15CS301 THEORY OF COMPUTATION

UNIT-I FINITE AUTOMATA

- Introduction : Basic Mathematical Notation and techniques
- Finite state systems, Basic definitions, Finite automaton : DFA
- Finite Automaton : NDFA, Finite Automaton with ε- moves
- Regular Languages - Regular Expressions
- Equivalence of NFA and DFA
- Equivalence of NDFA'S with and without ε-moves
- Equivalence of finite automaton and regular expressions
- Minimization of DFA
- pumping lemma for regular sets, problems based on pumping lemma.

Introduction:

⇒ Theory of computation is the branch that deals with how efficiently problems can be solved on a model of computation using ~~using~~ an algorithm.

⇒ The field is divided into three major branches

1. Automata theory
2. Computability theory
3. Computational Complexity theory.

Introduction to formal proof

The formal proof can be using deductive proof and inductive proof.

1) Deductive proof ⇒ Sequence of statements given with logical reasoning inorder to prove the first or initial statement.

Initial statement - Hypothesis.

2) Inductive proof ⇒ Recursive kind of proof which consists of sequence of parameterized statements that use the statement itself with lower value of its parameter.

Scanned by CamScanner

Example: Statement B is true because statement A is true

Here Statement A ⇒ Hypothesis

Statement B ⇒ conclusion.

Various forms of proof:

1) proofs about sets ⇒ set is a collection of elements or items.

This proof is of the kind "if and only if" that means an element X is in A if and only if it is in B.

Example: Let us prove $P \cup Q = Q \cup P$

L.H.S ⇒

Statement	Justification
X is in $P \cup Q$	Given
X is in P or X is in Q	1) And by definition of union
X is in Q or X is in P	2) And by definition of union
X is in $Q \cup P$	3) Rule 2) And by definition of union.

RHS ⇒

Statement	Justification
X is in $Q \cup P$	Given
X is in Q or X is in P	1) And by definition of union
X is in P or X is in Q	2) And by definition of union
X is in $P \cup Q$	3) Rule 2) And by definition of union

Hence $P \cup Q = Q \cup P$. Thus A=B is true as element X is in B if and only if X is in A.

2) Proof by Contradiction ⇒ The statement of the form if A and B we start with statement A is not true and thus by assuming false A we try to get the conclusion of statement B.

Example: prove $P \cup Q = Q \cup P$

Proof: Assume $P \cup Q = Q \cup P$ is not true

(ie) $P \cup Q \neq Q \cup P$

Scanned by CamScanner

Consider X is in Q or X is in P => X is in $P \cup Q$ then

X is in $Q \cup P$ according to definition of Union

Hence, The assumption which made initially is false.

Thus, $P \cup Q = Q \cup P$ is proved.

3) <u>proof by counter</u> example

"All possible conditions" in which the statement remains true

Ex: there is no such pair of integers such that

$a \bmod b = b \bmod a$

Proof: consider $a=2$ and $b=3$, then

$2 \bmod 3 \neq 3 \bmod 2$, thus the given pair is true for any pair of integers but if $a=b$ then naturally $\boxed{a \bmod b = b \bmod a,\ a=b}$

<u>Inductive proof</u>:

⇒ Inductive proofs are special proof based on some observations.

⇒ It is used to prove recursively defined objects.

The proof by mathematical induction can be carried out using following steps.

1. <u>Basis</u> ⇒ In this step we assume the lowest possible value. This is an initial step in the proof of mathematical induction.

   (ie) prove the result is true for $n=0$ or $n=1$

2. <u>Induction hypothesis</u> ⇒ In this step we assign value of n to some other value k.

   (ie) Check whether the result is true for $n=k$ or not.

3. <u>Inductive step</u> ⇒ In this step, if $n=k$ is true, then we check whether the result is true for $n=k+1$ or not.

   If we get the same result at $n=k+1$ then we can state that given proof is true by principle of mathematical Induction.

Scanned by CamScanner

Problem1 : prove by induction on 'n' that $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

Solution:

1) Basis of induction

Assume n=1, then $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

$$L.H.S = \sum_{i=0}^{1} 1 = 1$$

$$R.H.S = \frac{n(n+1)}{2} = \frac{1(1+1)}{2} = \frac{2}{2} = 1$$

2) Induction hypothesis

Assume n=k, then $1+2+3+\dots+k = \frac{k(k+1)}{2}$ is true.

3) Inductive step

Assume $n=k+1$,

N.K.T $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

$$L.H.S = \sum_{i=1}^{k+1} i$$

$$= \sum_{i=1}^{k} i + (k+1)$$

$$= \frac{k(k+1)}{2} + (k+1)$$

$$= \frac{k(k+1) + 2(k+1)}{2}$$

$$= \frac{k^2 + k + 2k + 2}{2}$$

$$LHS = \frac{k^2 + 3k + 2}{2} \quad \text{——— (1)}$$

$$RHS = \frac{n(n+1)}{2}$$

$$= \frac{(k+1)((k+1)+1)}{2}$$

$$= \frac{(k+1)(k+2)}{2}$$

$$= \frac{k^2 + 2k + k + 2}{2}$$

$$RHS = \frac{k^2 + 3k + 2}{2} \quad \text{——— (2)}$$

From (1) and (2) LHS = RHS

$$\therefore \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Hence, the given hypothesis is proved.

Scanned by CamScanner

Problem 2: Prove that $1^2+2^2+3^2+\cdots+n^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ using mathematical induction.

Solution:

1) Basis of induction $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Let $n=1$, $LHS = 1^2 = 1$

$$RHS = \frac{n(n+1)(2n+1)}{6} = \frac{1(1+1)(2(1)+1)}{6} = \frac{1\times\cancel{2}\times\cancel{3}}{\cancel{6}} = 1$$

$LHS = RHS$

2) Induction hypothesis

Let $n=k$, $1^2+2^2+3^2+\cdots+k^2 = \sum_{i=1}^{k} i^2 = \frac{k(k+1)(2k+1)}{6}$ is true.

3) Inductive step

Let $n=k+1$, $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

$$LHS = \sum_{i=1}^{k+1} i^2$$

$$= \sum_{i=1}^{k} i^2 + (k+1)^2$$

$$= \frac{k(k+1)(2k+1)}{6} + k^2+2k+1$$

$$= \frac{(k^2+k)(2k+1)+6k^2+12k+6}{6}$$

$$= \frac{2k^3+k^2+2k^2+k+6k^2+12k+6}{6}$$

$$LHS = \frac{2k^3+9k^2+13k+6}{6} \quad \text{—— (1)}$$

$$RHS = \frac{n(n+1)(2n+1)}{6}$$

$$= \frac{(k+1)(k+1+1)(2(k+1)+1)}{6}$$

$$= \frac{(k+1)(k+2)(2k+2+1)}{6}$$

$$= \frac{(k^2+2k+k+2)(2k+3)}{6}$$

$$= \frac{2k^3+3k^2+4k^2+6k+2k^2+3k+4k+6}{6}$$

$$RHS = \frac{2k^3+9k^2+13k+6}{6} \quad \text{—— (2)}$$

From (1) and (2) $LHS = RHS$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Hence, the given statement is proved using mathematical induction.

Scanned by CamScanner

Problem 3: Prove that for every integer $n \geq 0$ the number $4^{2n+1} + 3^{n+2}$ is multiple of 13.

Solution:

1) Basis of induction, Let $n=0$

$$4^{2n+1} + 3^{n+2} = 4^{2(0)+1} + 3^{0+2}$$
$$= 4 + 3^2$$
$$= 4+9$$
$$= 13$$
$$4^{2n+1} + 3^{n+2} = \text{Multiple of } 13$$

2) Induction hypothesis

Let $n=k$, $4^{2k+1} + 3^{k+2}$, for all $k \geq 0$ it is multiple of 13.

3) Inductive step.

Let $n=k+1$, $4^{2n+1} + 3^{n+2} = 4^{2(k+1)+1} + 3^{k+1+2}$

$$= 4^{2k+3} + 3^{k+3}$$
$$= 4^{2k+1} \cdot 4^2 + 3^{k+2} \cdot 3^1$$
$$= 16\left(4^{2k+1}\right) + 3\left(3^{k+2}\right)$$
$$= 13\left(4^{2k+1}\right) + 3\left(4^{2k+1}\right) + 3\left(3^{k+2}\right)$$
$$= 13\left(4^{2k+1}\right) + 3\left(4^{2k+1} + 3^{k+2}\right)$$
$$= 13\left(4^{2k+1}\right) + 3(13k)$$
$$= 13\left(4^{2k+1} + 3k\right)$$
$$4^{2n+1} + 3^{n+2} = \text{Multiple of } 13$$

$\therefore$ For every integer $n \geq 0$, the number $4^{2n+1} + 3^{n+2}$ is multiple of 13.

Problem 4: Prove that for every integer $n \geq 0$ the number $n^4 - 4n^2$ is Divisible by 3.

Problem 5: Prove that every tree has 'e' edges and 'e+1' nodes.

Scanned by CamScanner

Basic concept of Automata theory

- Automata theory has a basic fundamental unit called set.
- Set is used to represent the mathematical model.

Set => It is defined as collection of objects enclosed within { }.

Ex: A set of element which are less than 5,

$A = \{0, 1, 2, 3, 4\}$

Set of vowels $A = \{a, e, i, o, u\}$

Recursion => To define the elements of the set $A = \{x \mid x \text{ is a square of } n\}$ where $n \leq 10$

(ie) $A = \{0, 1, 4, 9, 16, \ldots 100\}$

Subset => Subset A is called subset of B if every element of set A is present in set B, denoted by $A \subseteq B$

Ex: $A = \{1, 2, 3\}$ and $B = \{1, 2, 3, 4, 5\}$ then $A \subseteq B$.

Empty set => Set having no element, $A = \{\ \}$ written as ϕ (phi).

Null String => Null element is denoted by ε

Power set => Set of all the subsets of its elements

Ex: $A = \{1, 2, 3\}$ power set $Q = \{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{3,2\}, \{2,3\}, \{1,2,3\}\}$

Operation on set =>

(i) **Union** => combination of both sets.

$A = \{1, 2, 3\}$, $B = \{1, 2, 4\}$ then $A \cup B = \{1, 2, 3, 4\}$

(ii) **Intersection** => Common elements from both the sets

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

(iii) **Difference** => Elements there in one set but not there in another

$A = \{1, 2, 3\}$, $B = \{1, 2, 4\}$ then $A - B = \{3\}$.

(iv) **Complement** => $\overline{A} = U - A$. $U = \{10, 20, 30, 40, 50\}$, $A = \{10, 20\}$

$\overline{A} = U - A = \{30, 40, 50\}$

Cartesian product => Cartesian product of two sets A and B is a set of all possible ordered pairs.

Ex: $A = \{a, b\}$ and $B = \{0, 1, 2\}$

$A \times B = \{\{a, b\} \mid a \in A, b \in B\}$ $A \times B = \{(a,0), (a,1), (a,2), (b,0), (b,1), (b,2)\}$

Scanned by CamScanner

Relations => The relation R is a collection for the set S which represents the pair of elements.

Properties: If $A = \{a, b\}$ then,

Reflexive relation R can be $= \{(a,a), (b,b)\}$

Irreflexive relation R can be $= \{(a,b)\}$

Transitive relation R can be $= \{(a,b), (b,a), (a,a)\}$

Symmetric relation R can be $= \{(a,b), (b,a)\}$

Asymmetric relation R can be $= \{(a,b)\}$

Equivalence relation => A relation is said to be equivalence relation if it is, reflexive, symmetric and transitive.

Kleen closure => set of strings of any length including null string ξ.

Ex: Let $\Sigma = \{a, b\}$ obtain Σ^*

$$\Sigma^* = \{\xi, a, b, aa, bb, ab, ba, aaa, bbb, aba, \ldots\ldots\}$$

Positive closure => consists of all the strings of any length except a null string.

Ex: Let $\Sigma = \{a, b\}$ $\Sigma^+ = \{a, b, aa, bb, ab, ba, aaa, bbb, aba, \ldots\}$

Alphabets, Strings and Languages

Alphabet: An alphabet is a finite collection of symbols.

Ex: $S = \{a, b, c, \ldots z\}$, $W = \{0, 1\}$ Here 0, 1 are alphabets, denoted by W.

Strings: Finite collection of symbols from alphabet.

Ex: $\Sigma = \{a, b\} = \{ab, bb, aa, aaa, bbb, \ldots\}$

Language: It is a collection of appropriate strings. The language is defined using an input set.

Ex: $\Sigma = \{\xi, 0, 00, 000, \ldots\ldots\}$

Here, the language is defined as 'any number of zeros'.

Operations => Union $L_1 \cup L_2$, Intersection $L_1 \cap L_2$

Concatenation $L_1 L_2$, Difference $L_1 - L_2$.

Scanned by CamScanner

Example: Describe the following language over the input set $A = \{a, b\}$

(i) $L_1 = \{a, ab, ab^2\}$ ⇒ All the strings with letter a followed by any number of b's.

(ii) $L_2 = \{a^n b^n \mid n \geq 1\}$ ⇒ Strings having equal number of a's followed by equal number of b's.

(iii) $L_3 = \{a^m b^n \mid n > 0\}$ ⇒ All the strings with any numbers of a's followed by atleast one b.

Application of Automata theory

1) It is the base for the formal languages and useful for programming language.
2) It plays an important role in compiler Design
3) To prove the correctness of the program automata theory is used.
4) Switching theory and design and analysis of digital circuits automata Theory is applied.
5) Automata theory deals with the design finite state machine.

Finite State Systems

⇒ Finite state system represents a mathematical model of a system with certain input. The model finally gives certain output.

⇒ The input when is given to the machine it is processed by various states, These states are called as intermediate states.

Example: Control mechanism of elevator. This mechanism only remembers the current floor number pressed, it does not remember all the previously pressed numbers.

Finite Automata - Definition

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

Q - finite set of states

Σ - input alphabet

q_0 - initial state and q_0 is in Q (ie) $q_0 \in Q$

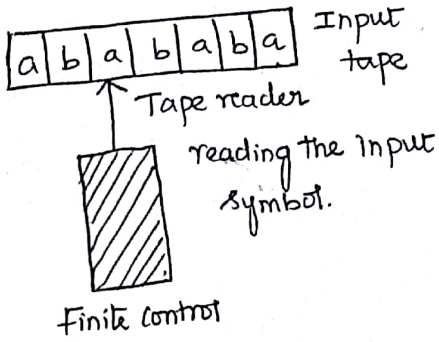
F - Final states

δ - Transition function - Used to determine next state.

Scanned by CamScanner

Finite Automata Model.

The finite automata can be represented using



(i) Input tape => Linear tape having number of cells. Each i/p symbol is placed in each cell.

(ii) Finite control => It decides the next state on receiving particular i/p from input tape.

The tape reader reads the cells one by one from left to right and at a time only one i/p symbol is read.

Acceptance of strings and Languages

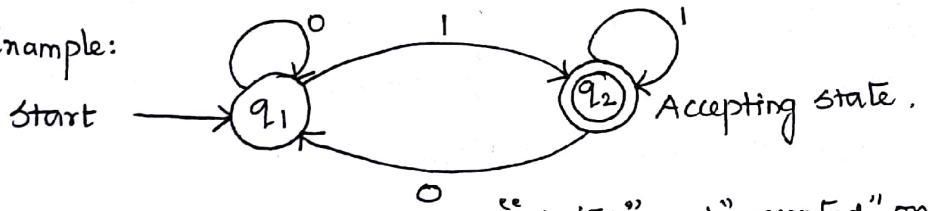
The strings and languages can be accepted by a finite automata, when it reaches to a final state.

Notations => (i) Transition diagram (ii) Transition table.

Transition diagram => Directed graph associated with the vertices of the graph correspond to the states of the finite automata.

Start state => $\rightarrow q_1$ Final state => q_2

Example:



If a string ends in a 0, it is "rejected" and "accepted" only if the string ends in 1 ∴ The language $L(M) = \{ w \mid w \text{ ends in } 1 \}$.

Transition table => Tabular representation of finite automata.

Input / States	0	1
$\rightarrow q_1$	q_1	q_2
$*q_2$	q_1	q_2

Language Acceptance by FA:

A string 'x' is accepted by FA, $M = (Q, \Sigma, \delta, q_0, F)$ only if $\delta(q_0, x) = P$ for some P in F.

$$L(M) = \{ x \mid \delta(q_0, x) \text{ is in } F \}$$

Scanned by CamScanner

DFA (Deterministic Finite Automata)

The finite Automata is called Deterministic finite Automata if there is only one path for a specific input from current state to next state.

A Deterministic finite automata is a Quintuple.

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

Q - Finite set of states

Σ - Finite set of i/p symbols

δ - Transition function

q_0 - Start state, $q_0 \in Q$.

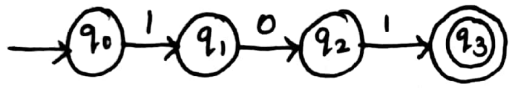
F - Final state, $F \subseteq Q$.

Problems:

1) Design an FA which accepts the only input 101 over the input set $\Sigma = \{0,1\}$.

Solution:

Here $M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_3\})$



δ: $\delta(q_0,1) = q_1$ $\quad$ $\delta(q_2,1) = q_3$.

$\delta(q_1,0) = q_2$

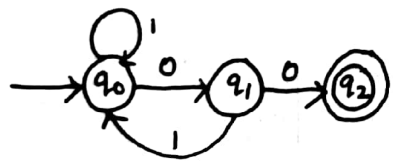
2) Design DFA for a) set of all strings ending in 00

b) Set of all strings with three consecutive Zero's.

Solution:

a)

Here $M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$

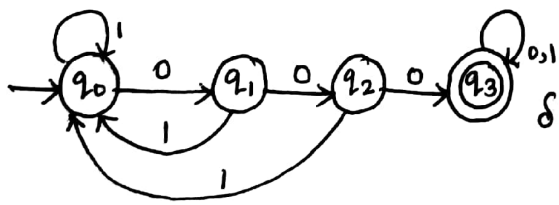


δ: $\delta(q_0,0) = q_1$ | $\delta(q_1,0) = q_2$

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

b)

Here $M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_3\})$



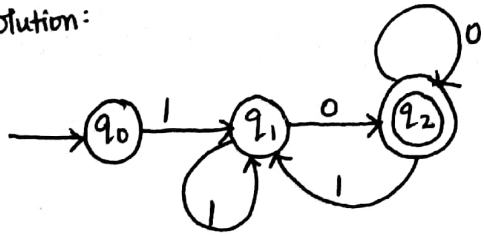
δ: $\delta(q_0,0) = q_1$ | $\delta(q_1,0) = q_2$ | $\delta(q_2,0) = q_3$ | $\delta(q_3,0) = q_3$

$\delta(q_0,1) = q_0$ | $\delta(q_1,1) = q_0$ | $\delta(q_2,1) = q_0$ | $\delta(q_3,1) = q_3$.

3) Design FA which accepts only those strings which start with 1 and ends with 0.

Solution:

Here $M = (\{q_0, q_1, q_2\}, \{0,1\}, q_0, \delta, \{q_2\})$



δ: $\delta(q_0,1) = q_1$ | $\delta(q_2,0) = q_2$

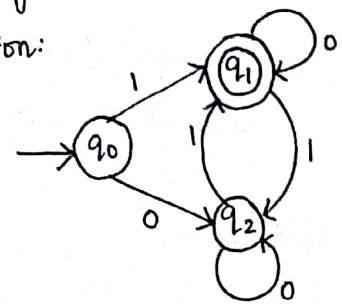
$\delta(q_1,0) = q_2$ | $\delta(q_2,1) = q_1$

$\delta(q_1,1) = q_1$

Scanned by CamScanner

4) Design FA which accepts odd number of 1's and any number of 0's.

Solution:



Here $M = (\{q_0, q_1, q_2\}, \{0,1\}, q_0, \delta, \{q_1\})$

δ: $\delta(q_0,0) = q_1$ | $\delta(q_1,0) = q_1$ | $\delta(q_2,0) = q_2$

$\delta(q_0,1) = q_2$ | $\delta(q_1,1) = q_2$ | $\delta(q_2,1) = q_1$

Testing: 10101 $\Rightarrow$ 1 0 1 0 1

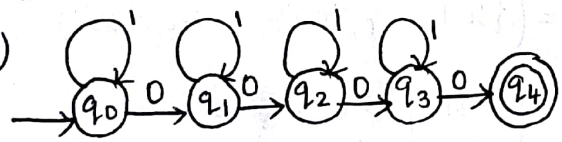
$$\begin{aligned}
\delta(q_0, 10101) &= \delta(\delta(q_0,1), 0101) \\
&= \delta(q_1, 0101) \\
&= \delta(\delta(q_1,0), 101) \\
&= \delta(q_1, 101) \\
&= \delta(\delta(q_1,1), 01) \\
&= \delta(q_2, 01) \\
&= \delta(\delta(q_2,0), 1) \\
&= \delta(q_2, 1) \\
&= q_1 \text{ Accepting state.}
\end{aligned}$$

5) Design DFA with a) All string that contain exactly 4 zeros.

b) All string that doesn't contain the substring 110.

Solution:

a)



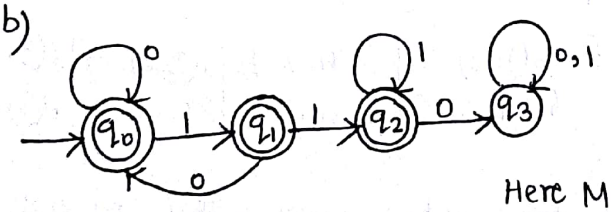
Here $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0,1\}, q_0, \delta, \{q_4\})$

δ: $\delta(q_0,0) = q_1$ | $\delta(q_1,0) = q_2$ | $\delta(q_2,0) = q_3$

$\delta(q_0,1) = q_0$ | $\delta(q_1,1) = q_1$ | $\delta(q_2,1) = q_2$

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

b)



Here $M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, q_0, \delta, \{q_0, q_1, q_2\})$

δ: $\delta(q_0,0) = q_0$ | $\delta(q_1,0) = q_0$ | $\delta(q_2,0) = q_3$ | $\delta(q_3,0) = q_3$

$\delta(q_0,1) = q_1$ | $\delta(q_1,1) = q_2$ | $\delta(q_2,1) = q_2$ | $\delta(q_3,1) = q_3$.

Scanned by CamScanner

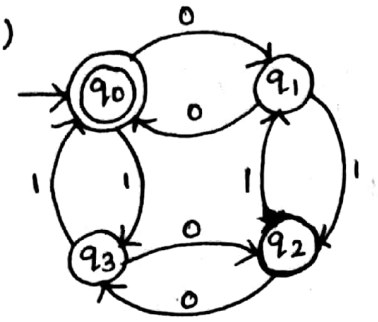
b) Design FA which accepts i) even number of 0's and even number of 1's. ⑦

ii) odd number of 0's and odd number of 1's.

iii) even number of 0's and odd number of 1's.

iv) odd number of 0's and even number of 1's.

(i)



$\delta(q_0, 0011) = \delta(\delta(q_0, 0), 011)$

$= \delta(q_1, 011)$

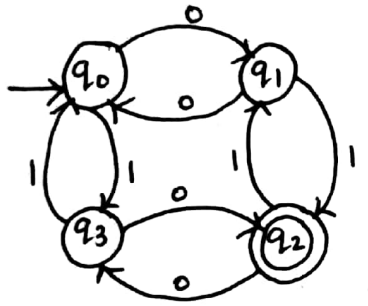
$= \delta(\delta(q_1, 0), 11)$

$= \delta(q_0, 11)$

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

$= \delta(q_3, 1) = q_0$ Accepted.

(ii)



$\delta(q_0, 101010) = \delta(\delta(q_0, 1), 01010)$

$= \delta(q_3, 01010)$

$= \delta(\delta(q_3, 0), 1010)$

$= \delta(q_2, 1010)$

$= \delta(\delta(q_2, 1), 010)$

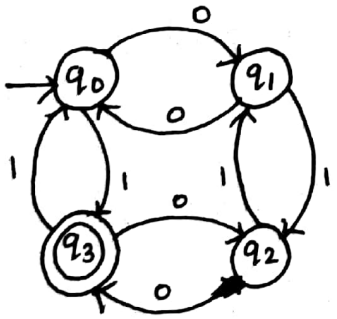
$= \delta(q_1, 010)$

$= \delta(\delta(q_1, 0), 10)$

$= \delta(q_0, 10) = \delta(\delta(q_0, 1), 0) = \delta(q_3, 0)$

$= q_2$ Accepted.

(iii)



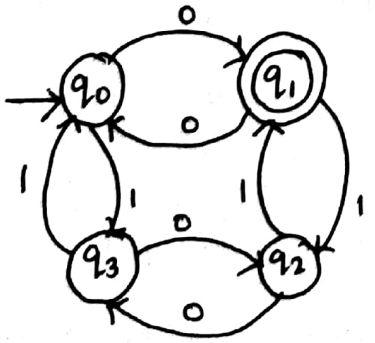
$\delta(q_0, 010) = \delta(\delta(q_0, 0), 10)$

$= \delta(q_1, 10)$

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

$= \delta(q_2, 0) = q_3$ Accepted.

(iv)



$\delta(q_0, 101) = \delta(\delta(q_0, 1), 01)$

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

$= \delta(q_2, 1)$

$= q_1$ Accepted.

Scanned by CamScanner

7) Design FA for the following

a) Number of 1's in W is 3 mod 4

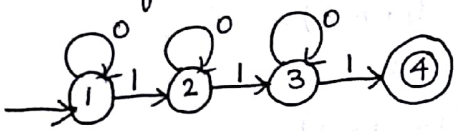
b) $L = \{0^n \mid n \bmod 3 = 2\}$

Solution:

a) Consider W = 010010001

No of 1's = 3 (ie) 3 mod 4 = 3

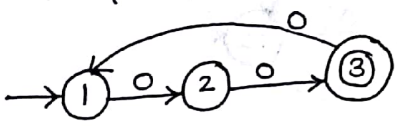
The no. of 1's should be exactly 3.



b) $L = \{0^n \mid n \bmod 3 = 2\}$

Here 1 mod 3 = 1, 2 mod 3 = 2, 3 mod 3 = 0 — Accepted, 5 mod 3 = 2

$L = \{000, 0000, 00000000, \ldots\}$



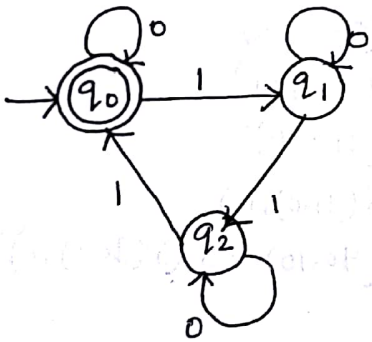
8) Construct DFA for the following

a) Number of 1's is a multiple of 3.

b) Number of 1's is not a multiple of 3.

Solution

a)



$\delta(q_0, 1110) = \delta(\delta(q_0, 1), 110)$

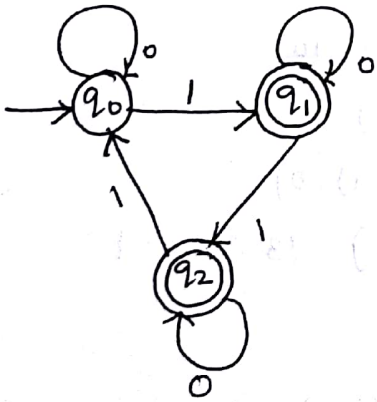
$= \delta(\delta(q_1, 1), 10)$

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

$= \delta(q_0, 0)$

$= q_0$ Accepted $\Rightarrow$ No of 1's is multiple of 3

b)



$\delta(q_0, 11110) = \delta(\delta(q_0, 1), 1110)$

$= \delta(\delta(q_1, 1), 110)$

$= \delta(\delta(q_2, 1), 10)$

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

$= \delta(q_1, 0)$

$= q_1$ Accepted.

Here no. of 1's is not a multiple of 3.

Scanned by CamScanner

NFA (Non-Deterministic Finite Automata)

The finite Automata is called NFA when there exists many paths for a specific input from current state to next state.

A non-deterministic finite Automata is defined by,

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

Where

Q - Finite set of internal states

Σ - Finite set of I/p symbols

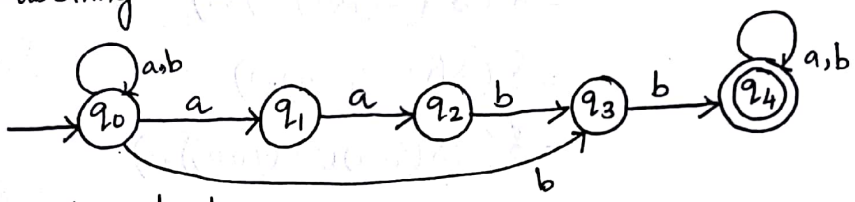
q_0 - start state, $q_0 \in Q$

δ - Transition function

F - Finite set of final states $F \subseteq Q$.

Example: Draw state transition diagram for FA over $\{a,b\}$ containing substring aabb.

Solution:

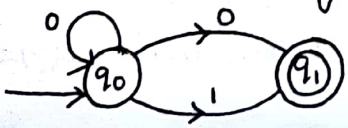


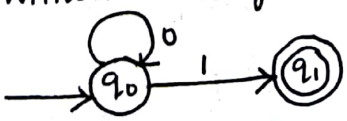
Transition table:

	I/p / States	0	1
Start State →	q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
	q_1	-	q_2
	q_2	q_2	q_2
	q_3	q_4	-
Final State	q_4	q_4	q_4

DFA Vs NFA.

DFA	NFA
1. Every i/p string leads to the unique state of finite automata.	For some i/p there can be more than one next states.
2. Requires more memory for storing the state information.	Requires more computation time.
3. It is not possible to move to next state without reading any symbol.	Possible to move to next state without reading any symbol.





Scanned by CamScanner

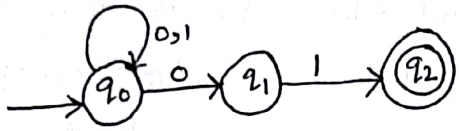
Extended Transition Function : $\hat{\delta}$

This is to represent transition functions with a string of input symbols 'w' and returns a set of states.

Suppose $w = xa$,

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

Example:



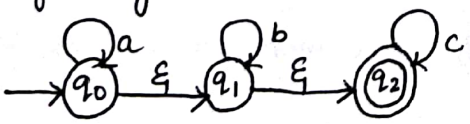
Process the input 001

$$\begin{aligned}
\hat{\delta}(q_0, 001) &= \delta(\hat{\delta}(q_0, 00), 1) \\
&= \delta(\delta(\hat{\delta}(q_0, 0), 0), 1) \\
&= \delta(\delta(\{q_0, q_1\}, 0), 1) \\
&= \delta((\delta(q_0, 0) \cup \delta(q_1, 0)), 1) \\
&= \delta(\{q_0, q_1\} \cup \phi), 1) \\
&= \delta(\{q_0, q_1\}, 1) \\
&= \delta(q_0, 1) \cup \delta(q_1, 1) \\
&= \{q_0\} \cup \{q_2\} \\
&= \{q_0, q_2\}
\end{aligned}$$

Finite Automaton with ε-moves

In the non-deterministic finite state machine, the transition that does not require input symbols for the state transition and is capable of transiting to zero or more states with ε is called epsilon transition.

Example: Find ε-closure for the following NFA.



ε-closure$(q_0) = \{q_0, q_1, q_2\}$

ε-closure$(q_1) = \{q_1, q_2\}$

ε-closure$(q_2) = \{q_2\}$.

ε-closure

The ε-closure(p) is a set of all states which are reachable from state p on ε-moves such that

(i) ε-closure$(p) = p$ where $p \in Q$

(ii) If there exists ε-closure$(p) = \{q\}$ and $\delta(q, \varepsilon) = r$ then ε-closure$(p) = \{q, r\}$.

Scanned by CamScanner

Equivalence of NFA and DFA

Theorem: Let L be a set accepted by NFA, then there exists a DFA that accepts L.

Proof: This can be proved by subset construction method

Let $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$

$D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$

The states of D are all the subset of the set states of N.

$$Q_D = 2^{Q_N}$$

An element of Q_D will be denoted by $[q_1, q_2, \dots q_i]$

where $q_1, q_2, \dots q_i \in Q$

Define $\delta_D([q_1, q_2, \dots q_i], a) = [P_1, P_2, \dots P_j]$

if and only if

$\delta_N([q_1, q_2, \dots q_i], a) = \{P_1, P_2, \dots P_j\}$

δ_D is computed by applying δ_N to each states of δ_D represented by $[q_1, q_2, \dots q_j]$

On applying to each of $q_1, q_2, \dots q_i$ and taking the union.

We get, $[P_1, P_2, \dots P_j]$ in Q_D

To show $\delta_D(\{q_0\}, x) = [q_1, q_2, \dots q_j]$ if and only if

$\delta_N(q_0, x) = \{q_1, q_2, \dots q_i\}$

This is proved by the method of induction.

F_D is the set of subset 'S' of Q_N such that $S_n \cap F_n \neq \phi$.

Problem: Construct DFA equivalent to the NFA $M = (\{a, b, c, d\}, \{0, 1\}, \delta, a, \{b, d\})$ where δ is defined in the following table.

δ	0	1
a	$\{b, d\}$	b
b	c	$\{b, c\}$
c	d	a
d	ϕ	a

Scanned by CamScanner

Solution:

(i) Find the states of $Q_N = \{a, b, c, d\}$

(ii) Construct subsets $Q_D = 2^{Q_N} = 2^4 = 16$

$$Q_D = \{ [a], [b], [c], [d], [a,b], [a,c], [a,d], [b,c], [b,d], [c,d], [a,b,c], [a,c,d], [b,c,d], [a,b,c,d], \phi \}$$

$[a,b,d]$

(iii) Find δ_D for all states

$\delta_D([a],0) = [b,d]$
$\delta_D([a],1) = [b]$
$\delta_D([b],0) = [c]$
$\delta_D([b],1) = [b,c]$
$\delta_D([c],0) = [d]$
$\delta_D([c],1) = [a]$
$\delta_D([d],0) = \phi$
$\delta_D([d],1) = [a]$

$\delta_D([a,b],0) = [b,c,d]$
$\delta_D([a,b],1) = [b,c]$

$\delta_D([a,c],0) = [b,d]$
$\delta_D([a,c],1) = [a,b]$
$\delta_D([a,d],0) = [b,d]$
$\delta_D([a,d],1) = [a,b]$
$\delta_D([b,c],0) = [c,d]$
$\delta_D([b,c],1) = [a,b,c]$
$\delta_D([b,d],0) = [c]$
$\delta_D([b,d],1) = [a,b,c]$
$\delta_D([c,d],0) = [d]$
$\delta_D([c,d],1) = [a]$

$\delta_D([a,b,c],0) = [b,c,d]$
$\delta_D([a,b,c],1) = [a,b,c]$
$\delta_D([a,c,d],0) = [b,d]$
$\delta_D([a,c,d],1) = [a,b]$
$\delta_D([b,c,d],0) = [c,d]$
$\delta_D([b,c,d],1) = [a,b,c]$
$\delta_D([a,b,c,d],0) = [b,c,d]$
$\delta_D([a,b,c,d],1) = [a,b,c]$

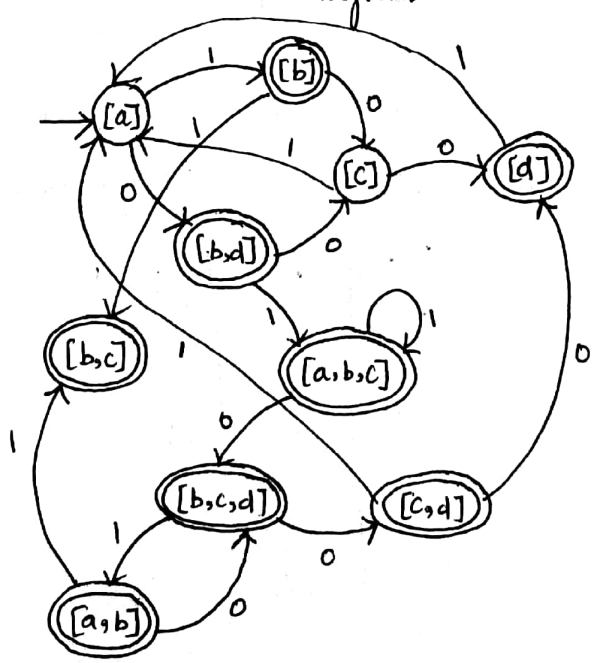
(iv) $F_N = \{b, d\}$ (Final state) $F_D \Rightarrow$ all set containing $\{b,d\}$

$$F_D = \{ [b], [d], [a,d], [b,d], [c,d], [a,b], [b,c], [a,b,c], [b,c,d], [a,b,d], [a,b,c,d] \}$$

(v) Transition table:

	0	1
[a]	[b,d]	[b]
* [b]	[c]	[b,c]
[c]	[d]	[a]
* [d]	ϕ	[a]
* [a,b]	[b,c,d]	[b,c]
[a,c]	[b,d]	[a,b]
* [a,d]	[b,d]	[a,b]
* [b,c]	[c,d]	[a,b,c]
* [b,d]	[c]	[a,b,c]
* [c,d]	[d]	[a]
* [a,b,c]	[b,c,d]	[a,b,c]
* [a,c,d]	[b,d]	[a,b]
* [b,c,d]	[c,d]	[a,b,c]
* [a,b,c,d]	[b,c,d]	[a,b,c]
ϕ	ϕ	ϕ

(vi) Transition diagram



Scanned by CamScanner

Equivalence of NDFA's with and without ε-moves

Theorem: If L is accepted by NFA with ε-transitions, then there exist L which is accepted by NFA without ε-transitions.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with ε-transition

Construct M' which is NFA without ε-transition.

$$M' = (Q, \Sigma, \delta', q_0, F') \text{ where,}$$

$$F' = \begin{cases} F \cup \{q_0\}, & \text{if } \varepsilon\text{-closure}(q_0) \text{ contains a state of } F \\ F, & \text{otherwise.} \end{cases}$$

By induction: $\begin{cases} \delta' \text{ and } \hat{\delta} \text{ are same} \\ \delta \text{ and } \hat{\delta} \text{ are different} \end{cases}$

Let x be any string $\delta'(q_0, x) = \hat{\delta}(q_0, x)$

This statement is not true if $x = \varepsilon$ because,

$$\delta'(q_0, \varepsilon) = \{q_0\} \text{ and } \hat{\delta}(q_0, \varepsilon) = \varepsilon\text{-closure}(q_0).$$

Basis: $|x| = 1$, x is a symbol whose value is a $\delta'(q_0, a) = \hat{\delta}(q_0, a)$ [By definition of $\hat{\delta}$]

Inductive: Let $x = wa$ where a is in Σ

$$\delta'(q_0, wa) = \delta'(\hat{\delta}(q_0, w), a)$$
$$= \delta'(P, a)$$

Now we must know that

$$\delta'(P, a) = \hat{\delta}(q_0, wa)$$

But

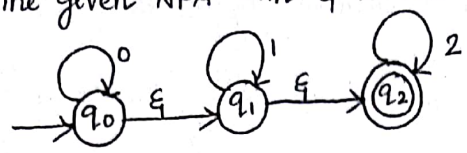
$$\delta'(P, a) = \bigcup_{q \text{ in } P} \delta'(q, a) = \bigcup_{q \text{ in } P} \hat{\delta}(q, a)$$
$$= \hat{\delta}(\hat{\delta}(q_0, w), a)$$
$$= \hat{\delta}(q_0, wa)$$
$$= \hat{\delta}(q_0, x)$$

Hence $\delta'(q_0, x) = \hat{\delta}(q_0, x)$

Hence proved.

Scanned by CamScanner

Problem: Convert the given NFA with ϵ to NFA without ϵ.



Solution:

(i) Find ϵ-closure of all the states

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

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

$\epsilon\text{-closure}(q_2) = \{q_2\}$.

(ii) Find δ' transitions for each state on each input symbol.

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

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

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

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

$= \epsilon\text{-closure}(q_0)$

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

$\delta'(q_0,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0,\epsilon),1)$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0),1)$

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

$= \epsilon\text{-closure}(\phi \cup q_1 \cup \phi)$

$= \epsilon\text{-closure}(q_1)$

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

$\delta'(q_1,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1,\epsilon),0))$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1),0)$

$= \epsilon\text{-closure}(\delta(q_1,q_2),0)$

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

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

$\delta'(q_1,0) = \phi$

$\delta'(q_1,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1,\epsilon),1)$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1),1)$

$= \epsilon\text{-closure}(\delta(q_1,q_2),1)$

$= \epsilon\text{-closure}(q_1 \cup \phi)$

$= \epsilon\text{-closure}(q_1)$

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

$\delta'(q_2,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2,\epsilon),0)$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2),0)$

$= \epsilon\text{-closure}(\delta(q_2,0))$

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

$\delta'(q_2,0) = \phi$

$\delta'(q_2,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_2,\epsilon),1)$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2),1)$

$= \epsilon\text{-closure}(\delta(q_2,1))$

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

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

$\delta'(q_0,2) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0,\epsilon),2)$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0),2))$

$= \epsilon\text{-closure}(\delta(q_0,q_1,q_2),2)$

$= \epsilon\text{-closure}(\phi \cup \phi \cup q_2)$

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

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

$\delta'(q_1,2) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1,\epsilon),2)$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1),2)$

$= \epsilon\text{-closure}(\delta(q_1,q_2),2)$

$= \epsilon\text{-closure}(\phi \cup q_2)$

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

Scanned by CamScanner

$\delta'(q_2,2) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_2,\varepsilon),2))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_2),2)$

$= \varepsilon\text{-closure}(\delta(q_2,2))$

$= \varepsilon\text{-closure}(q_2)$

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

(iii) Summarize all the computed δ' transitions.

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

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

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

$\delta'(q_1,0) = \phi$

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

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

$\delta'(q_2,0) = \phi$

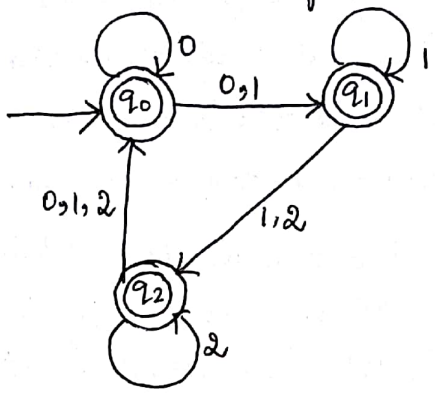
$\delta'(q_2,1) = \phi$

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

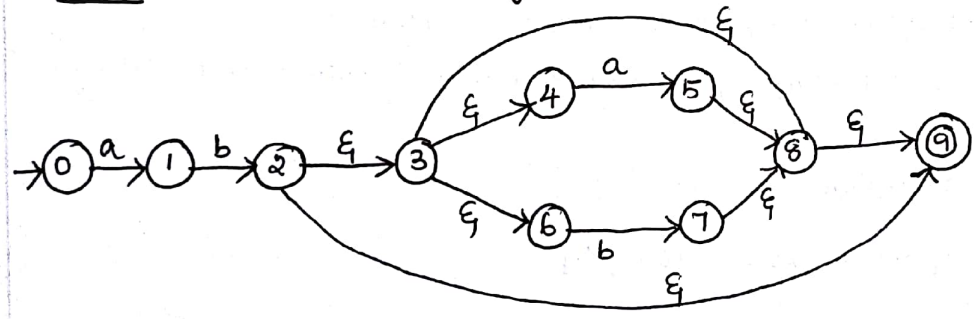
(iv) Transition table

Input / State	0	1	2
q_0	$\{q_0,q_1,q_2\}$	$\{q_1,q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1,q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

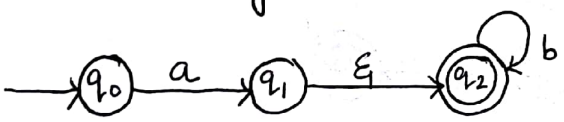
(v) Transition diagram.



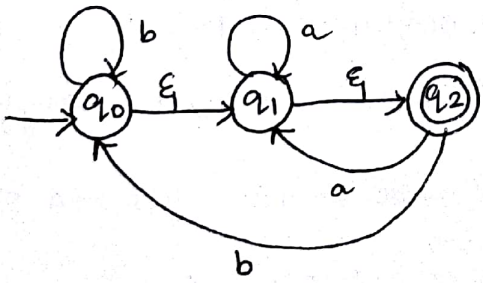
Problem2: Convert the following NFA with ε to NFA without ε.



Problem3: Convert the following NFA with ε to NFA without ε.



Problem4: Construct NFA with ε to ordinary NFA.



Scanned by CamScanner

Regular Language - Regular Expression

⇒ Regular languages are those languages which are accepted by finite automaton.

Ex: Set of strings of 0's and 1's $L = \{(0+1)^* \mid (0,1) \in \Sigma\}$

⇒ The languages accepted by finite automata are easily described by simple expression called regular expressions.

Properties: (closure)

1. ϕ is a regular expression which denotes the empty set.
2. ε is a RE and denotes the set $\{\varepsilon\}$ and it is a null string.
3. For each 'a' in Σ 'a' is a RE and denotes the set $\{a\}$
4. If r and s are regular expression denoting the language L_1 and L_2 respectively then

   $r+s$ is equivalent to $L_1 \cup L_2$ (union)

   rs is equivalent to $L_1 L_2$ (concatenation)

   r^* is equivalent to L_1^* (closure)

Problem 1: Describe the following by regular expression.

a) L_1 = The set of all strings of 0's and 1's ending in 00

b) L_2 = The set of all strings of 0's and 1's beginning with 0 and ending with 1.

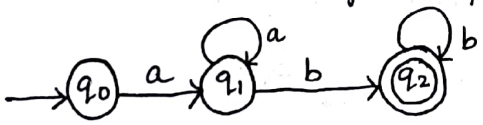
Solution

a) Regular Expression $L_1 = (0+1)^*00$

b) Regular expression $L_2 = 0(0+1)^*1$

Problem 2: Construct a DFA for the regular expression aa^*bb^*

Solution:



Problem 3: Find out the language generated by the regular expression $(0+1)^*$

Solution: $L = \{\varepsilon, 0, 1, 00, 11, 01, 10, 000, 111, \ldots\ldots\}$

$L = \{$Any number of 0's and 1's including null string$\}$

Problem 4: Give regular expression to describe an identifier and positive integer.

Solution: identifier = $(a\text{-}z\ A\text{-}Z)\ (a\text{-}z\ A\text{-}Z\ 0\text{-}9)^*$

Positive integer = $(0\text{-}9)^+$

Scanned by CamScanner

Equivalence of finite Automaton and regular expressions

Every language defined by the regular expressions can be also be defined by the finite state automata.

Theorem: If $L = L(A)$ for some DFA, then there is a regular expression R such that $L = L(R)$.

Proof: Let A be a DFA which defines the language L.

Let the states be $\{1, 2, \ldots n\}$

⇒ Construct RE of the form $R_{ij}^{(k)}$ – set of string w that takes the FA from state 'i' to state 'j' without going through any states labeled greater than k.

⇒ Let us prove this theorem by induction for $R_{ij}^{(k)}$.

Basis: If $k=0$, there will be no intermediate node between i and j

(i) $i \longrightarrow j$ An arc from node 'i' to 'j'

(ii) i An arc from the node 'i' itself.

The transition may be (i) $R_{ij}^{(0)} = \phi$ (ii) $R_{ij}^{(0)} = a$ (iii) $R_{ij}^{(0)} = a_1 + a_2 + \cdots + a_\ell$

if $i = j$ then $R_{ij}^{(0)} = \varepsilon + a$

if $i \neq j$ then $R_{ij}^{(0)} = a_1 + a_2 + \cdots + a_\ell$

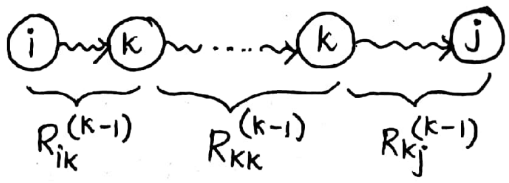
If there is no arc form i to j then, $R_{ij}^{(0)} = \phi$.

Induction: Let there is a path from state i to state j that goes through 'i' which is not greater than k.

Here it is assumed that we have already got $R_{ij}^{(k-1)}$

(i) Suppose if path does not goes through the state k, then $R_{ij}^{(k)} = R_{ij}^{(k-1)}$

(ii) If the path goes through the states k atleast one.



The set of lables for all path is represented by the RE

$$R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$

By combining the above two path going through k and not going through k, we have

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$

∴ Regular expression $R_{ij}^{(k)}$ can be constructed for all values of i, j and k.

Scanned by CamScanner

ARDEN'S THEOREM: (FA → RE)

Let P and Q be two regular expressions over Σ. If P does not contain ε, then the equation in $R = Q + RP$ has a solution $R = QP^*$

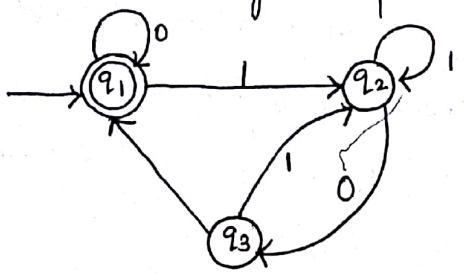
Using this theorem, it is easy to find the regular expression

The condition to apply this theorem are,

(i) Finite automata does not have ε-moves

(ii) It has only 1 start state.

Example: Construct the regular expression for the given finite automata.



Solution: The transitions are defined by the equations,

$$q_1 = q_1 0 + q_3 0 + \varepsilon \quad \text{——(1)}$$

$$q_2 = q_1 1 + q_2 1 + q_3 1 \quad \text{——(2)}$$

$$q_3 = q_2 0 \quad \text{——(3)}$$

Substitute (3) in (2)

$$q_2 = q_1 1 + q_2 1 + q_2 01$$

$$q_2 = q_1 1 + q_2 (1+01)$$

$$q_2 = q_1 1 (1+01)^* \quad \text{——(4) [By theorem]}$$

Substitute (4) in (1)

$$q_1 = q_1 0 + q_2 0.0 + \varepsilon$$

$$= q_1 0 + q_1 1 (1+01)^* 0.0 + \varepsilon$$

$$= q_1 0 + q_1 (1 (1+01)^* 00) + \varepsilon$$

$$= q_1 (0 + 1 (1+01)^* 00) + \varepsilon$$

$$= \varepsilon (0 + 1 (1+01)^* 00)^* \quad \text{[By theorem]}$$

$$q_1 = (0 + 1 (1+01)^* 00)^*$$

Since q_1 is the final state, the required regular expression is

$$= (0 + 1 (1+01)^* 00)^*$$

Scanned by CamScanner

THAMSON'S CONSTRUCTION (RE → FA)

Every language defined by a regular expression is also defined by a finite automaton.

Proof: To prove $L = L(E)$ for some ε-NFA.

Basis: (Zero operator)

(i) Exactly one accepting state → q_0 $r = \varepsilon$

(ii) No ars into the initial state → q_0 q_4 $r = \phi$

(iii) No arcs out of the accepting state → $q_0 \xrightarrow{a} q_1$ $r = a$.

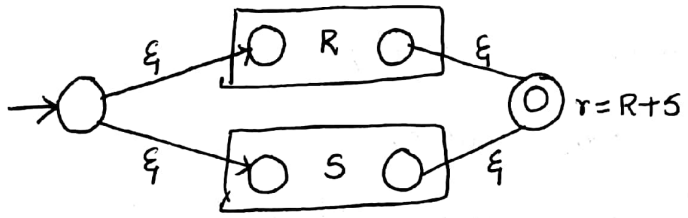
Induction: This theorem can be true for n number of operators.

In any type of regular expression there are only three cases possible.

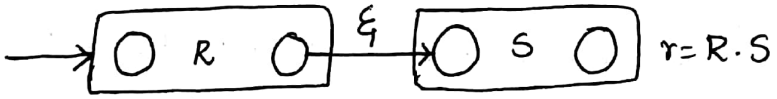
1. Union 2. Concatenation 3. Closure

Case(i) Union. $r = R + S$

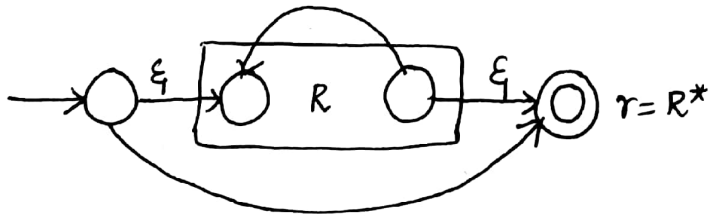
Let $r = R+S$, where r_1 and r_2 be the regular expressions.



Case(ii) $r = R \cdot S$ Concanetation

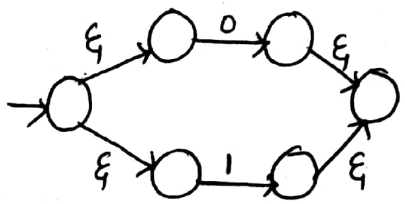


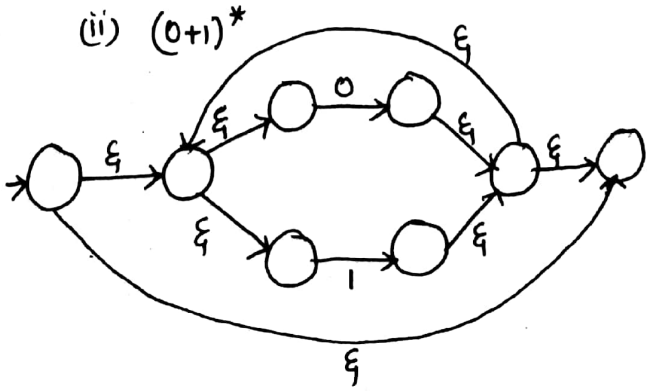
Case(iii) Closure



Problem 1: Convert the regular expression $(0+1)^* 1 (0+1)$ to an ε-NFA.

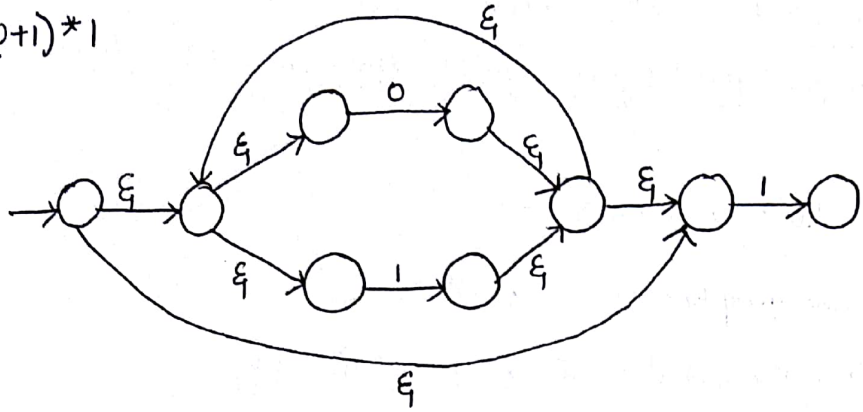
Solution: (i) Find the RE for $0+1$ (ii) $(0+1)^*$



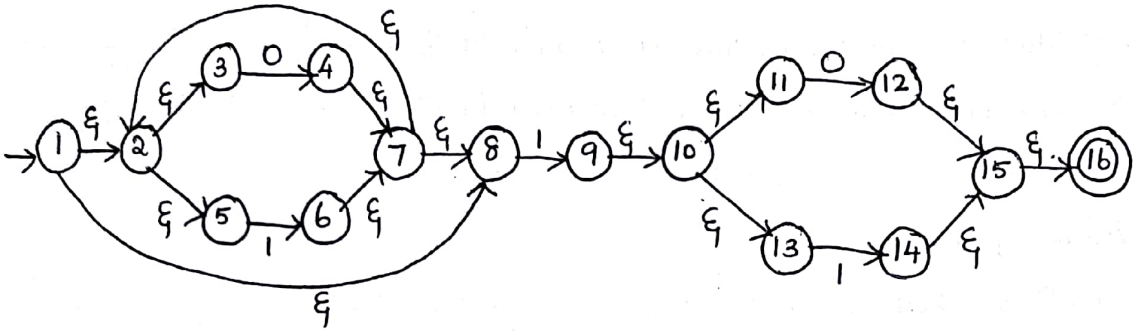


Scanned by CamScanner

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



(iv) $(0+1)^*\,1\,(0+1)$



Problem 2: Construct a NFA equivalent $(0+1)^*\,(00+11)$

Problem 3: Construct an NFA accepting L given by

$$L = \{ x \mid (a,b)^* \mid,\ |x| \geq 3 \text{ and the third symbol of } x \text{ from right is 'b'} \}$$

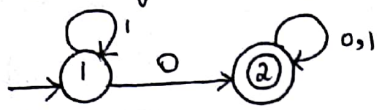
$$[RE = (a+b)^*\, b\, (a+b)\,(a+b)]$$

Problem 4: Construct an NFA to accept the language indicated by the following regular expression $((01+001)^*0^*)^*$.

Scanned by CamScanner

Problems: FA → RE

1) Convert the following to a regular expression.



Solution: Find $R_{ij}^{(k)}$ for all the values of i, j, k.

Let $k=0$, $R_{ij}^{(0)} = \begin{cases} \varepsilon + a_1 + a_2 + \dots + a_l & \text{if } i=j \\ a_1 + a_2 + \dots + a_r & \text{if } i \neq j \end{cases}$

$R_{11}^{(0)} = \varepsilon + 1 \quad R_{12}^{(0)} = 0 \quad R_{21}^{(0)} = 0 \quad R_{22}^{(0)} = \varepsilon + 0 + 1$

If $k=1$, Then $R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$ —— (1)

$R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)} \left(R_{11}^{(0)}\right)^* R_{1j}^{(0)}$ —— (2)

Find $R_{11}^{(1)}, R_{12}^{(1)}, R_{21}^{(1)}, R_{22}^{(1)}$

(2) $\Rightarrow R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$

$= (\varepsilon+1) + (\varepsilon+1)(\varepsilon+1)^*(\varepsilon+1)$

$= (\varepsilon+1) + (\varepsilon+1)\, 1^* (\varepsilon+1)$

$= (\varepsilon+1) + 1^*$

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

(2) $\Rightarrow R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{11}^{(0)}$

$= \phi + \phi (\varepsilon+1)^* (\varepsilon+1)$

$= \phi + (\varepsilon+1)^* (\varepsilon+1)$

$\boxed{R_{21}^{(1)} = \phi}$

(2) $\Rightarrow R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$

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

$= 0 + (\varepsilon+1)\, 1^* 0$

$= 0 + 1^* 0$

$\boxed{R_{12}^{(1)} = 1^* 0}$

(2) $\Rightarrow R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)} \left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$

$= (\varepsilon+0+1) + \phi (\varepsilon+1)^* 0$

$= (\varepsilon+0+1) + \phi$

$\boxed{R_{22}^{(1)} = \varepsilon+0+1}$

Let $k=2$

$R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)} \left(R_{22}^{(1)}\right)^* R_{2j}^{(1)}$ —— (3)

Find $R_{11}^{(2)}, R_{12}^{(2)}, R_{21}^{(2)}, R_{22}^{(2)}$

Scanned by CamScanner

③ $\Rightarrow R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)} \left(R_{22}^{(1)}\right)^* R_{21}^{(1)}$

$= 1^* + 1^* 0 \, (\xi+0+1)^* \phi$

$= 1^* + \phi$

$\boxed{R_{11}^{(2)} = 1^*}$

③ $\Rightarrow R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)} \left(R_{22}^{(1)}\right)^* R_{21}^{(1)}$

$= \phi + (\xi+0+1)\,(\xi+0+1)^* \phi$

$= \phi + \phi$

$\boxed{R_{21}^{(2)} = \phi}$

③ $\Rightarrow R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} \left(R_{22}^{(1)}\right)^* R_{22}^{(1)}$

$= 1^* 0 + 1^* 0 \, (\xi+0+1)^* (\xi+0+1)$

$= 1^* 0 + 1^* 0 \, (0+1)^* (\xi+0+1)$

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

$\boxed{R_{12}^{(2)} = 1^* 0 \, (0+1)^*}$

③ $\Rightarrow R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)} \left(R_{22}^{(1)}\right)^* R_{22}^{(1)}$

$= (\xi+0+1) + (\xi+0+1)\,(\xi+0+1)^* (\xi+0+1)$

$= (\xi+0+1) + (\xi+0+1)\,(0+1)^*$

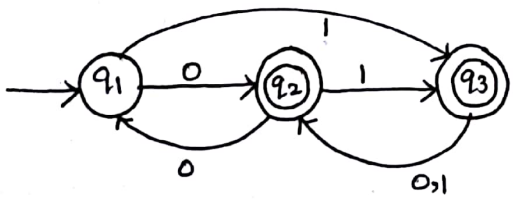
$= (\xi+0+1) + (0+1)^*$

$\boxed{R_{22}^{(2)} = (0+1)^*}$

Regular expression for the accepting state

$$\boxed{R_{12}^{(2)} = 1^* 0 \, (0+1)^*}$$

<u>Problem 2</u>: Convert the following to a regular expression.



Solution: To find $R_{12}^{(3)} + R_{13}^{(3)}$

$$R_{ij}^{(0)} = \begin{cases} \xi + a_1 + a_2 + \cdots + a_r & \text{if } i=j \\ a_1 + a_2 + \cdots + a_r & \text{if } i \neq j \end{cases}$$

<u>K=0</u>

$R_{11}^{(0)} = \xi$ $\quad R_{21}^{(0)} = 0$ $\quad R_{31}^{(0)} = \phi$

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

$R_{13}^{(0)} = 1$ $\quad R_{23}^{(0)} = 1$ $\quad R_{33}^{(0)} = \xi$.

<u>K=1</u>

$$R_{ij}^{(K)} = R_{ij}^{(K-1)} + R_{iK}^{(K-1)} \left(R_{KK}^{(K-1)}\right)^* R_{Kj}^{(K-1)}$$

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

Scanned by CamScanner

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

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

$= \varepsilon + \varepsilon$

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

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

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

$= 0 + 0$

$\boxed{R_{12}^{(1)} = 0}$

$R_{13}^{(1)} = R_{13}^{(0)} + R_{11}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)}$

$= 1 + \varepsilon(\varepsilon)^* 1$

$\boxed{R_{13}^{(1)} = 1}$

$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{11}^{(0)}$

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

$= 0 + 0 \cdot \varepsilon$

$= 0 + 0$

$\boxed{R_{21}^{(1)} = 0}$

$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)}$

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

$\boxed{R_{22}^{(1)} = \varepsilon + 00}$

$R_{23}^{(1)} = R_{23}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)}$

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

$\boxed{R_{23}^{(1)} = 1 + 01}$

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

$= \phi + \phi(\varepsilon)^* \varepsilon$

$\boxed{R_{31}^{(1)} = \phi}$

$R_{32}^{(1)} = R_{32}^{(0)} + R_{31}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)}$

$= 0 + 1 + \phi(\varepsilon)^* 0$

$= 0 + 1 + \phi$

$\boxed{R_{32}^{(1)} = 0 + 1}$

$R_{33}^{(1)} = R_{33}^{(0)} + R_{31}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)}$

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

$\boxed{R_{33}^{(1)} = \varepsilon}$

$\underline{\underline{K=2}}$ $\quad R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)}(R_{22}^{(1)})^* R_{2j}^{(1)}$

$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^* R_{21}^{(1)}$

$= \varepsilon + 0(\varepsilon + 00)^* 0$

$= \varepsilon + 0(00)^* 0$

$= \varepsilon + (00)^*$

$\boxed{R_{11}^{(2)} = (00)^*}$

$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^* R_{22}^{(1)}$

$= 0 + 0(\varepsilon + 00)^*(\varepsilon + 00)$

$= 0 + 0(00)^*(\varepsilon + 00)$

$= 0 + 0(00)^*$

$\boxed{R_{12}^{(2)} = 0(00)^*}$

$R_{13}^{(2)} = R_{13}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^* R_{23}^{(1)}$

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

$= 1 + 0(00)^*(1 + 01)$

$= 1 + (00)^*(\varepsilon + 0)1 \quad [(00)^*(\varepsilon + 0) = 0^*]$

$= 1 + 0^* 1$

$\boxed{R_{13}^{(2)} = 0^* 1}$

Scanned by CamScanner

$$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)} \left(R_{22}^{(1)}\right)^* R_{21}^{(1)}$$
$$= 0 + (\varepsilon + 00)\ (\varepsilon + 00)^* 0$$
$$= 0 + (\varepsilon + 00)\ (00)^* 0$$
$$= 0 + (00)^* 0$$
$$\boxed{R_{21}^{(2)} = (00)^* 0}$$

$$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)} \left(R_{22}^{(1)}\right)^* R_{22}^{(1)}$$
$$= (\varepsilon + 00) + (\varepsilon + 00)\ (\varepsilon + 00)^* (\varepsilon + 00)$$
$$= (\varepsilon + 00) + (\varepsilon + 00)\ (00)^* (\varepsilon + 00)$$
$$= (\varepsilon + 00) + (00)^*$$
$$\boxed{R_{22}^{(2)} = (00)^*}$$

$$R_{23}^{(2)} = R_{23}^{(1)} + R_{22}^{(1)} \left(R_{22}^{(1)}\right)^* R_{23}^{(1)}$$
$$= (1+01) + (\varepsilon + 00)\ (\varepsilon + 00)^* (1+01)$$
$$= (1+01) + (\varepsilon + 00)\ (00)^* (1+01)$$
$$= (1+01) + (00)^* (1+01)$$
$$= (1+01) + (00)^* (\varepsilon + 0)1$$
$$= (1+01) + 0^* 1$$
$$= 1 + 0^* 1$$
$$\boxed{R_{23}^{(2)} = 0^* 1}$$

$$R_{31}^{(2)} = R_{31}^{(1)} + R_{32}^{(1)} \left(R_{22}^{(1)}\right)^* R_{21}^{(1)}$$
$$= \phi + (0+1)\ (\varepsilon + 00)^* 0$$
$$\boxed{R_{31}^{(2)} = (0+1)(00)^* 0}$$

$$R_{32}^{(2)} = R_{32}^{(1)} + R_{32}^{(1)} \left(R_{22}^{(1)}\right)^* R_{22}^{(1)}$$
$$= (0+1) + (0+1)\ (\varepsilon + 00)^* (\varepsilon + 00)$$
$$= (0+1) + (0+1)\ (00)^* (\varepsilon + 00)$$
$$= (0+1) + (0+1)\ (00)^*$$
$$\boxed{R_{32}^{(2)} = (0+1)\ (00)^*}$$

$$R_{33}^{(2)} = R_{33}^{(1)} + R_{32}^{(1)} \left(R_{22}^{(1)}\right)^* R_{23}^{(1)}$$
$$= \varepsilon + (0+1)\ (\varepsilon + 00)^* (1+01)$$
$$= \varepsilon + (0+1)\ (00)^* (1+01)$$
$$= \varepsilon + (0+1)\ (00)^* (\varepsilon + 0)1$$
$$\boxed{R_{33}^{(2)} = \varepsilon + (0+1)\ 0^* 1}$$

Calculate $r_{12}^{(3)} + r_{13}^{(3)}$, to find regular expression

$$r_{12}^{(3)} = r_{12}^{(2)} + r_{13}^{(2)} \left(r_{33}^{(2)}\right)^* r_{32}^{(2)}$$
$$= 0\,(00)^* + 0^*1\ \left(\varepsilon + (0+1)\,0^*1\right)^* (0+1)\ (00)^*$$
$$= 0\ (00)^* + 0^*1\ \left((0+1)\ 0^*1\right)^*\ (0+1)\ (00)^*$$

$$r_{13}^{(3)} = r_{13}^{(2)} + r_{13}^{(2)} \left(r_{33}^{(2)}\right)^* r_{33}^{(2)}$$
$$= 0^*1 + 0^*1\ (\varepsilon + (0+1)\ 0^*1)^*\ (\varepsilon + (0+1)0^*1)$$
$$= 0^*1\ ((0+1)0^*1)^*$$

$$\therefore\ r_{12}^{(3)} + r_{13}^{(3)} = 0^*1\ \left((0+1)0^*1 + (\varepsilon + (0+1)\ 00)^*\right) + 0\,(00)^*$$

Scanned by CamScanner

Minimization of DFA.

The minimization of FSM means reducing the number of states from given FA. While minimizing FSM, first find out which two states are equivalent, we can represent those two states by one representative state.

Pbm: Give the minimized DFA for the following expression $(a/b)^*abb$

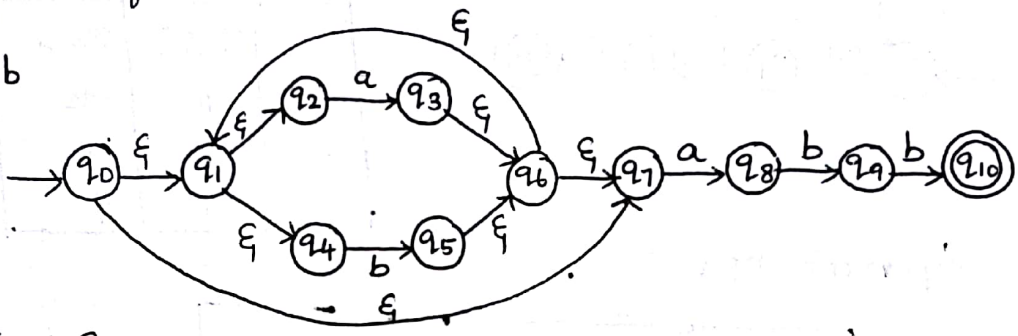
Solution: Steps:

(1) Design of NFA with ϵ for given regular expression

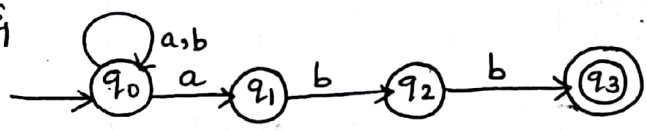
(2) Conversion of NFA with ϵ to without ϵ

(3) Conversion of NFA to DFA.

$(a/b)^*abb$



NFA without ϵ



Transition table:

	a	b
q_0	$\{q_0,q_1\}$	q_0
q_1	ϕ	q_2
q_2	ϕ	q_3
q_3	ϕ	ϕ

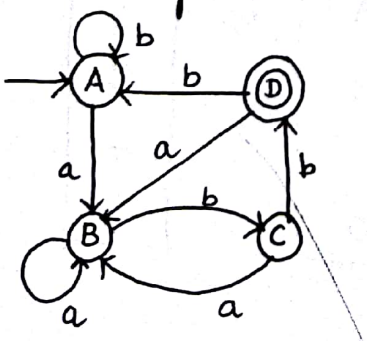
Equivalent DFA

	a	b
→ q_0	$\{q_0,q_1\}$	q_0
q_1	ϕ	q_2
q_2	ϕ	q_3
* q_3	ϕ	ϕ
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
$\{q_0,q_2\}$	$\{q_0,q_1\}$	$\{q_0,q_3\}$
* $\{q_0,q_3\}$	$\{q_0,q_1\}$	q_0

Assume $q_0 = A$, $\{q_0,q_1\} = B$

$\{q_0,q_2\} = C$, $\{q_0,q_3\} = D$

Transition diagram,



Transition table,

States \ I/P	a	b
A	B	A
B	B	C
C	A	D
D	B	A

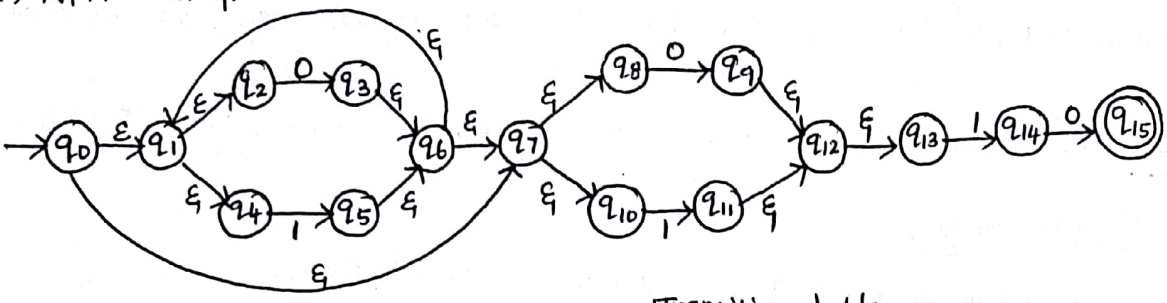
Scanned by CamScanner

Problem 2: Construct the minimized DFA for the regular expression

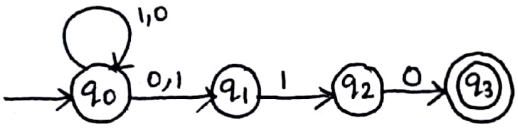
$(0+1)^* (0+1) 10$

Solution:

(i) NFA with ε.



(ii) NFA without ε



Transition table

S \ I/P	0	1
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$
q_1	ϕ	q_2
q_2	q_3	ϕ
q_3	ϕ	ϕ

Equivalent DFA

State \ I/P	0	1
q_0	$[q_0, q_1]$	$[q_0, q_1]$
q_1	ϕ	$[q_2]$
q_2	$[q_3]$	ϕ
q_3	ϕ	ϕ
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_3]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_3]$	$[q_0, q_1]$	$[q_0, q_1, q_2]$

Assume

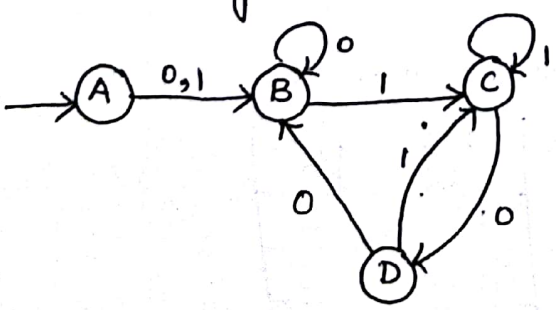
$q_0 = A$

$[q_0, q_1] = B$

$[q_0, q_1, q_2] = C$

$[q_0, q_1, q_3] = D$

Transition Diagram (Min-DFA)

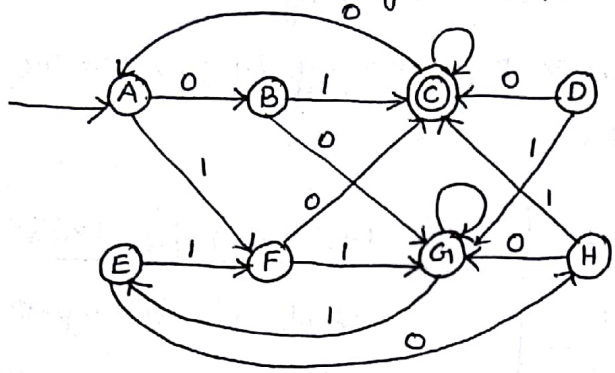


Transition table:

	0	1
A	B	B
B	B	C
C	D	C
D	B	C

Scanned by CamScanner

Problem: 3 Minimize the DFA as given below



Solution:

Transition table

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

Find the equivalent state

Consider pair (B,H)

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

$\delta(H,0) = G$ | $\delta(H,1) = C$

∴ (B,H) are equivalent.

Since (B,H) are equivalent, consider the pair (A,E)

$\delta(A,0) = B$ (or) H | $\delta(A,1) = F$

$\delta(E,0) = H$ (or) B | $\delta(E,1) = F$

∴ (A,E) are equivalent

Consider the pair (D,F)

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

$\delta(F,0) = C$ | $\delta(F,1) = G$

∴ (D,F) are equivalent.

∴ The equivalent pairs are (A,E), (B,H) and (D,F)

∴ The minimized DFA will be

	0	1
→ A	B	D
B	G	C
C	A	C
D	C	G
G	G	A

Table filling

B	X						
C	X	X					
D	X	X	X				
E	(A,E)	X	X	X			
F	X	X	X	(D,F)	X		
G	X	X	X	X	X	X	
H	X	(B,H)	X	X	X	X	X
	A	B	C	D	E	F	G

Scanned by CamScanner

Problem 4: Define distinguishable and indistinguishable states. Using table filling method, minimize the following DFA. Draw the transition diagram for resulting DFA.

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

Distinguishable states

↳ If for some input string w, $\hat{\delta}(p,w)$ gives an accepting state and $\hat{\delta}(q,w)$ gives a non-accepting state then states p and q are called distinguishable or non-equivalent state.

Indistinguishable states

↳ If for some input string w, $\hat{\delta}(p,w)$ and $\hat{\delta}(q,w)$ both produces either accepting or non accepting states.

Solution:

Here Final states are C, F, I

Other states are A, B, D, E, G, H

Since, C, F, I are final state

Consider the pair (E, H):

$\delta(E,0) = F$ | $\delta(E,1) = I$

$\delta(H,0) = I$ | $\delta(H,1) = C$

∴ (E, H) is equivalent state

Consider the pair (B, H)

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

$\delta(H,0) = I$ | $\delta(H,1) = C$

∴ (B, H) is equivalent state.

Consider the pair (B, E)

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

$\delta(E,0) = F$ | $\delta(E,1) = I$

∴ (B, E) is equivalent state.

Since (B, H) and (B, E) is equivalent,

Consider the pair (A, G)

$\delta(A,0) = B$ or H | $\delta(A,1) = E$ or B

$\delta(G,0) = H$ or B | $\delta(G,1) = B$ or E

∴ (A, G) is equivalent state.

Scanned by CamScanner

Since (B,E) and (E,H) are equivalent

Consider the pair (A,D)

$\delta(A,0) = B$ or E | $\delta(A,1) = E$ or H

$\delta(D,0) = E$ or B | $\delta(D,1) = H$ or E

$\therefore$ (A,D) is equivalent state.

Since (E,H) and (B,H) are equivalent

Consider the pair (D,G)

$\delta(D,0) = E$ or H | $\delta(D,1) = H$ or B

$\delta(G,0) = H$ or E | $\delta(G,1) = B$ or H

$\therefore$ (D,G) are equivalent.

Since (D,G) and (B,H) are equivalent

Consider the pair (C,F)

$\delta(C,0) = D$ or G | $\delta(C,1) = H$ or B

$\delta(F,0) = G$ or D | $\delta(F,1) = B$ or H

$\therefore$ (C,F) are equivalent.

Since (A,G) and (B,E) are equivalent

Consider the pair (F,I)

$\delta(F,0) = G$ or A | $\delta(F,1) = B$ or E

$\delta(I,0) = A$ or G | $\delta(I,1) = E$ or B

$\therefore$ (F,I) is equivalent state.

Since (A,D) and (E,H) are equivalent

Consider the pair (C,I)

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

$\delta(I,0) = A$ or D | $\delta(I,1) = E$ or H

$\therefore$ (C,I) is equivalent.

Table filling:

	A	B	C	D	E	F	G	H
B	x							
C	x	x						
D	(A,D)	x	x					
E	x	(B,E)	x	x				
F	x	x	(C,F)	x	x			
G	x	x	x	(D,G)	x	x		
H	x	(B,H)	x	x	(E,H)	x	x	
I	x	x	(C,I)	x	x	(F,I)	x	x

Here

State A = G = D

State B = H = E

state C = I = F

Minimized DFA is

I/p / State	0	1
A	B	B
B	C	C
C	A	B

Scanned by CamScanner

Pumping lemma for Regular sets

⇒ Used for checking whether given string is accepted by regular expression or not.

⇒ whether given language is regular or not.

Theorem: Let L be a regular set. Then there is a constant n such that if z is any word in L. and $|z| \geq n$ we can write $z = uvw$ such that $|uv| \leq n$, $|v| \geq 1$ for all $i \geq 0$, uv^iw is in L. The n should not be greater than the number of states.

Proof: If the language L is regular it is accepted by a DFA.

$M = (Q, \Sigma, \delta, q_0, F)$ with some particular number of states say n.

Consider the input can be $a_1, a_2, \ldots a_m$, $m \geq n$. The mapping function δ could be written as $\delta(q_0, q_1, \ldots q_i) = q_i$.

If q_m is in F (ie) $q_1, q_2, \ldots q_m$ is in $L(M)$ then $a_1 a_2 \ldots a_j a_{k+1} a_{k+2} \ldots a_m$ is also in $L(M)$.

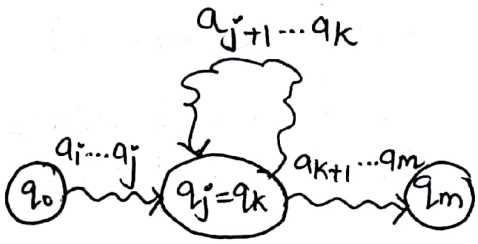


Fig: Pumping lemma

Since there is path from q_0 to q_m that goes through q_j but not around the loop labelled $a_{j+1} \ldots a_k$. Thus.

$$\delta(q_0, a_1, a_j a_{k+1} \ldots a_m) = \delta(\delta(q_0, q_1, \ldots q_j), a_{k+1}, \ldots a_m)$$
$$= \delta(q_j, q_{k+1} \ldots q_m)$$
$$= \delta(q_k, q_{k+1} \ldots q_m)$$
$$= q_m$$

Thus $a_1 \ldots a_j (a_{j+1} \ldots a_k)^i a_{k+1} \ldots a_m$ is in $L(M)$ for any $i \geq 0$

Scanned by CamScanner

Problem 1: Show that $L = \{0^{n^2} \mid n \geq 1\}$ is not regular.

Solution

(i) Assume L is regular.

(ii) No. of states $= n^2 > n$

(iii) Let $z = 0^{n^2}$, consider $n^2 = r$, $z = 0^r$

(iv) According to the pumping lemma,

z can be written as,

$z = uvw$ such that $|uv| \leq n$, $|v| \geq 1$

$\therefore$ $z = 0^r$ can be expressed as,

Assume, $uv = 0^m$, where $m < r$

$v = 0^q$, where $q < m$

and $w = 0^{r-m}$

(v) check the condition.

(a) $|uv| \leq n$, $|0^m| \leq n$, $m \leq n$

(b) $|v| \geq 1$, $|0^q| \geq 1$, $q \geq 1$

(c) for all $i \geq 0$, the string uv^iw is in L

$$uv^iw = uv^{i-1}(v)w$$
$$= (uv)\,(v)^{i-1}w$$
$$= 0^m\,(0^q)^{i-1}\,0^{r-m}$$
$$= 0^{m+q(i-1)+r-m}$$
$$\boxed{uv^iw = 0^{q(i-1)+r}}$$

For $i=0$, $uv^0w = uw = 0^{r-q} \neq 0^r \notin L$

$i=1$, $uvw = 0^r \in L$

$i=2$ $uv^2z = 0^{q+r} \notin L$

Since for $i = 0, 2$, we have the string that does not belongs to the language L.

$\therefore$ $L = \{0^{n^2} \mid n \geq 1\}$ is not regular.

Scanned by CamScanner

Problem 2: Show that $L=\{a^n b^n \mid n \geq 1\}$ is not regular.

Solution:

i) Assume L is regular.

ii) No. of states $L=\{a^n b^n \mid n \geq 1\}$ (ie) $n+n=2n>n$

iii) Let $z=a^n b^n$

According to the pumping lemma, z can be written as,

$z=uvw$ such that $|uv| \leq n$, $|v| \geq 1$

iv) Consider $z=a^n b^n = a^m a^{n-m} b^n$

$z=a^m a^{n-m} b^n$ can be expressed as,

$$\boxed{uv=a^m, \quad v=a^j \quad w=a^{n-m} b^n}$$

v) check the conditions,

(a) $|uv| \leq n \Rightarrow |a^m| \leq n$, $m \leq n$

(b) $|v| \geq 1 \Rightarrow |a^j| \geq 1$, $j \geq 1$

(c) for all $i \geq 0$ the string uv^iw is in L.

$$uv^iw = u\, v^{i-1} (v)\, w$$
$$= (uv)\,(v)^{i-1} w$$
$$= a^m\, (a^j)^{i-1}\, a^{n-m} b^n$$

$$\boxed{uv^iw = a^{n+j(i-1)} b^n}$$

For $i=0$, $uw = a^{n-j} b^i \notin L$

$i=1$, $uvw = a^n b^n \in L$

$i=2$, $uv^2w = a^{n+j} b^n \notin L$

Since for $i=0, 2$, we have the string that does not belongs to the language L.

$\therefore L=\{a^n b^n \mid n \geq 1\}$ is not regular.

Scanned by CamScanner

Problem 3: Show that $L = \{a^m b^n a^{m+n} \mid m \geq 1, n \geq 1\}$ is not regular.

Solution:

i) Assume L is regular.

ii) Let $z = a^m b^n a^{m+n}$

iii) According to pumping lemma

$$z = uvw, \quad |uv| \leq n, \quad |v| \geq 1$$

$z = a^m b^n a^{m+n}$ is expressed as,

$$\boxed{uv = a^m, \quad v = a^q, \quad w = a^{P-m} b^n a^O} \quad [O = m+n]$$

iv) Check the condition.

(a) $|uv| \leq n, \quad |a^m| \leq n, \; m \leq n$

(b) $|y| \geq 1, \quad |a^q| \geq 1, \; q \geq 1$

(c) for all $i \geq 0$ the string uv^iw is in L

$$uv^iw = uv^{i-1}vw$$
$$= (uv)(v)^{i-1}w$$
$$= a^m (a^q)^{i-1} a^{P-m} b^n a^O$$

$$\boxed{uv^iw = a^{q(i-1)+P} b^n a^O}$$

For $i=0$, $uw = a^{P-q} b^n a^O \notin L$

$i=1$, $uvw = a^P b^n a^O \in L$

$i=2$, $uv^2w = a^{P+q} b^n a^O$

Take some random value for P, q, n and O

$P=4, \; n=3, \; O=10, \; q=1$

$a^{4+1} b^3 a^{10} = a^5 b^3 a^{10}$ [Here $m=5, n=3$, $m+n=8 \neq 10$]

$\notin L$.

$\therefore$ Since for $i=0$ and $i=2$, string does not belongs to L.

$\therefore L = \{a^m b^n a^{m+n} \mid m \geq 1, n \geq 1\}$ is not regular.

Scanned by CamScanner

Problem 4: Is the following language regular? Justify your answer.

$$L = \{0^{2n} \mid n \geq 1\}$$

Solution:

This is a language length of string is always even.

(ie) $n=1$, $L=00$

$n=2$, $L=0000$ and so on.

Let $L = uvw$

$L = 0^{2n}$

$|z| = 2n = uv^iw$

If we add $2n$ to this string length

$|z| = 4n = uv.vw$ = even length of string.

Thus even after pumping $2n$ to the string we get the even length, so the language $L=\{0^{2n} \mid n \geq 1\}$ is regular language.

Problem 5: prove $L=\{a^p \mid p \text{ is a prime}\}$ is not regular.

Solution: Let, assume L is regular and p is a prime number.

$L = a^p$

$|z| = uvw$, $i=1$

Consider $L = uv^iw$ where $i=2$

$= uv.vw$

Adding 1 to p, we get

$p < |uvvw|$

$p < p+1$

But $p+1$ is not prime number.

Assumption is contradictory. Thus $L=\{a^p \mid p \text{ is a prime}\}$ behaves as it is not a regular language.

Scanned by CamScanner

UNIT-II GRAMMAR.

Grammar Introduction:

A grammar of a language (G) is defined as,

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

V - finite set of objects called Variables (Non-terminal)

T - finite set of objects called Terminals

$S \in V$ - Start symbol

P - finite set of productions.

Types of Grammar:

- Type 0 grammars / unrestricted grammars (Recursively Enumerable Language)
- Type 1 grammars / content sensitive grammars
- Type 2 grammars / Context free grammars
- Type 3 grammars / Regular grammar.

1) Type 0 grammars

- No restrictions on the production rules
- Production rule is of the form

$$\alpha \rightarrow \beta \mid \alpha \neq \beta$$

Where $\alpha, \beta \rightarrow$ can be strings composed by terminals and non-terminals.

- This grammar can be modeled using Turing Machine.

2) Type 1 grammars

- Content sensitive grammar (CSG) :
- Production rule is of the form

$$\alpha A \beta \Rightarrow \alpha \gamma \beta$$

Here $A \rightarrow$ Non-terminal symbol

$\alpha, \beta, \gamma \rightarrow$ combination of terminals and non-terminals.

- This grammar can be modeled using linear bounded automata.

Scanned by CamScanner

3) Type 2 grammar:

- Context Free Grammar (CFG).
- production rule is of the form $A \rightarrow \alpha$

  A - Non terminal symbol

  α - Terminal or non-terminal symbol.

- This grammar can be modeled using push down automata.

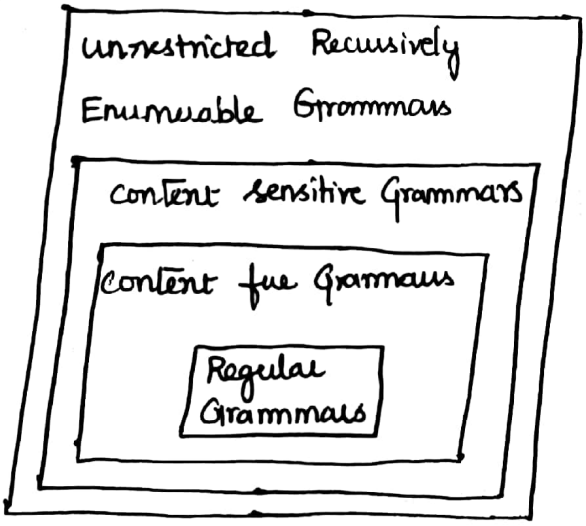
4) Type 3 grammar:

- Regular grammar that describe regular / formal languages.
- production rule consist of,
  - only one terminal at the left hand side.
  - Right hand side having a single terminal and may or may not be followed by non terminals.

    $A \rightarrow a$, $A \rightarrow aB$

- This grammar can be modeled using finite automata.

Chomsky hierarchy.



$\Rightarrow$ Regular grammars $\subseteq$ Context free grammars $\subseteq$ Context sensitive grammars $\subseteq$ Unrestricted grammars.

Context free Language and Grammars.

Definition: The context free grammar can be formally defined as a set denoted by $G=(V,T,P,S)$ where V and T are set of non-terminals and terminals respectively

P is set of production rules. $NT \rightarrow NT$

$NT \rightarrow T$

S is a start symbol.

Scanned by CamScanner

Example:

$$P = \begin{cases} S \to S+S \\ S \to S*S \\ S \to (S) \\ S \to 4 \end{cases}$$

Syntax of any English statement is,

SENTENCE $\to$ NOUN VERB

NOUN $\to$ Rama / Seeta / Gopal

VERB $\to$ goes / Writes / sings.

Dervied strings: "Rama sings"

Problems:

1) Contruct the CFG for the regular expression $(0+1)^*$.

Solution: $CFG = (V, T, P, S)$

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

where $P = \{S \to 0S \mid 1S$
$S \to \varepsilon\}$

Example:

$(0+1)^* = \{\varepsilon, 0, 1, 01, 10, 00, 11, \dots\}$

2) Construct CFG for the language L which has all the strings which are all palindrome over $\Sigma = \{a, b\}$

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

$S \to aSa \mid bSb \mid a \mid b, \varepsilon$

Example: abaaba

$S \to a\underline{S}a$
$\to ab\underline{S}ba$
$\to abaSaba$
$\to aba\varepsilon aba$
$\to abaaba$

3) Construct CFG for $\{0^m 1^n \mid 1 \le m \le n\}$

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

$V = \{S, A\}$, $T = \{0,1\}$

$P = \{S \to 0S1 \mid 0A \mid 01$
$A \to 1A \mid 1\}$

Example: 00111

$S \to 0\underline{S}1$
$\to 00\underline{S}1$
$\to 001\underline{A}1$
$\to 00111$

4) Construct CFG for $L = \{a^m b^n c^p \mid m+n = p \text{ and } p \ge 1\}$

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

$V = \{S, A\}$ $T = \{a, b, c\}$

$P = \{S \to aSc \mid bAc \mid ac \mid bc$
$A \to bc \mid bc$

Scanned by CamScanner

5) Consider the alphabet $\Sigma = \{a, b, (,), +, *, \cdot, /, \ldots \epsilon\}$. Construct a CFG that generates all strings in E^* that are RE over $\{a, b\}$.

Solution:

$E \to E+E$

$E \to E*E$

$E \to E \cdot E$

$E \to E/E$

$E \to a \mid b \mid \epsilon$.

Problems: Grammar to Language

1) If $S \to aSb \mid aAb$, $A \to bAa$, $A \to ba$ is a CFG then determine CFL.

Solution:

$S \to a\underline{S}b$

$\to aa\underline{S}bb$

$\to aaa\underline{A}bbb$

$\to \underbrace{aaa}_{a^n}b\underbrace{abbb}_{b^n}$

$L(G) = \{a^n b^m a^m b^n \mid m, n \geq 1\}$

2) If $S \to aSa \mid bSb \mid \epsilon$ is CFG. Find $L(G)$.

Solution:

$S \to a\underline{S}a$

$\to aa\underline{S}aa$

$\to aa\,b\underline{S}baa$

$\to \underbrace{aab}_{W}\underbrace{baa}_{W^R}$

$\therefore L(G) = \{WW^R \mid W \in (a, b)^*\}$

3) Find the context free language for the following grammars.

(i) $S \to aSbS \mid bSaS \mid \epsilon$

(ii) $S \to aSb \mid ab$.

Solution

(i) $S \to a\underline{S}bS$

$\to ab\underline{S}aSbS$

$\to aba\underline{S}bS$

$\to abab\underline{S}$

$\to abab$

$S \to b\underline{S}aS$

$\to ba\underline{S}$

$\to ba$

$\therefore L$ containing equal number of a's and b's

(ii) $S \to aSb \mid ab$

$S \to a\underline{S}b$

$\to aa\underline{S}bb$

$\to aaabbb$

L containing equal number of a's followed by by equal number of b's

$L = \{a^n b^n \mid n \geq 1\}$

Scanned by CamScanner

Derivations, Ambiguity, Derivation tree

Derivations: Use the productions from head to body (ie) from start symbol expanding till reaches the given string.

Two types of derivations are,

(i) Left Most Derivation (LMD)

(ii) Right Most Derivation (RMD)

⇒ LMD is a derivation in which the leftmost non-terminal is replaced first from the sentential form.

(ie) $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$, then α is left sentential form.

⇒ RMD is a derivation in which rightmost non-terminal is replaced first from the sentential form.

(ie) $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$, then α is right sentential form.

Derivation tree (parse tree)

- It is a graphical representation for the derivation of the given production rules for a given CFG.

Properties

i) Root node is always a node indicating start symbol.

ii) Derivation is read from left to right

iii) Leaf nodes are always terminal nodes.

iv) Interior nodes are always non-terminal nodes.

Example:

Consider the grammar G has the production

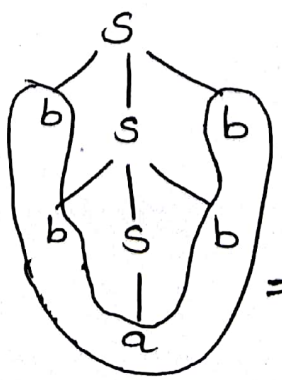
$S \rightarrow bSb \mid a \mid b$ and string "bbabb"

Derivation:

$S \rightarrow bSb$

$\rightarrow bbSbb$

$\rightarrow bbabb$



⇒ bbabb

Scanned by CamScanner

Problem 1: Construct the derivation tree for the string "aaabbabbba" using LMD and RMD. using $S \to aB \mid bA$, $A \to a \mid aS \mid bAA$, $B \to b \mid bS \mid aBB$

Solution:

LMD:

$S \Rightarrow aB$

$\Rightarrow aaBB \quad (B \to aBB)$

$\Rightarrow aaaBBB \quad (B \to aBB)$

$\Rightarrow aaabBB \quad (B \to b)$

$\Rightarrow aaabbB \quad (B \to b)$

$\Rightarrow aaabbaBB \quad (B \to aBB)$

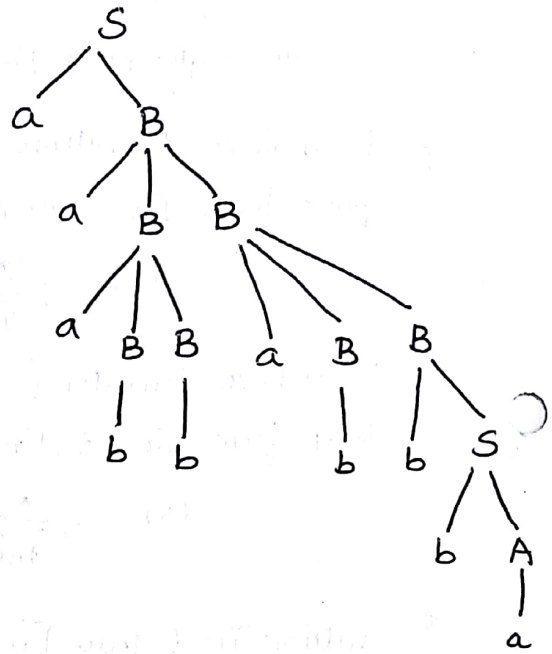
$\Rightarrow aaabbabB \quad (B \to b)$

$\Rightarrow aaabbabbS \quad (B \to bS)$

$\Rightarrow aaabbabbbA \quad (S \to bA)$

$\Rightarrow aaabbabbba \quad (A \to a)$

Parse tree:



RMD:

$S \Rightarrow aB$

$\Rightarrow aaBB \quad (B \to aBB)$

$\Rightarrow aaBbS \quad (B \to bS)$

$\Rightarrow aaBbbA \quad (S \to bA)$

$\Rightarrow aaBbba \quad (A \to a)$

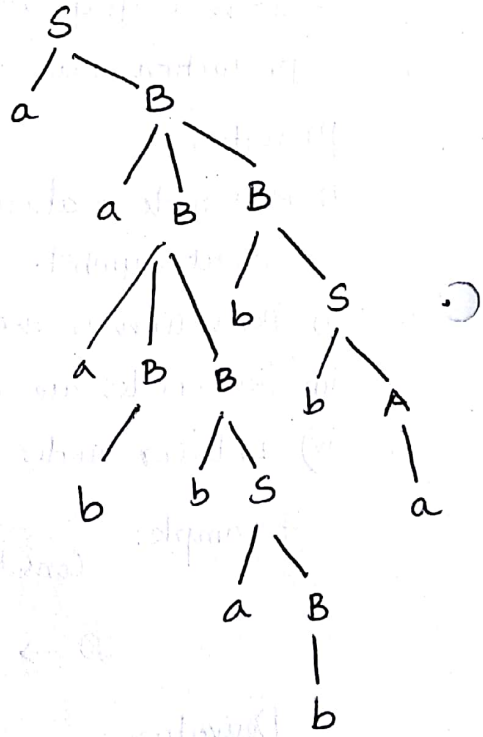
$\Rightarrow aaaBBbba \quad (B \to aBB)$

$\Rightarrow aaaBbsbba \quad (B \to bS)$

$\Rightarrow aaaBbaBbba \quad (S \to aB)$

$\Rightarrow aaaBbabbba \quad (B \to b)$

$\Rightarrow aaabbabbba \quad (B \to b)$



Scanned by CamScanner

Pbm 2: Write a grammar G to recognize all prefix expressions involving all binary arithmetic operators. Construct the parse tree for the sentence '-*+abc/de'.

Solution:

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

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

$T = \{+, -, *, /, a, b, c, d, e\}$

S is a start symbol.

Productions are,

$S \rightarrow A|B|C|D|E$

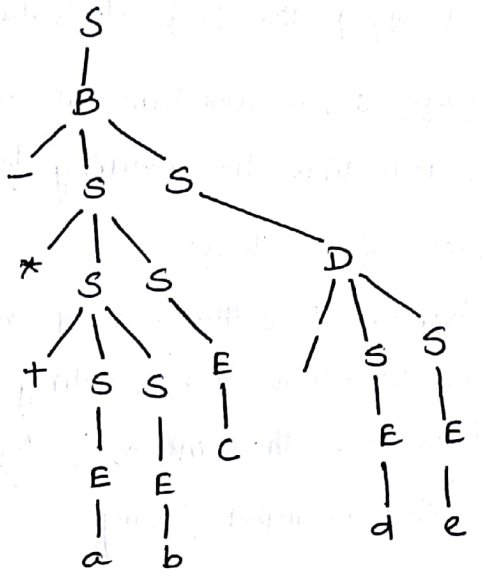
$A \rightarrow +SS$

$B \rightarrow -SS$

$C \rightarrow *SS$

$D \rightarrow /SS$

$E \rightarrow a|b|c|d|e$



Ambiguity:

If there exists more than one parse trees for a given grammar, that means there could be more than one leftmost or rightmost derivation possible and then that grammar is said to be ambiguous grammar.

Problems

1) The CFG is given by $G = (V, T, P, S)$ where $V = \{E\}$, $T = \{id\}$, $S = \{E\}$

$P = \{E \rightarrow E+E,\ E \rightarrow E*E,\ E \rightarrow id\}$. Is the grammar ambiguous?

Solution:

Consider the string id*id+id.

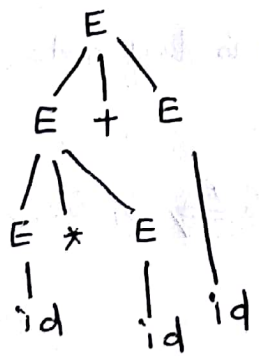
$E \Rightarrow E+E$

$\Rightarrow E*E+E$

$\Rightarrow id*E+E$

$\Rightarrow id*id+E$

$\Rightarrow id*id+id$



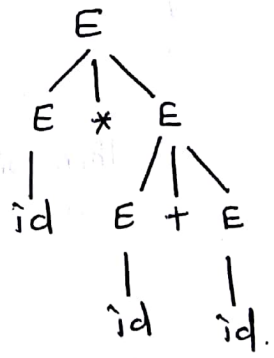
$E \Rightarrow E*E$

$\Rightarrow E*E+E$

$\Rightarrow id*E+E$

$\Rightarrow id*id+E$

$\Rightarrow id*id+id$



Here, we obtain two different parse tree for the string id*id+id.

∴ The given grammar is ambiguous.

Scanned by CamScanner

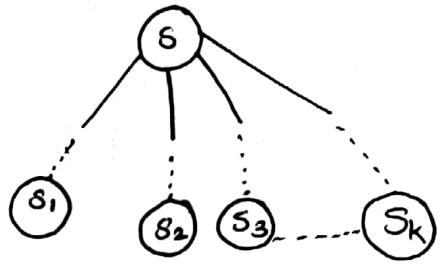
Relationship between derivation and derivation trees.

Theorem: Let $G = (V, T, P, S)$ be a context free grammar. Then $S \overset{*}{\Rightarrow} \alpha$ if and only if there is a derivation tree in grammar G which gives the string a.

Proof: For a non-terminal S there exists $S \overset{*}{\Rightarrow} w$ if and only if there is a derivation tree starting from root S and yielding W.

Basis of induction:

Assume that there is only one interior node S.
The derivation tree yielding $S_1, S_2, S_3 \ldots S_n$.
From S is that means $S \overset{*}{\Rightarrow} S_1, S_2 \ldots S_n$
$\overset{*}{\Rightarrow} a$ is input string.



Induction hypothesis:

⇒ We assume that for k-1 nodes the derivation tree can be drawn. We then prove that for k vertices also we can have a derivation tree.

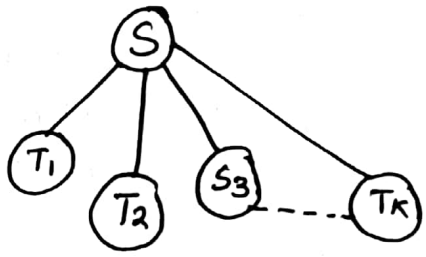
⇒ That means the input string can be derived as $S \rightarrow S_1 S_2 S_3 \ldots S_k$.

⇒ There are two cases,

(i) S_i may be a leaf variable

(ii) S_i may be an interior node yielding a.

⇒ The S derives 'a' by fewer number of k steps then $a \in S_1 S_2 S_3 S_4 \ldots S_k$.



If $a_i = S_i$ then S_i is leaf node (terminal) and if $S_i \overset{*}{\Rightarrow} a_i$ then S_i is an interior node.

This proves that $S \overset{*}{\Rightarrow} S_1, S_2, S_3, \ldots S_n \overset{*}{\Rightarrow} a$ can be obtained.

Scanned by CamScanner

Simplification of CFG.

Simplification of grammar means reduction of grammar by removing useless symbols.

- Elimination of useless symbols.
- Elimination of unit productions.
- Elimination of Null production (ε).

Elimination of useless symbols.

Any symbol is useful when it appears on the right hand side, in the production rule and generates some terminal string. If no such derivation exists then it is supposed to be an useless symbol.

Example: Eliminate the useless symbol from the following grammar.

$S \rightarrow aS \mid A \mid C$
$A \rightarrow a$
$B \rightarrow aa$
$C \rightarrow aCb$

Solution: Production with terminal symbols are

$A \rightarrow a$
$B \rightarrow aa$

Start symbol $S \rightarrow aS \mid A \mid C$

Here, there is no production for B

$\therefore$ B is useless symbol.

IIIly $S \rightarrow C \rightarrow aCb \rightarrow aaCbb \rightarrow aaaCbbb \rightarrow \cdots$

Here, no terminating symbol for C.

$\therefore$ C is useless symbol.

$\therefore$ Eliminate B and C, we get

$$\boxed{\begin{array}{l} S \rightarrow aS \mid A \\ A \rightarrow a \end{array}}$$

Scanned by CamScanner

Elimination of ε production:

If there is ε production, remove it, without changing the meaning of the grammar.

Example: Eliminate ε-productions from the CFG.

$A \rightarrow 0B1 \mid 1B1$

$B \rightarrow 0B \mid 1B \mid \varepsilon$.

Solution:

$A \rightarrow 0B1 \mid 1B1$

$B \rightarrow \varepsilon$, $A \rightarrow 01 \mid 11$

$B \rightarrow 0B \mid 1B$

$B \rightarrow \varepsilon$, $B \rightarrow 0 \mid 1$

∴ After Elimination,

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

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

Removing Unit productions

The unit productions are the productions in which one non-terminal gives another non-terminal.

$X \rightarrow Y$

Example: Eliminate the unit production from following grammar,

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow C \mid b$
$C \rightarrow D$
$D \rightarrow E \mid bC$
$E \rightarrow d \mid Ab$

Here, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$ are unit productions.

∴ $D \rightarrow E \mid bc$ can be written as $D \rightarrow d \mid Ab \mid bc$

|||ly $C \rightarrow E \mid bc$, B becomes $B \rightarrow d \mid Ab \mid bc \mid b$.

After removing unit productions.

$S \rightarrow AB$ $B \rightarrow d \mid Ab \mid bC \mid b$ $D \rightarrow d \mid Ab \mid bC$ ✗

$A \rightarrow a$ $C \rightarrow d \mid Ab \mid bC$ $E \rightarrow d \mid Ab$. ✗

Scanned by CamScanner

Chomsky normal form (CNF)

A context free grammar $G = (V, T, P, S)$ is said to be in CNF if each production in G is of the form

$X \to YZ$

$X \to \alpha$, where $X, Y, Z \in V$, and $\alpha \in T$.

[$NT \to NT.NT$
$NT \to T$]

Problems:

1) Convert the given CFG to CNF $S \to aSa \mid bSb \mid a \mid b$

Solution:

Productions are

$S \to aSa$
$S \to bSb$
$S \to a$
$S \to b$

Here, the productions which are already in CNF is

$S \to a$
$S \to b$

Apply CNF rule to other productions,

$S \to aSa \quad \mid C_a \to a$

$S \to C_a S C_a$

$$\boxed{\begin{array}{l} S \to C_a A \\ A \to S C_a \end{array}}$$

$S \to bSb \quad \mid C_b \to b$

$S \to C_b S C_b$

$$\boxed{\begin{array}{l} S \to C_b B \\ B \to S C_b \end{array}}$$

The resultant productions are,

$S \to C_a A \mid C_b B \mid a \mid b$
$A \to S C_a$
$B \to S C_b$
$C_a \to a$
$C_b \to b$

Scanned by CamScanner

2) Reduce the following grammar to Chomsky normal form.

$S \to a \mid AAB$
$A \to ab \mid aB \mid \varepsilon$
$B \to aba \mid \varepsilon$.

Solution:

Productions are,

$S \to a$
$S \to AAB$
$A \to ab$
$A \to aB$
$A \to \varepsilon$
$B \to aba$
$B \to \varepsilon$

Here, Eliminate ε production.
$A \to \varepsilon$ and $B \to \varepsilon$

$\Rightarrow$ After elimination, productions are

$S \to a \mid AAB \mid AA \mid AB \mid B$
$A \to ab \mid aB \mid a$
$B \to aba$

$\Rightarrow$ Eliminate unit production. $\to \varepsilon$

$S \to a \mid AAB \mid AA \mid AB \mid aba$
$A \to ab \mid aB \mid a$
$B \to aba$

Productions already in CNF is

$S \to a$
$S \to AA$
$S \to AB$
$A \to a$

Apply CNF rules to other productions,

$S \to \underbrace{AA}_{S}B$

$\boxed{S \to SB}$

$S \to aba$

$S \to C_a \underbrace{C_b C_a}_{C_1}$

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

$\boxed{\begin{array}{l} S \to C_a C_1 \\ C_1 \to C_b C_a \end{array}}$

$\boxed{\begin{array}{l} A \to C_a C_b \\ A \to C_a B \end{array}}$

$B \to C_a \underbrace{C_b C_a}_{C_1}$

$\boxed{B \to C_a C_1}$

$\therefore$ The resultant productions are,

$S \to SB \mid C_a C_1 \mid AA \mid AB \mid a$
$C_1 \to C_b C_a$
$A \to C_a C_b \mid a \mid C_a B$
$B \to C_a C_1$

Scanned by CamScanner

3) Convert the given CFG to CNF.

$S \to aB$ $\quad$ $A \to bAA$

$S \to bA$ $\quad$ $B \to b$

$A \to a$ $\quad$ $B \to bS$

$A \to aS$ $\quad$ $B \to aBB$

Solution:

Productions already in CNF is,

$A \to a$

$B \to b$

Apply CNF rules to other productions.

$C_a \to a$, $C_b \to b$

$S \to C_aB$

$S \to C_bA$

$A \to C_aS$

$A \to C_b \underbrace{AA}_{C_1}$

$A \to C_bC_1$

$C_1 \to AA$

$B \to C_bS$

$B \to C_a \underbrace{BB}_{C_2}$

$B \to C_aC_2$

$C_2 \to BB$

The resultant productions are,

$S \to C_aB$

$S \to C_bA$

$A \to C_aS$

$A \to a$

$A \to C_bC_1$

$C_1 \to AA$

$B \to b$

$B \to C_bS$

$B \to C_aC_2$

$C_2 \to BB$

4) Convert the grammar with productions into CNF

$A \to bAB \mid \lambda$, $B \to BAa \mid \lambda$.

5) Convert the grammar $S \to AB \mid aB$, $A \to aab \mid \varepsilon$, $B \to bbA$ into CNF.

6) Convert to chomsky Normal Form.

$S \to A \mid CB$ $\quad$ $B \to 1B \mid 1$ $\quad$ $D \to 2D \mid 2$

$A \to C \mid D$ $\quad$ $C \to 0C \mid 0$

Scanned by CamScanner

Greibach Normal Form (GNF)

A grammar $G=(V,T,P,S)$ is said to be in GNF if every production rule is of the form.

$$X \rightarrow a\alpha$$

where $a \in T$, $X \in V$

$\alpha \in V^*$

Right hand side of every productions starts with a terminal, followed by a string of variables of zero/more length.

Problems:

1) Convert the given CFG to GNF

$S \rightarrow ABA$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid \varepsilon$.

Solution:

Simplify the CFG, Eliminate ε production $A \rightarrow \varepsilon$, $B \rightarrow \varepsilon$.

$S \rightarrow ABA \mid AB \mid BA \mid AA \mid A \mid B$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Eliminate unit productions,

$S \rightarrow ABA \mid AB \mid BA \mid AA \mid aA \mid a \mid bB \mid b$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$.

Apply GNF rules,

$S \rightarrow \underline{A}BA$

$S \rightarrow aABA \mid aBA$

$S \rightarrow \underline{A}B$

$S \rightarrow aAB \mid aB$

$S \rightarrow BA$

$S \rightarrow bBA \mid bA$

$S \rightarrow AA$

$S \rightarrow aAA \mid aA$

$\therefore$ The resultant productions are,

$S \rightarrow aABA \mid aBA \mid aAB \mid aB \mid bBA \mid bA$

$S \rightarrow aAA \mid aA \mid a \mid b$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Scanned by CamScanner

2) convert given CFG to GNF where $V = \{S, A\}$, $T = \{0, 1\}$ and P is,

$S \to AA \mid 0$

$A \to SS \mid 1$

Solution:

Replace S as A_1 and A as A_2

CFG becomes, $A_1 \to A_2 A_2 \mid 0$

$A_2 \to A_1 A_1 \mid 1$

Start with $A_2 \to A_1 A_1 \mid 1$

$\underbrace{A_2 \to A_2}_{\text{left recursion.}} A_2 A_1 \mid 0A_1 \mid 1$

Introduce B_2, to eliminate left recursion $\Rightarrow$ $A_2 \to A_2 A_2 A_1$, $A_2 \to 0A_1 \mid 1$

$B_2 \to A_2 A_1 \mid A_2 A_1 B_2$, $A_2 \to 0A_1 B_2 \mid 1B_2$

The productions are

$A_2 \to 0A_1 \mid 1$

$A_2 \to 0A_1 B_2 \mid 1B_2$ $\Rightarrow$ $A_1 \to A_2 A_2 \mid 0$

$A_1 \to 0A_1 A_2 \mid 1A_2 \mid 0A_1 B_2 A_2 \mid 1B_2 A_2$ (GNF)

$B_2 \to A_2 A_1$

$B_2 \to 0A_1 \mid 1A_1 \mid 0A_1 B_2 A_1 \mid 1B_2 A_1$ (GNF)

$B_2 \to A_2 A_1 B_2$

$B_2 \to 0A_1 A_1 B_2 \mid 1A_1 B_2 \mid 0A_1 B_2 A_1 B_2 \mid 1B_2 A_1 B_2$ (GNF).

Converting Back $A_1 = S$ and $A_2 = A$

$S \to 0SA \mid 1A \mid 0SB_2 A \mid 1B_2 A \mid 0$

$A \to 0S \mid 1 \mid 0SB_2 \mid 1B_2$

$B_2 \to 0SS \mid 1S \mid 0SB_2 S \mid 1B_2 S$

$B_2 \to 0SSB_2 \mid 1SB_2 \mid 0SB_2 SB_2 \mid 1B_2 SB_2$

Scanned by CamScanner

3) Convert the grammar $S \to AB$, $A \to BS \mid b$, $B \to SA \mid a$ into Greibach Normal form. (9)

Solution:

Consider the grammar,

$$S \to AB, \quad A \to BS \mid b, \quad B \to SA \mid a$$

Assume $S = A_1$, $A = A_2$, and $B = A_3$.

$$A_1 \to A_2 A_3$$
$$A_2 \to A_3 A_1 \mid b$$
$$A_3 \to A_1 A_2 \mid a$$

Consider,

$$A_3 \to A_1 A_2 \mid a$$

$$A_3 \to A_2 A_3 A_2 \mid a$$

$$A_3 \to A_3 A_1 A_3 A_2 \mid b A_3 A_2 \mid a$$

Now,

$$\underbrace{A_3 \to A_3}\, A_1 A_3 A_2, \quad A_3 \to b A_3 A_2 \mid a$$

Introduce B_3 to eliminate left recursion $\Rightarrow$

$$B_3 \to A_1 A_3 A_2 \mid A_1 A_3 A_2 B_3$$
$$A_3 \to b A_3 A_2 \mid a \mid b A_3 A_2 B_3 \mid a B_3. \quad (GNF)$$

$$A_2 \to A_3 A_1 \mid b$$

$$A_2 \to b A_3 A_2 A_1 \mid a A_1 \mid b A_3 A_2 B_3 A_1 \mid a B_3 A_1 \mid b \quad (GNF)$$

$$A_1 \to A_2 A_3$$

$$A_1 \to b A_3 A_2 A_1 A_3 \mid a A_1 A_3 \mid b A_3 A_2 B_3 A_1 A_3 \mid a B_3 A_1 A_3 \mid b A_3$$

$$B_3 \to A_1 A_3 A_2$$

$$\to b A_3 A_2 A_1 A_3 A_3 A_2 \mid a A_1 A_3 A_3 A_2 \mid b A_3 A_2 B_3 A_1 A_3 A_3 A_2 \mid$$
$$a B_3 A_1 A_3 A_3 A_2 \mid b A_3 A_3 A_2.$$

Scanned by CamScanner

$B_3 \rightarrow A_1A_3A_2B_3$

$\rightarrow bA_3A_2A_1A_3A_3A_2B_3 \mid aA_1A_3A_3A_2B_3 \mid bA_3A_2B_3A_1A_3A_3A_2B_3 \mid$

$aB_3A_1A_3A_2B_3 \mid bA_3A_3A_2B_3.$

$\therefore$ The resultant productions are,

$A_1 \rightarrow bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2B_3A_1A_3 \mid bA_3$

$A_2 \rightarrow bA_3A_2A_1 \mid aA_1 \mid bA_3A_2B_3A_1 \mid aB_3A_1 \mid b$

$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2B_3 \mid aB_3$

$B_3 \rightarrow bA_3A_3A_2bA_3A_2A_1A_3A_2 \mid aA_1A_3A_3A_2 \mid$

$bA_3A_2B_3A_1A_3A_3A_2 \mid aB_3A_1A_3A_3A_2 \mid$

$bA_3A_2A_1A_3A_3A_2B_3 \mid aA_1A_3A_3A_2B_3 \mid$

$bA_3A_2B_3A_1A_3A_3A_2B_3 \mid aB_3A_1A_3A_2B_3 \mid bA_3A_3A_2B_3.$

Scanned by CamScanner