State equations for all the States based on incoming edges.
2. Add ϵ to one equation to Initial State
3. Simplify the equation using Arden's theorem & find Regular expression.

Ex:



$q_0 = q_1 \cdot 1 + q_2 \cdot 0 + \epsilon$ ← incoming edges

$q_1 = q_0 \cdot 0$

$q_2 = q_0 \cdot 1$

$q_3 = q_1 \cdot 0 + q_2 \cdot 1 + q_3 \cdot 0 + q_3 \cdot 1$

$R = Q + RP$

$R = QP^*$

$q_0 = q_0 \cdot 0 \cdot 1 + q_0 \cdot 1 \cdot 0 + \epsilon$

$q_0 = q_0 (0 \cdot 1 + 1 \cdot 0) + \epsilon$

R R P Q

$q_0 = \epsilon \cdot (0 \cdot 1 + 1 \cdot 0)^*$

$q_0 = (0 \cdot 1 + 1 \cdot 0)^*$

Ex:

$q_1 = q_1 \cdot 1 + q_1 \cdot 0 \cdot 1 + \epsilon$

$q_1 = q_1 (1 + 0 \cdot 1) + \epsilon$

R R P Q

$q_1 = \epsilon \cdot (1 + 0 \cdot 1)^* = (1 + 0 \cdot 1)^*$

$q_1 = q_1 \cdot 1 + q_2 \cdot 1 + \epsilon$

$q_2 = q_1 \cdot 0$

$q_3 = q_2 \cdot 0 + q_3 \cdot 0 + q_3 \cdot 1$

$q_3 = q_1 \cdot 0 + q_3 \cdot 0 + q_3 \cdot 1$

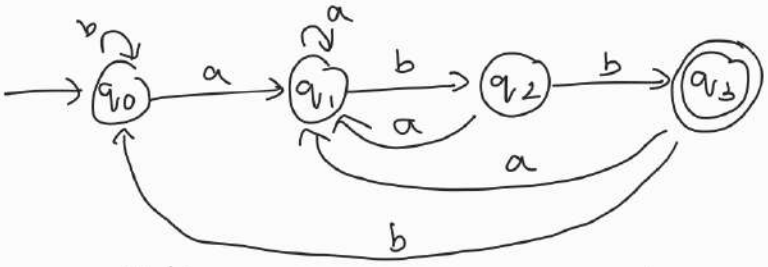
$q_3 = q_1 \cdot 0 + q_3 (0 + 1)$

$q_3 = (1 + 0 \cdot 1)^* 0 + q_3 (0 + 1)$

R Q R P

$q_3 = (1 + 0 \cdot 1)^* 0 \ (0 + 1)^*$

Q) Construct Regular Expression for the given finite Automata (DFA) using Arden's Algorithm.



$q_0 = q_0 \cdot b + q_3 \cdot b + \epsilon$

$q_1 = q_0 \cdot a + q_1 \cdot a + q_2 \cdot a + q_3 \cdot a$

$q_2 = q_1 \cdot b$

$q_3 = q_2 \cdot b$

Consider q_1 state substituting q_2 & q_3 values in q_1

$q_1 = q_0 \cdot a + q_1 \cdot a + q_1 \cdot b \cdot a + q_2 \cdot b \cdot a$

$= q_0 \cdot a + q_1 \cdot a + q_1 \cdot b \cdot a + q_1 \cdot b \cdot b \cdot a$

$q_1 = q_0 a + q_1 (a + ba + bba)$

Consider the State q_2

$q_2 = q_1 b$

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

$R = \underset{R}{Q} + \underset{R}{R}\underset{P}{P}$

$q_1 = q_0 a (a + ba + bba)^*$

$R = QP^*$

Simplify q_0

$q_0 = q_0 b + q_3 b + \epsilon$

$q_0 = q_0 b + q_0 a (a+ba+bba)^* bbb + \epsilon$

$\underset{R}{q_0} = \underset{R}{q_0}\underset{P}{(b + a(a+ba+bba)^* bbb)} + \underset{Q}{\epsilon}$

$\epsilon P^* = P^*$

$q_0 = (b + a(a+ba+bba)^* bbb)^*$

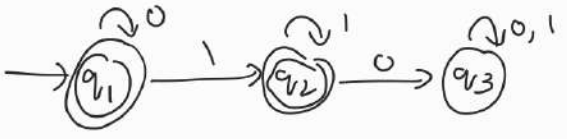
Consider q_3

$q_3 = q_2 \cdot b$

$q_3 = q_0 a (a+ba+bba)^* b b$

result → $q_3 = (b + a(a+ba+bba)^* bbb)^* a (a+ba+bba)^* bb$

Converting of finite Automata to Regular Expression using Arden's Theorem



$q_1 = q_1 \cdot 0 + \epsilon$

$q_2 = q_1 \cdot 1 + q_2 \cdot 1$

$q_3 = q_2 \cdot 0 + q_3 \cdot 0 + q_3 \cdot 1$

$\underset{R}{q_1} = \underset{R}{q_1}\cdot\underset{P}{0} + \underset{Q}{\epsilon}$

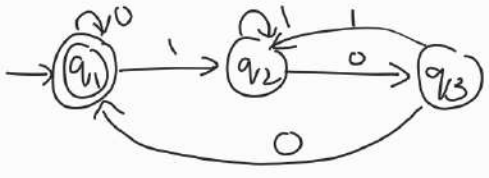
$q_1 = \epsilon 0^* = 0^*$

$\underset{R}{q_2} = \underset{Q}{0^* \cdot 1} + \underset{R}{q_2}\cdot\underset{P}{1}$

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

$q_1 + q_2 = 0^* + 0^* 11^*$

Conversion of DFA To Regular expression



$q_2 = q_1 \cdot 1 + q_2 \cdot 1 + q_3 \cdot 1$

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

$\underset{R}{q_2} = \underset{Q}{q_1 \cdot 1} + \underset{R}{q_2}\underset{P}{(1 + 0 \cdot 1)}$

$q_2 = q_1 \cdot 1 (1+0\cdot1)^*$

$q_1 = q_1 \cdot 0 + q_3 \cdot 0 + \epsilon$

$q_2 = q_1 \cdot 1 + q_2 \cdot 1 + q_3 \cdot 1$

$q_3 = q_2 \cdot 0$

$q_1 = q_1 \cdot 0 + q_3 \cdot 0 + \epsilon$

$q_1 = q_1 \cdot 0 + q_2 \cdot 0 \cdot 0 + \epsilon$

$q_1 = q_1 \cdot 0 + q_1 \cdot 1 (1+0\cdot1)^* 0 \cdot 0 + \epsilon$

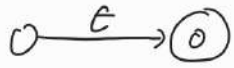
$\underset{R}{q_1} = \underset{R}{q_1}\underset{P}{(0 + 1(1+0\cdot1)^* 00)} + \underset{Q}{\epsilon}$

$q_1 = \epsilon \cdot (0 + 1(1+0\cdot1)^* 00)^*$

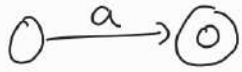
$q_1 = (0 + 1(1+0\cdot1)^* 00)^*$

Converting Regular Expression to Finite Automata

ϵ

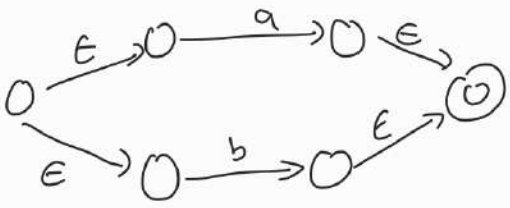


a

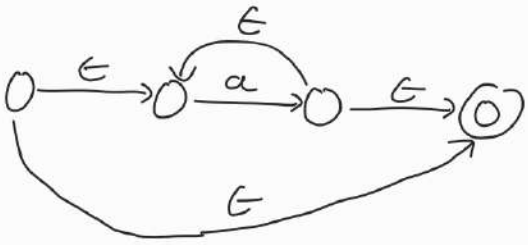


a.b

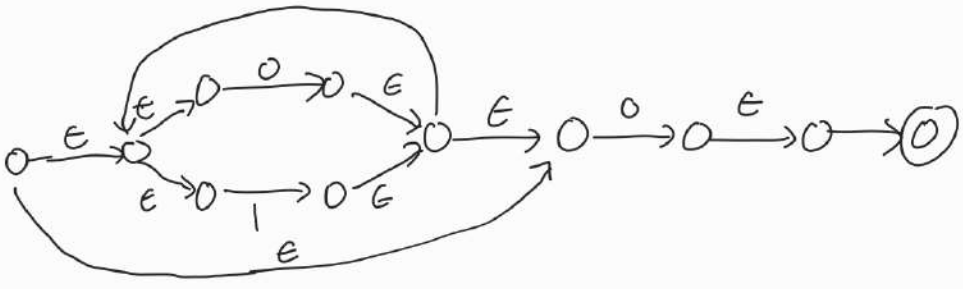
a+b



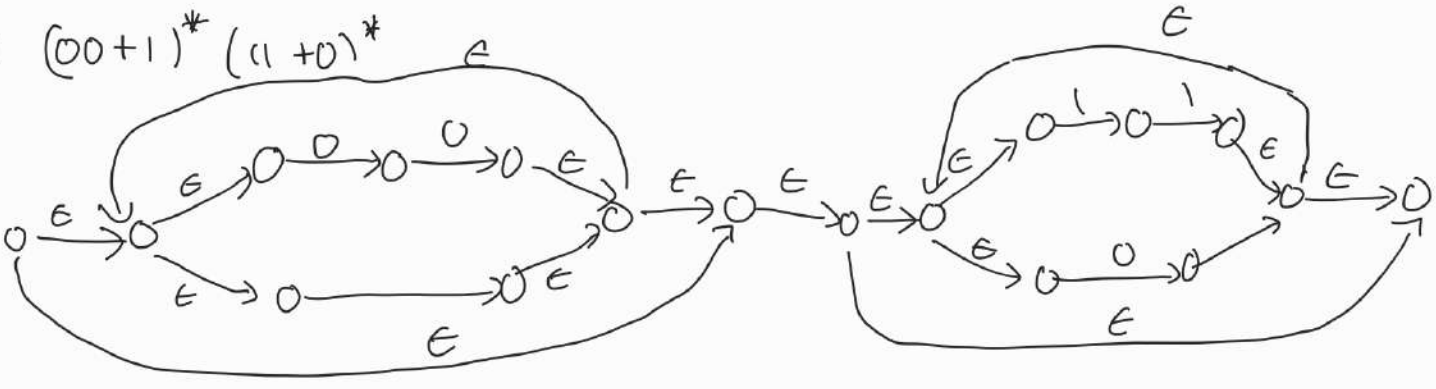
a*



Q] $(0+1)^*\cdot 0\cdot 1$

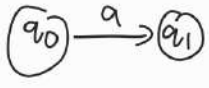


Q) $(00+1)^*(11+0)^*$

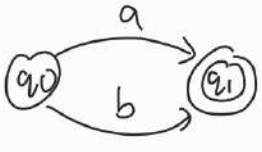


Converting Regular Expression to Finite Automata using Direct method

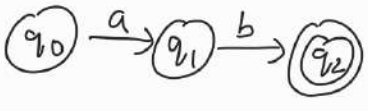
a



a+b



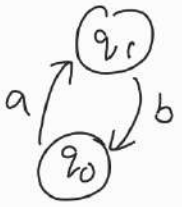
a.b



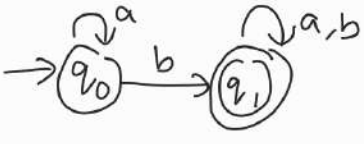
a*

(a+b)*

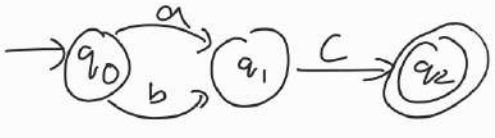
(ab)*



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

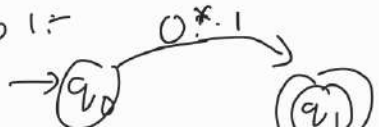


Q] $(a+b)c$

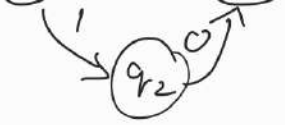


Q) 0^*1+10

step 1:-

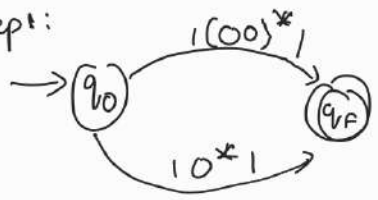


step 2:

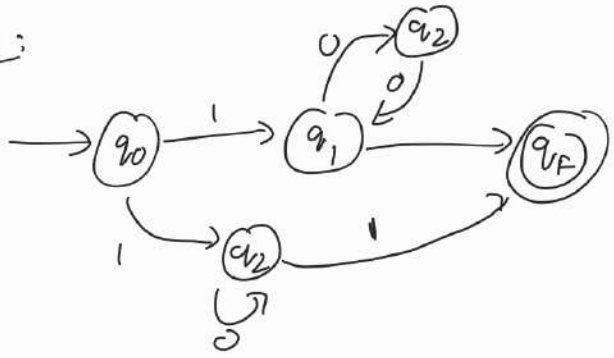


Q] $1(00)^*1 + 10^*1$

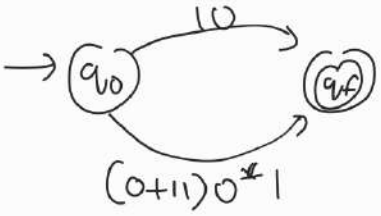
Step1:

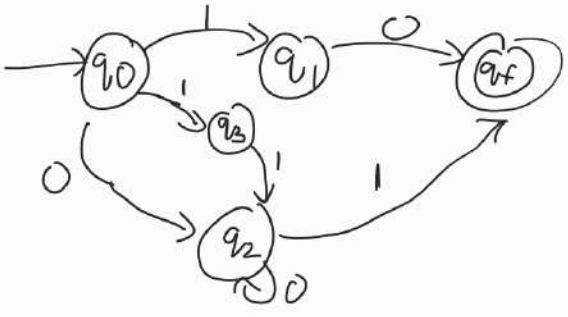


Step2:



Q] $10 + (0+11)0^*1$





UNIT-3 Grammars

4 tuples

$G = (V, T, P, S)$

↓ Grammar

- S → start symbol $A \to a$
- P → Set of Production Rules $\alpha \to \beta$
- T → set of Terminal Symbol
- V → Set of variable (or) non terminals

1. Left linear Regular grammar → $A \to B\alpha$ → non-terminal or variable
2. Right linear Regular grammar → $A \to \alpha B$ → non-terminal or variable

Converting FA to Right linear Grammar

Steps for left linear Grammar

1. Construct Reverse of FA.
2. then write right linear grammar.
3. then take reverse of Right linear grammar.

Right linear → 

$A \to aB$

Converting FA to Left linear Grammar

$B \to aB$

$B \to bB$

$B \to \epsilon$

→ final state

$B \to aB$ | $B \to Ba$

$B \to bB$ | $B \to Bb$

$B \to aA$ | $B \to Aa$

$A \to \epsilon$ | $A \to \epsilon$

Construct Right linear grammar to LLG

$S \to bB$

$B \to bC$

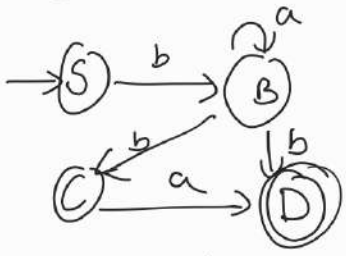
$B \to aB$

$C \to a$

$B \to b$

Step 1:-

Construct FA

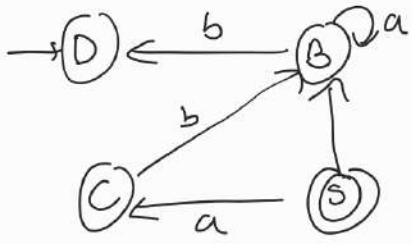


assume it as final state

Step 2:

Exchange Initial state & final state

Reverse the edges



Step 3:- write RLG

$S \to bB$

$S \to aC$

$C \to bB$

$B \to aB$

$B \to bD$

Step 4:- write LLG

$S \to Bb$

$S \to Ca$

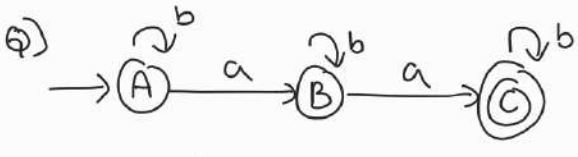
$C \to Bb$

$B \to Ba$

$B \to Db$

Construction of Regular Grammar from finite Automata

Q)



$A \to bA$

$A \to aB$

$B \to aC$

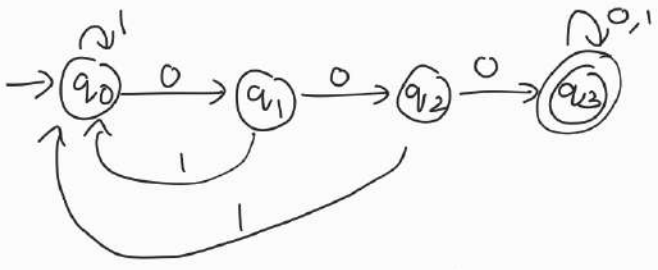
$\to B \to a$ → Reaches final state

$B \to bB$

$C \to bC$

$\to C \to b$

Q)



$q_0 \to 1q_0$

$q_0 \to 0q_1$

$q_1 \to 0q_2$

$q_1 \to 1q_0$

$q_2 \to 0q_3$

$q_2 \to 0$

$q_2 \to 1q_0$

$q_3 \to 0q_3$

$q_3 \to 0$

$q_3 \to 1q_3$

$q_3 \to 1$

(UNIT – 2)

Pumping lemma for Regular languages

- Pumping lemma is used to prove that a language is not Regular.

Let 'L' be a Regular language. Then there is a constant 'n' & select a string 'z' such that $|z| \geq n$.

Divide 'z' into 3 parts $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$ and for all $i \geq 0$, uv^iw is in L.

Q) prove that $L = \{a^n b^n \mid n \geq 1\}$ is not Regular.

Let L be a Regular Language $L = \{ab, aabb, aaabbb, \ldots\}$

Let $n = 4$ $z = aabb$, $|z| = 4 \geq 4$ ← this is n

$u = a$, $v = ab$, $w = b$

$|uv| \leq n$ $\quad |v| \geq 1$ $\quad uv^iw$ → $i \geq 0$

$|a \cdot ab| \leq n$ $\quad |ab| \geq 1$ $\quad a \cdot (ab)^0 \cdot b \Rightarrow ab \in L$

$3 \leq 4$ $\quad 2 \geq 1$ $\quad a \cdot (ab)^1 \cdot b \Rightarrow a \cdot ab \cdot b \in L$

$a \cdot (ab)^2 \cdot b \Rightarrow a \cdot ab \cdot ab \cdot b \notin L$

∴ Hence proved that $L = \{a^n b^n \mid n \geq 1\}$ is not Regular.

Q) prove that $L = \{a^p / p$ is a prime$\}$ is not Regular

let L is a Regular language, 'n' is an Intger constant. Select a string 'w' from L such that $|z| \geq n$

$L = \{aa, aaa, aaaaa, \ldots\}$ $|z| \geq n$

Let $n = 3$

$w = aaa$ such that $|z| \geq n$

$|aaa| \geq 3$

$3 \geq 3$

Divide z into 3 parts such as uvw that $|uv| \leq n$, $|v| \geq 1$, for $i \geq 0$, uv^iw is in L

$z = \underset{u}{a}\,\underset{v}{a}\,\underset{w}{a}$

$|a \cdot a| \leq 3$ $\quad |a| \geq 1$

$2 \leq 3$ $\quad 1 \geq 1$

uv^im $\quad i=0 \Rightarrow a \cdot a^0 \cdot a \Rightarrow aa \Rightarrow aa \in L$

$i=1 \Rightarrow a \cdot a^1 \cdot a \Rightarrow aaa \in L$

$i=2 \Rightarrow a \cdot a^2 \cdot a \Rightarrow aaaa \notin L$

Q) show that a language of balanced parenthesis is not regular.

4) Show that a language of balanced parentheses is not regular.

$L = \{ (), (()), ()(), \ldots \}$

Let $n = 2$, $z = (())$ $|z| \geq n$

Divide 'z' into 3 parts such as uvw such that $|uv| \leq n$, $|v| \geq 1$ for $i \geq 0$, $uv^i w$ in L

$|(())| \geq 2$

$4 \geq 2$

$\underbrace{(}_{u}\underbrace{(}_{v}\underbrace{))}_{w}$

$|uv| \leq n$

$|(( )| \leq 2$

$2 \leq 2$

$|v| \geq 1$

$|(| \geq 1$

$1 \geq 1$

$uv^i w \Rightarrow i = 0 = ((^{\circ}))$

$\Rightarrow ( )) \notin L$

Closure Properties Of Regular languages

- Regular languages are closed under union, concatenation, Kleene closure, complement, Intersection & Reversed.

Union :- L_1 & L_2 are 2 R La's

$L_1 \cup L_2$ is also a Regular Lang's

Concatenation :- $L_1 \cdot L_2$

$L_1 \cdot L_2 = a^* \cdot b^*$

Kleene closure :- L is a R.Lang then L^*

$L = a$ $L = a^*$

Complement :- $\bar{L}$ is a Reg. Lang

L = set of all strings containing 1100 as substring

$\bar{L}$ = " " " " not " 1100 " " .

Intersection :- L_1 & L_2 the $L_1 \cap L_2$

$L_1 = a^*$ $L_2 = b^*$ $L_1 \cap L_2 = a^* \cap b^*$

Reversal

$L =$ 

Reversal 

UNIT-3 Context free Grammar

- 4-tuples $G = (V, T, P, S)$

where

V is set of variables or non-terminals upper case letters

T is set of terminals or symbols lower case letters

P is set of production Rules

$A \to \alpha$, where $A \in V$, $\alpha \in (V \cup T)^*$

S is start symbol

ex: Consider the grammar $E \to E + E \mid E \times E \mid (E) \mid id$ obtain id + id * id

S 1 2 3 4

Terminals

$E \to E + E$ ($\because E \to E + E$)

$E \to id + E$ ($\because E \to id$)

$E \to id + E * E$ ($\because E \to E * E$)

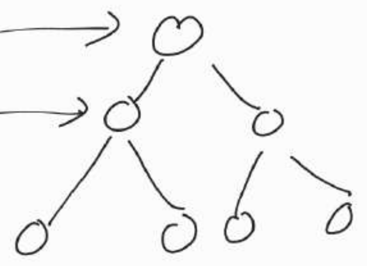
$E \to id + id * E$ [$\because E \to id$]

$E \to id + id * id$

Derivation : Left most derivation, Right most derivation.

- Derivation tree is diagrammatic representation of derivation

1. Root node → start symbol
2. Internal nodes → non terminals
   ↳ nodes with children
3. leaf nodes → terminals.



Q) $E \to E + E \mid E * E \mid (E) \mid id$ obtain id + id * id

LMD

LMD

RMD

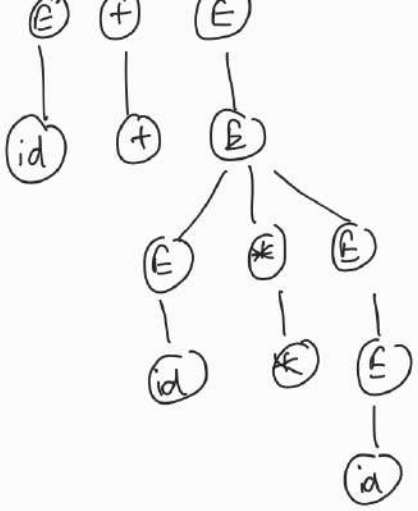
$E \to E + E$

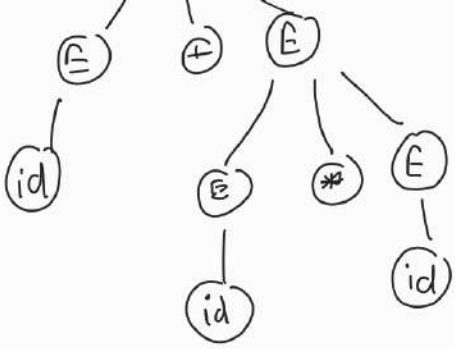
$E \rightarrow id + E$

$E \rightarrow id + E * E$

$E \rightarrow id + id * E$

$E \rightarrow id + id * id$





RMD

$E \rightarrow E + E$

$E \rightarrow E + E * E$

$E \rightarrow E + E * id$

$E \rightarrow E + id * id$

$E \rightarrow id + id * id$

Construction of CFG for a Language

Q) Any number of a's over one set $\Sigma = \{a\}$

$L = \{\epsilon, a, aa, aaa, \ldots\}$ $\quad RE = a^*$

Production Rule (P): $S \rightarrow aS$, $S \rightarrow \epsilon$ $\quad S \rightarrow aS \mid \epsilon$ $\quad G = (\{S\}, \{a\}, P, S)$

Q) Any no. of a's & b's $\quad \Sigma = \{a, b\}$

$L = \{\epsilon, a, b, ab, aab, abb, \ldots\}$ $\quad RE = (a+b)^*$

P: $S \rightarrow aS \mid bS \mid \epsilon$ $\quad G = (\{S\}, \{a, b\}, P, S)$

↳ or

Ex: aab

$S \rightarrow aS$

$S \rightarrow aaS$

$S \rightarrow aabS$

$S \rightarrow aab\epsilon$

Q) Containing strings of atleast 2a's $\quad \Sigma = \{a, b\}$

$L = \{aa, aaa, aaaabb, \ldots\}$ $\quad RE = (a+b)^* a (a+b)^* a (a+b)^*$

P: $S \rightarrow AaAaA$

$A \rightarrow aA \mid bA \mid \epsilon$ $\quad G = (\{S, A\}, \{a, b\}, P, S)$

Q) one occurence of 000

$L = \{000, 1000, 0001, \ldots\}$ $\quad RE = (0+1)^* 000 (0+1)^*$

P: $S \to A000A$

$A \to 0A \mid 1A \mid \in$

$G = (\{S, A\}, \{0, 1\}, P, S)$

Q) equal no. of a's & b's $\Sigma = \{a, b\}$

$L = \{ab, ba, abba, \ldots\}$

P : $S \to aSbS \mid bSaS \mid \in$

Q) construction of CFG for $(0+1)^*$

$\Sigma = \{0, 1\}$ $L = \{\in, 0, 1, 01, 10, 11 \ldots\}$

P: $S \to 0S \mid 1S \mid \in$

$G = (\{S\}, \{0, 1\}, P, S)$

Q) accepts palindromes $\Sigma = \{a, b\}$

$L = \{\in, a, b, aa, bb, aba, baab, \ldots\}$

P: $S \to aSa \mid bSb \mid a \mid b \mid \in$ $G = (\{S\}, \{a, b\}, P, S)$

ex: abba

$S \to aSa$

$S \to abSba$

$S \to ab \in ba$

$S \to abba$

$S \to aSa \mid bSb \mid \in$ [even]

$S \to aSa \mid bSb \mid a \mid b$ [odd]

ex: aba

$S \to aSa$

$S \to aba$

Q) CFG for $L = \{wcw^R \mid w \in \{a, b\}^*\}$

P: $S \to aSa \mid bSb \mid c$

$G = (\{S\}, \{a, b, c\}, P, S)$

Q) exactly one 'a' $\Sigma = \{a, b\}$

$L = \{a, bab, bbba, \ldots\}$ $RE = b^*ab^*$

P: $S \to BaB$

$B \to \in \mid bB$

$G = (\{S, B\}, \{a, b\}, P, S)$

Q) Atleast one 'a' $\Sigma = \{a, b\}$

$RE = (a+b)^* \; a \; (a+b)^*$

P: $S \to AaA$

$A \to aA \mid bA \mid \epsilon$

$G = (\{S, A\}, \{a, b\}, P, S)$

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

$S \to A^* \mid B^*$

$A \to 0A \mid \epsilon$

$B \to 0B \mid 1B \mid \epsilon$

Simplification of CFG / Removal of Unit Production

Q) $S \to 0A \mid 1B \mid C$

$A \to 0S \mid 00$

$B \to 1 \mid A$

$C \to 01$

unit productions :- $S \to C$, $B \to A$

Eliminate $S \to C$, $C \to 01 \Rightarrow S \to 01$

Eliminate $B \to A$, $A \to 0S \mid 00 \Rightarrow B \to 0S \mid 00$

Final grammar

$S \to 0A \mid 1B \mid 01$

$A \to 0S \mid 00$

$B \to 1 \mid 0S \mid 00$

$C \to 01$

- If a production is the form $A \to B$ & $B \to x_1 x_2 \ldots x_n$ then write the production as $A \to x_1 x_2 \ldots x_n$

Q) $S \to AB$

$A \to a$

$B \to C \mid b$

$C \to D$

$D \to E \mid bC$

$E \to d \mid Ab$

unit product

$B \to C$

$C \to D$

$D \to E$

Eliminate $D \to E$, $E \to d \mid Ab$

$\Rightarrow D \to d \mid Ab \mid bC$

Eliminate $C \to D$, $D \to E$

$\Rightarrow C \to E \Rightarrow C \to d \mid Ab$

Eliminate $B \to C$, $C \to D \Rightarrow B \to D \Rightarrow B \to d \mid Ab \mid bC \mid b$

Final grammar

$S \to AB$

$A \to a$

$B \to d \mid Ab \mid bC \mid b$

$C \to d \mid Ab$

$D \rightarrow d \mid Ab \mid bC$

$E \rightarrow d \mid Ab$

Removal of NULL (Epsilon)

Q) $S \rightarrow XYX$

$X \rightarrow 0X \mid \epsilon$

$Y \rightarrow 1Y \mid \epsilon$

Identify NULL production

$X \rightarrow \epsilon$

$Y \rightarrow \epsilon$

To eliminate $X \rightarrow \epsilon$, Replace each occurence of X with ϵ.

$S \rightarrow XY\bar{X} \mid YX \mid XY \mid Y$

ϵ YX

$X \rightarrow 0X \mid 0$

$Y \rightarrow 1Y \mid \epsilon$

To eliminate $Y \rightarrow \epsilon$

$S \rightarrow XYX \mid YX \mid XY \mid Y \mid XX \mid X$

$X \rightarrow 0X \mid 0$

$Y \rightarrow 1Y \mid 1$

Q) $A \rightarrow 0B1 \mid 1B1$

$B \rightarrow 0B \mid 1B \mid \epsilon$

$B \rightarrow \epsilon$

$A \rightarrow 0B1 \mid 1B1) 01 \mid 11$

$B \rightarrow 0B \mid 1B \mid 0 \mid 1$

Removal of Useless Symbols

1. Non terminal must not be derived from Start Symbol
2. If a Non-terminal can't produce a string of terminology

Q) $S \rightarrow XY \mid 0$

$X \rightarrow 1$

Remove $S \rightarrow XY$

$S \rightarrow 0$ ✓

$X \rightarrow 1$ ✓

Q) $S \rightarrow A11B \mid 11A$

$S \rightarrow A11B$ ✗

final grammar

S → 11B|11

A → 0 ✓

B → BB ✗

S → 11B|11 ✗

S → 11A ✓

S → 110 ✓

S → 11 ✓

S → 11A

S → 11

A → 0

Ambiguous Grammar

- if there exists more than 1 parse tree for the given I/P string
- more than 1 left most parse tree
- more than 1 right most parse tree
- or combination of both

Q) E → E+E | E*E | id

Left most id + id * id

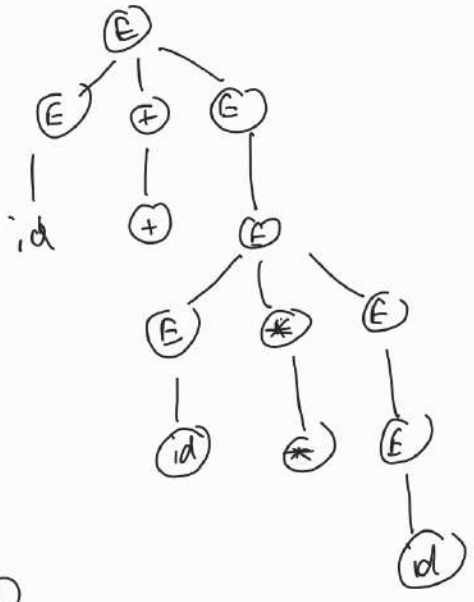
E → E + E

E → id + E

E → id + E * E

E → id + id * E

E → id + id * id



CM PIT

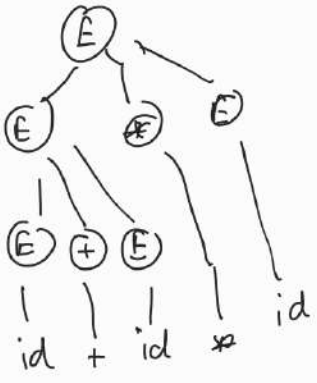
E → E * E

E → E + E * E

E → id + E * E

E → id + id * E

E → id + id * E



Elimination of left Recursion / Removal of Ambiguity

1. left Recursion

A → A α | β

A( )

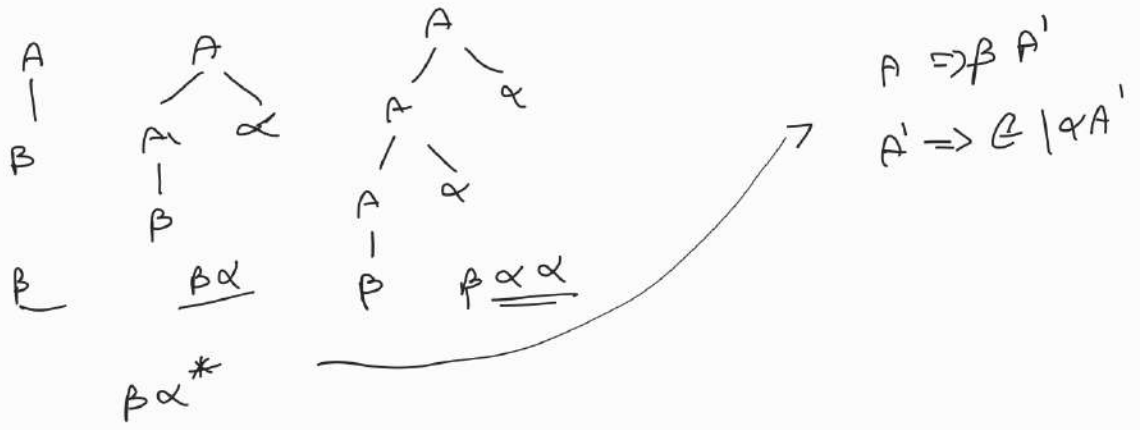
2. Right recursion

A → αA | B

A( )

$A \to \beta A'$

$A' \to \alpha A' \mid \epsilon$



Q) eliminate left recursion for following grammar

$E \to E + T / T$

$T \to T * F \mid F$

$F \to (E) / id$

$A \to A\alpha \mid \beta$

$A \Rightarrow \beta A'$

$A' \Rightarrow \alpha A' \mid \epsilon$

$E \to E + T \mid T$ → left recursion

$E \to TE'$

$E' \to +TE' \mid \epsilon$

$T \to T * F \mid F$

$T \to FT'$

$T' \to *FT' \mid \epsilon$

$F \to (E) \mid id$

↳ no left rec

$E \to TE'$

$E' \to +TE' \mid \epsilon$

$T \to FT'$

$T' \to *FT' \mid \epsilon$

$F \to (E) \mid id$

Q) $S \to S0S1S \mid 01$

$A \to A\alpha \mid \beta$

$A \to \beta A'$

$A' \to \alpha A' \mid \epsilon$

$A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \beta_1 \mid \beta_2 \mid \ldots$

$A \to \beta_1 A' \mid \beta_2 A'$

$A \to \alpha_1 A' \mid \alpha_2 A' \mid \epsilon \ldots$

$S \to 01S'$

$S' \Rightarrow oSiSS'/\epsilon$

Elimination of Left factoring

$A \to \alpha\beta_1 | \alpha\beta_2 | \ldots\ldots r_1 | r_2 \ldots\ldots$

$A \to \alpha A' | r_1 | r_2$

$A' \to \beta_1 | \beta_2 | \ldots$

Q) $S \to iEtS | iEtSeS | a$

✓ $E \to b$

$S \to \underset{A}{iEtS} | \underset{\alpha}{iEtS}\underset{\alpha}{eS} | \underset{\beta_2}{a}\underset{r}{}$

✓ $S \to iEtSS' | a$

✓ $S' \to \epsilon | eS$

Chomsky Normal Form (CNF)

- A CFG is said to be in CNF, if production are of the form $A \to BC$, $A \to a$ where A, B, C are non-terminals & a is a terminal

Q) $S \to bA | aB$

$A \to bAA | aS | a$

$B \to aBB | bS | b$

$A \to a$

$B \to b$

Find the equivalent grammer in CNF

$S \to bA$ $S \to aB$ $A \to bAA$ $A \to aS$ $B \to aBB$ $B \to bS$

$S \to C_bA$ $S \to C_aB$ $A \to C_bAA$ $A \to C_aS$ $B \to C_aBB$ $B \to C_bS$

$C_b \to b$ $C_a \to a$ $C_b \to b$ $C \to a$ $C_a \to a$ $C_b \to b$

$A_1 \to C_bAA$ $B \to C_aBB$

$A_1 \to C_bD_1$ $B \to C_aD_2$

$D_1 \to AA$ $D_2 \to BB$

Greibach Normal Form (GNF)

$A \rightarrow a\alpha$ — a: terminal; α: one or more terminals; A: non terminal

1. Simplify the gammar
2. Check wheather the grammar in CNF
3. Non terminals $A_1, A_2, A_3 \ldots$
4. $A_i \rightarrow A_j \beta$
   - $i < j$ leave the production
   - $i > j$ Substitution
5. $i = j$ left recursion.

Eg:- GNF

$A \rightarrow a$

$A \rightarrow aB$

$A \rightarrow BC$

Q) $S \rightarrow AA/0$

$A \rightarrow SS/1$

replace S with A_1, A with A_2

$A_1 \rightarrow A_2 A_2 / 0$

$A_2 \rightarrow A_1 A_1 / 1$

$A_2 \rightarrow A_1 A_1 / 1$; $i=2$ $j=1$ $(2>1)$

$A_1 \rightarrow A_2 A_2 / 0 \rightarrow (1)$; $i=1$ $j=2$ $(1<2)$

Substituting A_1 in A_2

$A_2 \rightarrow A_2 A_2 A_1 / 0 A_1 / 1$

$i=2$ $j=2$ $(i=j)$ left recursion

$A \rightarrow A\alpha / \beta$

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' / \epsilon$

$A_2 \rightarrow 0A_1 B_2 / 1B_2$

$B_2 \rightarrow A_2 A_1 B_2 / \epsilon$

$A_2 \rightarrow 0A_1 B_2 / 1B_2 / 0A_1 / 1$ → (2)

$B_2 \rightarrow A_2 A_1 B_2 / A_2 A_1$ → (3)

$A_1 \rightarrow A_2 A_2 / 0$

$A_1 \rightarrow 0A_1 B_2 A_2 / 1B_2 A_2 / 0A_1 A_2 / 1A_2 / 0$

$B_2 \rightarrow 0A_1 B_2 A_1 B_2 / 0A_1 B_2 A_1 / 1B_2 A_1 B_2 / 1B_2 A_1 / 0A_1 A_1 B_2 / 0A_1 A_2 B_1 / A_1 B_1 2 / 1A_2 A_1$

Pumping lemma for Context Free Languages

- It is used to prove a language as not content free.

Let 'L' be content free lang let 'n' be a integer constant. Select a string 'z' from 'L' such that $|z| \geq n$

Divide the string z into 5 parts $uvwxy$ such that $|vwx| \leq n$

$|vx| \geq 1$

for $i \geq 0$ $uvwxy$ is in Language

Q] Show that $L = \{a^n b^n c^n \mid n \geq 1\}$ is not a CFL.

Let 'L' is a CFL

Let $n = 3$ $\quad L = \{abc, aabbcc, aaabbbccc, \ldots\}$

$z = \underbrace{aaa}_{u}\underbrace{b}_{v}\underbrace{b}_{w}\underbrace{ccc}_{y}$ $\quad |z| \geq n$, $q \geq 3$

$|vwx| \leq n \Rightarrow |abb| \leq 3$

$3 \leq 3$

$|vx| \geq 1$

$|ab| \geq 1$

$2 \geq 1$

$i \geq 0$, $uv^i wx^i y \in L$

$i = 0 = aaa\,a^0 b b^0 ccc$

$= aabbccc \notin L$