3 Regular Expression, Grammar and Language

3.1 REGULAR EXPRESSION

- Regular expression represents a regular language.
- For every regular expression, regular language can be generated and for every regular langauge, a regular expression is possible.
- Similar to arithmetic, where operations + and × are used to build up expressions such as (5 + 3) × 4;
  Regular operations are used to build up expressions which describe languages are known as a regular expression.
  Example: $(0+1)0^*$
- The notation involves a combination of strings of symbols from some alphabets $\sum$, parentheses and the operators +, . and *

Example: If L = {∈, aa, aaaa, aaaaaa, ...} then L can be expressed as (aa)*.

Application of regular expression:

a) In Unix grep or equivalent commands for finding strings.

b) In lexical analyser generators such as Lex or flex.

Introduction: Formal definition:

Definition

Let $\sum$ be a given alphabet, then

a) $\phi, \in, a \in \sum$ are all regular expressions.

These are called as primitive regular expressions.

b) If r_1 and r_2 are regular expressions, then $r_1 + r_2$, r_1r_2, r_1^* are also regular expressions.

c) A string is a regular expression if and only if it can be derived from the primitive regular expression by a finite number of applications of the rules mentioned in b).

Note:

Don't confuse between the Regular expression ∈ and ϕ.

- The regular expression ∈ represents the language containing a single string which is empty string.
- The regular expression ϕ represents the language that does not contain any string.

Operators of regular expression:

1) Union

- If R_1, R_2 are any two regular expressions, then their union $(R_1 + R_2)$ will also be a regular expression.

2) Concatenation

- If R_1, R_2 are any two regular expressions, then their concatenation (R_1. R_2) will also be a regular expression.

3) Kleene closure

- If R_1 is any regular expression, then Kleene closure of R_1 (R_1^*) will also be a regular expression.

There are three operations on languages that the operators of regular expressions represent. These operations are:

1) Union

- The union of two languages L_1 and L_2 denoted as $L_1 \cup L_2$.
- $L_1 \cup L_2$ means a set of strings are either in L_1 or L_2 or both.

Example: $L_1 = \{10, 111\}$
$L_2 = \{\in, 01\}$, then $L_1 \cup L_2 = \{\in, 01, 10, 111\}$

2) Concatenation

- The concatenation of two languages L_1 and L_2 is denoted as $L_1 . L_2$.
- $L_1 .L_2$ is the set of all strings that can be formed by taking any string from language L_1 and concatenating it with any string from language L_2.

Example: $L_1 = \{001, 10, 111\}$
$L_2 = \{\in\}$
$L_1 .L_2 = L_1 L_2 = \{001, 10, 111\}$

Note:

$\in$ is the identity which concatenating with any string resulting same string.

Example: String 100 concatenating with epsilon, $100.\in = 100$ over $\sum = \{0, 1\}$.

3) Closure (Star closure/Kleene closure)

- The closure of any language L is represented by L^*.
- Where L^* represents the set of all strings that can be formed by taking any number of strings from L, with repetitions and concatenating all of them.

$L = \{0, 1\}$
L^* = all strings of 0's and 1's over {0, 1}

$$L^* = \bigcup_{i \geq 0} L^i$$

$L^\circ = \{\in\}$
$L^1 = L$
$L^2 = L.L$
$L^i = L.L.L$..... i times

Language associated with regular expressions:

- Regular expressions are used to describe regular languages.

- The language L(r) denoted by any regular expression 'r' is defined as:
  - **i)** ϕ is regular expression denoting { } i.e empty set.
  - **ii)** $\in$ is a regular expression denoting $\{\in\}$.
  - **iii)** For every $a \in \Sigma$, a is a regular expression denoting {a}.
- If r_1 and r_2 are two regular expressions, then:
  - **i)** $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
  - **ii)** $L(r_1.r_2) = L(r_1).L(r_2)$
  - **iii)** $L((r_1)) = L(r_1)$
    {If r_1 is a regular expression, then (r_1), a parenthesized r_1, is also a regular expression denoting the same language as r_1.}
  - **iv)** $L(r_1^*) = (L(r_1))^*$

These four rules are used to reduce L(r) to simpler components recursively.

Examples:

i) $r = a^*$
$L(r) = L(a^*) = \{\in, a, aa, aaa, ...\}$

ii) $r = a + b$
$L(r) = \{a, b\}$

iii) $r = ab$
$L(r) = \{ab\}$

iv) $r = (a + b)^*$
$L(r) = \{\in, a, b, aa, ab,.....\}$

v) $r = a^+ = a.a^*$
$L(r) = \{a, aa, ...\}$

vi) $r = a^* + ba$
$L(r) = \{\in, a, aa, aaa, ...,ba\}$

Rack Your Brain

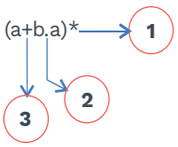
Find all strings in L((a+b)b(a+ab)*) of length less than four?

Order of precedence of regular-expression operators:

i) The star operator (*) is of the highest precedence.
ii) Next is concatenation or "dot" operator, which comes in precedence.
iii) Next is the union (+ operator) comes in precedence..

Order of precedence : * > · > +

Example: (a+b.a)* → 1, 2, 3



Note:

Since concatenation and union are associative operator. It does not matter in what order we group consecutive concatenations and consecutive unions but we shall assume grouping from left.

SOLVED EXAMPLES

Q1 **Which of the following is correct?**

1) $(r^*)^+ = r^+$ **2)** $(r^*)^* = r^*$

Sol:

1) **LHS:** $(r^*)^+ = \{\in, r, rr, ...\}$

RHS: $r^+ = \{r, rr, ...\}$

LHS ≠ RHS

2) **LHS:** $(r^*)^* = \{\in, r^*, r^*r^*, ...\}$

$= \{\in, r^*, r^*, ...\}$

$= \{\in, r, rr, ...\}$

RHS: $r^* = \{\in, r, rr, ...\}$

LHS = RHS

Option 2) is correct.

Q2 **Which of the following is incorrect?**

1) $r^* + r^+ = r^*$ **2)** $r^*.r^* = r^+$

Sol:

1) LHS: $r^* + r^+ = \{\in, r, rr, ...\} \cup \{r, rr, ...\}$

$= \{\in, r, rr, ...\}$

RHS: $r^* = \{\in, r, rr, ...\}$

LHS = RHS

2) LHS: $r^*. r^* = \{\in, r, rr, ...\}. \{\in, r, rr, ...\}$

$= \{\in, r, rr, ...\}$

RHS: $r^+ = \{r, rr, rrr, ...\}$

LHS ≠ RHS

Option 2) is incorrect.

Q3 **r^* is finite iff $r = \phi$ or $r = \in$**

1) True **2) False**

Sol: If $r = \phi$, then $r^* = \phi^* = \{\in\}$
If $r = \in$, then $r^* = \in^* = \{\in\}$
Option 1) True.

Q4 **Which of the following is correct?**

Let $r_1 = (a^*b)^*$ and $r_2 = (a + b^*)^*$

1) $L(r_1) = L(r_2)$

2) $L(r_1) \subset L(r_2)$

3) $L(r_1) \supset L(r_2)$

4) None of these

Sol: **Given:**
$r_1 = (a^*b)^*$ and $r_2 = (a+b^*)^*$
$L(r_1) = \{\in, b, ab, aab,.....\}$
$L(r_2) = \{\in, a, b, aa, ab, ba, bb, aab,.....\}$
So, $L(r_1) \subset L(r_2)$
Option 2)

Rack Your Brain

Identify which of the following are identical?

I) a* **II)** (aa)* **III)** (aa*)a **IV)** (a + ∈)a*

1) I and III only **2)** I and IV only **3)** I, III and IV only **4)** None of these

Q5 **Which of the following regular expression does not generate string containing "baa" as substring?**

1) $a^*(ba)^*$ **2) $a^*b^*(ba)^*a$** **3) $(ab^*+a)^*(ab)^*b^*a^*$** **4) $(bba^* +b)^*a$**

Sol:

Option 1) : $a^*(ba)^* = \{\in, a, ba, aba, baba, ...\}$
It cannot generate string containing 'baa' as a substring.

Option 2) : $a^*b^*(ba)^* a = \{a, aa, ba, baa, ...\}$
It can generate string containing 'baa' as a substring.

Option 3) : $(ab^*+ a)^* (ab)^* b^*a^* = \{\in, a, b, aa, bb, ab, ba, baa, abba,.....\}$
It can generate string containing 'baa' as a substring.

Option 4) : $(bba^*+ b)^* a = \{a, ba, bba, bbaa, ...\}$
It can also generate string containing 'baa' as a substring.

Previous Years' Question

Which of the following statements is **TRUE** about the regular expression 01*0?

(GATE IT-2005)

1) It represents a finite set of finite strings
2) It represents an infinite set of finite strings
3) It represents a finite set of infinite strings
4) It represents an infinite set of infinite strings

Sol: 2)

Equivalence between regular language and regular expression:

- For every regular language, there is a regular expression and for every regular expression, there is a regular language.

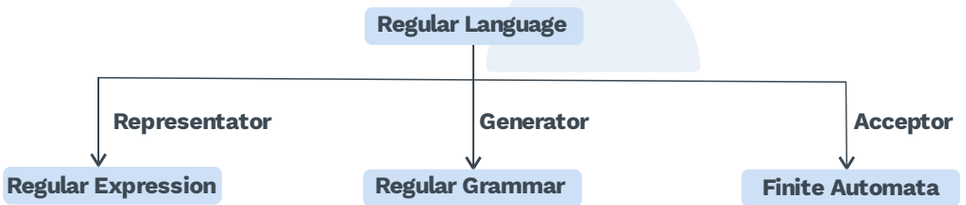


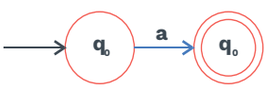
Fig. 3.1

Theorem

"Let r be a regular expression, then there exists some non-deterministic finite acceptor that accepts L(r). Consequently, L(r) is a regular language."

Example:

NFA accepts ϕ



NFA accepts {a}

Examples of equivalence between Regular language and regular expression:

SOLVED EXAMPLES

Q1 **Find the regular expression over $\sum$ = {a, b} for the language :**

i) contains all strings including $\in$

ii) contains all strings excluding $\in$

Sol:

i) L = {$\in$, a, b, aa, ab, ba, bb, ...}

$$\boxed{RE = (a+b)^*}$$

ii) L = {a, b, aa, ab, ba, bb, ...}

$$\boxed{RE = (a+b)^+}$$

Q2 **Regular expressions for the language over $\sum$ = {a, b} where:**

i) set of all strings start with a

ii) set of all strings that ends in 'b'

iii) set of all strings contains substring 'ab'

Sol:

i) L = {a, aa, ab, ...}

$$\boxed{RE = a(a+b)^*}$$

ii) L = {b, ab, bb, ...}

$$\boxed{RE = (a+b)^*b}$$

iii) L = {ab, aab, bab, ...}

$$\boxed{RE = (a+b)^*ab(a+b)^*}$$

Q3 **Find the regular expression for the language over $\sum$ = {a, b} where**

i) all the strings having 3rd symbol from left is 'b'

ii) all the strings having 3rd symbol from right is 'a'

Sol:

i) L = {bab, bbb, aab, abba, baba, ...}

$$\boxed{RE = (a+b)(a+b)b(a+b)^*}$$

ii) L = {aab, aba, abb, baba, ...}

$$\boxed{RE = (a+b)^*a(a+b)(a+b)}$$

Q4 **Find the regular expression for the language over $\sum$ = {a, b}**

i) where strings start and end with same symbol.

ii) where strings start and end with different symbol.

Sol:

i) L = {a, b, aa, aba, bb, bab, ...}

$$\boxed{RE = a(a+b)^*a + b(a+b)^*b + a + b}$$

ii) L = {ab, ba, aab, ...}

$$\boxed{RE = a(a+b)^*b + b(a+b)^*a}$$

Q5 **Find the regular expression for the language over $\sum$ = {a, b} where,**

i) set of all strings having length exactly 2

ii) set of all strings having length atleast 2

iii) set of all strings having length atmost 2

Sol:

i) L = {aa, ab, ba, bb}

RE = ab + aa + bb + ba = a(a+b)+b(a+b)

$$\boxed{RE = (a+b)(a+b)}$$

Similarly, set of all strings having length exactly 3 over $\sum$ = {a, b}

RE = (a + b) (a + b) (a + b)

ii) L = {aa, ab, ba, bb, aaa, aab,.....}

$$\boxed{RE = (a+b)(a+b)(a+b)^*}$$

iii) L = {$\in$, a, b, aa, ab, ba, bb}

RE = $\in$+ a + b + ab + aa + ba + bb

$$\boxed{RE = (a+b+\in)(a+b+\in)}$$

Q6 **Find the regular expression over $\sum$ = {a, b} for the language which accepts**

i) Even length string

ii) Odd length string

iii) All the strings divisible by 3

iv) All the strings where w $\cong$ 2 (mod 3)

v) All the strings where number of a's are exactly equal to 2.

vi) All the strings where number of a's are 'atleast 2'

vii) All the strings where number of a's are 'atmost 2'.

viii) All the strings where number of a's are even.

Sol:

i) Even length string

L = {∈, aa, ab, ba, bb, ...}

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

ii) Odd length string

L = {a, b, aaa, ...}

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

iii) all the strings divisible by 3:

L = {aaa, aab, aba, ...}

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

iv) all the strings where w ≅ 2 (mod 3)

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

v) all the strings the where the number of a's are exactly 2.

L={aa,aab,aba,baa.....}

$$RE = b^*ab^*ab^*$$

vi) all the strings where the number of a's are 'at least 2'.

L = {aa, aab, aba, aaa, ...}

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

vii) all the strings where the number of a's are 'atmost 2'.

L = {∈, a, ab, ba, aa, ...}

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

viii) all the strings where the number of a's are even.

L = {∈, b, bb, aa, aab, aba .....}

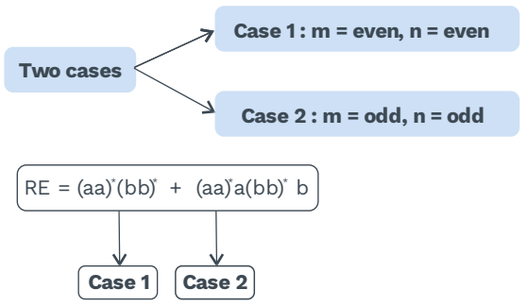
$$RE = (b^* ab^* ab^*)^* + b^*$$

- $(b^* ab^* ab^*)^*$ → **represents number of a's = 2, 4, 6.....**
- b^* → **For 0 number of a's**

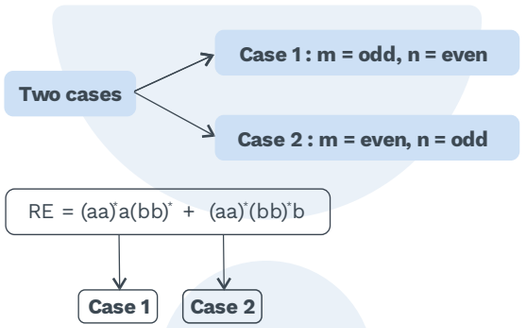
Q7 **Find the regular expression for the language :**

i) L = {$a^m b^n$ | m + n = even} **ii) L = {$a^m b^n$ | m + n = odd}**

Sol: **i)** $L = \{a^m b^n \mid m + n = \text{even}\}$



ii) $L = \{a^m b^n \mid m + n = \text{odd}\}$



Q8 **Find the regular expression for the language over $\sum$ = {a, b} :**

i) string starts with 'a' and |w| = even

ii) strings starts with 'a' and |w| = odd

Sol: **i)** $L = \{aa, ab, \ldots\ldots\ldots\ldots\}$

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

ii) $L = \{a, aaa, aab, \ldots\ldots\ldots\}$

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

Note:

A regular expression is not unique for a language. It means for any regular language, there are more than one (Infinite) regular expression possible.

Properties of regular expressions:

Let r, s and t be some regular expression.

i) Commutativity:

In this property, after changing the position of operands result will remain the same.

$$r + s = s + r$$
$$r.s \neq s.r$$

For some case:

r.s = s.r Example: r = a, s = a*

ii) Associativity:

Properties which allow us to regroup the operands when the operand is applied twice.

$$r + (s + t) = (r + s) + t$$
$$r.(s.t) = (r.s).t$$

iii) Distributive:

Property which involves two operators and states that one operator can be pushed down to be applied to each argument of the other operator individually.

Left distributive : $r(s + t) = rs + rt$
Right distributive : $(s + t)r = sr + tr$

Note:

$r + s.t \neq (r + s) . (r + t)$

iv) Identity:

When an operation is done between operand in the set and the identity element it results in the operand itself.

$\phi + r = r + \phi = r$ (ϕ is identity for union)
$\in .r = r. \in = r$ ($\in$ is identity for concatenation)

v) Annihilator:

$\phi.r = r.\phi = \phi$ (ϕ is annihilator for concatenation)

vi) Idempotent:

An operator is idempotent; if we apply it to two same operands, the output will be that operand itself.

$$r + r = r \text{ (Idempotence law for union)}$$

vii) Closure:

- $\phi^* = \in$
- $\phi^+ = \phi$
- $\in^+ = \in$
- $\in^* = \in$
- $\in + rr^* = r^*$
- $(r^*)^* = r^*$
- $r^+ = rr^* = r^*r$
- $(r^+)^* = r^*$
- $(r^*)^+ = r^*$
- $(r^+)^+ = r^+$
- $r^*r^* = r^*$
- $r^+r^* = r^+$
- $r^*r^+ = r^+$
- $(pq)^*p = p(qp)^*$ but $\begin{Bmatrix}(pq)^*q \neq p(qq)^* \\ (pq)^*r \neq p(qr)^*\end{Bmatrix}$

Note:

- $(a+b)^* = (a^*+b^*)^* = (a+b^*)^* = (a^*+b)^* = (a^*b^*)^*$
  $= (b^*a^*)^* = a^*(ba^*)^* = b^*(ab^*)^*$
- $(r+s)^* = (r+s)^* r^* = r^*(r+s)^*$

Previous Years' Question

Which one of the following regular expressions represents the language; set of all binary strings having two consecutive 0's and two consecutive 1's? **(GATE-2016 (set-1))**

1) (0+1)* 0011(0+1)*+ (0+1)*1100(0+1)*

2) (0+1)* (00(0+1)* 11+ 11(0+1)*00)(0+1)*

3) (0+1)* 00(0+1)*+ (0+1)*11(0+1)*

4) 00(0+1)* 11+11(0+1)*00

Sol: 2)

Rack Your Brain

Which of the following are equivalent?

A) (ab)*b **B)** a(bb)* **C)** a(ba)* **D)** (ab)*a

1) A) and C) only **2)** A) and D) only

3) C) and D) only **4)** B) and C) only

Rack Your Brain

Which of these are equivalent?

I) ∈ + r + rrr* **II)** ∈ + rr* **III)** r*

1) I and III only **2)** II and III only

3) All are equal **4)** None of these

Previous Years' Question

Which one of the following languages over the alphabet {0,1} is described by the regular expression :

(O+1)* 0(0+1)*0(0+1)*? **(GATE-2009)**

1) The set of all strings containing the substring 00

2) The set of all strings containing at most two 0's

3) The set of all strings containing at least two 0's

4) The set of all strings that begin and end with either 0 or1

Sol: 3)

3.2 EQUIVALENCE BETWEEN FINITE AUTOMATA AND REGULAR EXPRESSION

- Regular Expressions and Finite Automata are equivalent in their descriptive power.
- We can convert any Regular Expression into a finite automaton that recognises the language it describes and vice-versa.
- Definition of Regular language itself says the regular language is one that is recognised by some finite automaton.
- In other to show that the regular expressions define the same class of languages (i.e. regular language) accepted by finite automata, we must show that:

i) For every language, for which we have finite automata, there exists a regular expression also.

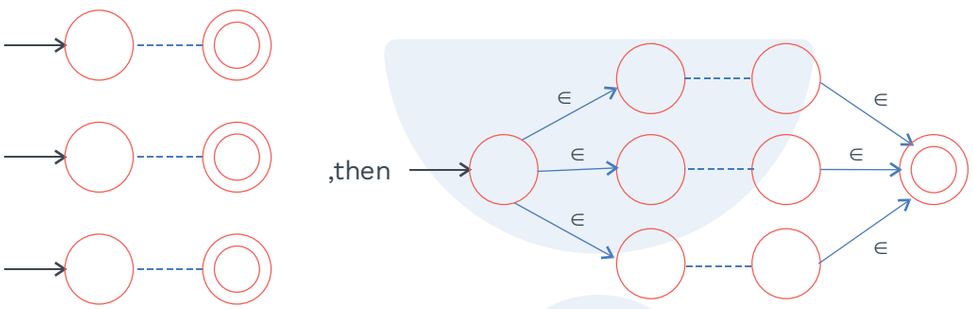
ii) Similarly, for every language, for which we have a regular expression there exists a finite automata also.

State elimination method:

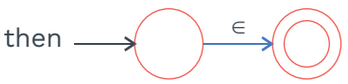
- State Elimination method can be applied for NFA, DFA, ∈-NFA to convert the finite machine into a regular expression.

Rules:

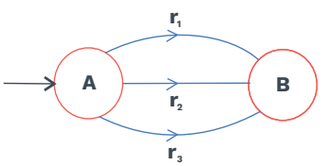
1) There should be only one initial and final state.



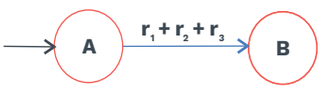
2) Separate initial state and accepting (final) state using ∈ transition,



3) Simplify parallel edge (parallel edge can be combined to a single edge),

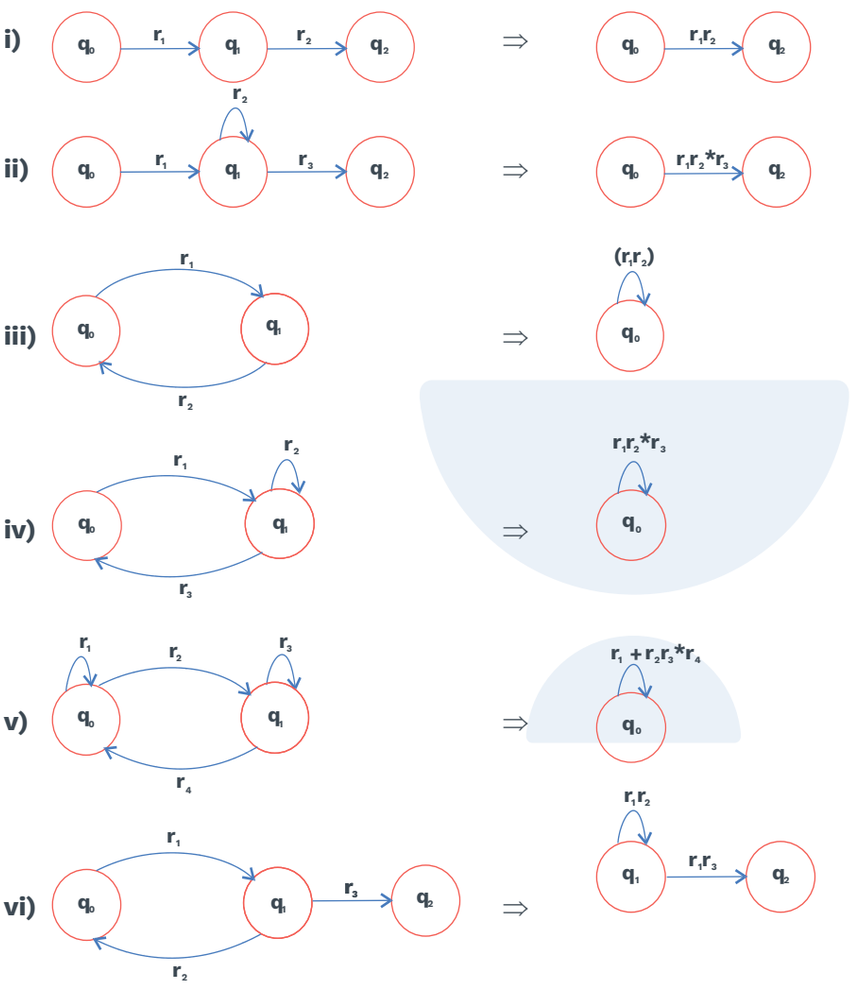


Then,

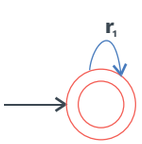


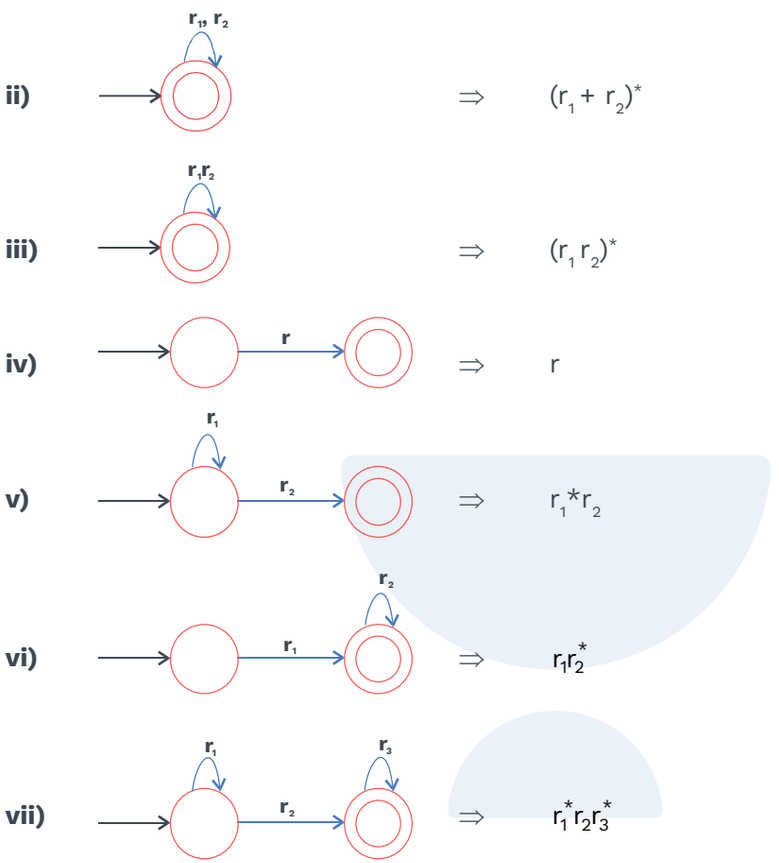
4) Elimination of nodes/states.

Example:



5) Continue to do the elimination of states till the transition graph takes any of these from:

Finite Automata	**Regular Expression**
i) 	$\Rightarrow \quad r_1^*$



Arden's method:

Let P, Q be the regular expression over $\sum$.

If P does not contain $\in$, then Regular expression R can be written as:

$\boxed{R = Q + RP}$ and it has a unique solution $\boxed{R = QP^*}$

It means that if we get an equation in the form of R = Q + RP, then the regular expression we get as a solution will be $\boxed{R = QP^*}$.

Note:

If P contains $\in$, then regular expression R will have an infinite number of solutions.

Proof of arden's theorem:

$R = Q + RP$

$= Q + (Q+RP)P$

$= Q + QP + RPP$

Put R = Q + RP again. On solving, it recursively

$R = Q + QP + QP^2 + QP^3 + \ldots\ldots\ldots\ldots$

$R = Q(\in + P + P^2 + \ldots\ldots\ldots\ldots)$

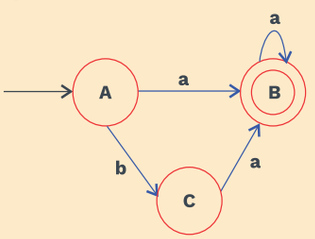
$\boxed{R = QP^*}$

Note:

We can write equation for every other state and start state based on all incoming arrows.

SOLVED EXAMPLES

Q1 **Calculate regular expression for A, B and C using Arden's lemma.**



Sol:

$A = \in$

$B = Aa + Ba + Ca$

$C = Ab$

$\boxed{A = \in}$

$\boxed{C = \in .b = b}$

$B = \in .a + Ba + ba$ (Put values of A and C in B's equation)

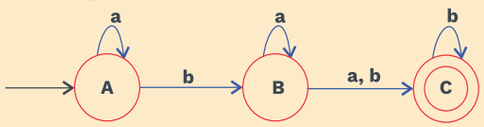
$B = \in .a + ba + Ba$

$B = \underset{R}{\square} \underbrace{a + ba}_{Q} + \underbrace{\square a}_{RP}$ (P does not contain $\in$)

$\therefore \quad B = QP^*$

$B = (a + ba)a^*$

Q2 **Calculate regular expression for A, B, C?**



Sol: Write equations based on given Finite Automata:

$A = \in + Aa$

$B = Ab + Ba$

$C = B(a + b) + Cb$

$\Rightarrow$ $\underset{R}{A} = \underset{Q}{\in} + \underset{RP}{Aa}$ (P does not contain $\in$)

$A = \in a^*$

$\boxed{A = a^*}$

$\Rightarrow$ $\underset{R}{B} = Ab + Ba = \underset{Q}{a^*b} + \underset{RP}{Ba}$ (P does not contain $\in$)

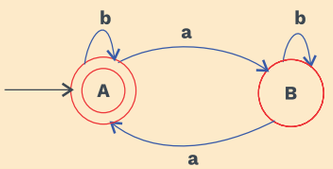
$\boxed{B = (a^*b)a^*}$

$\Rightarrow$ $C = B(a+b) + Cb$

$\underset{R}{C} = \underbrace{(a^*b)a^*(a+b)}_{Q} + \underset{RP}{Cb}$ (P does not contain $\in$)

$\boxed{C = a^*ba^*(a+b)b^*}$

Q3 **Calculate regular expression for A and B?**



Sol: Write equations based on given Finite Automata:

$A = \in + Ab + Ba$

$B = Aa + Bb$

$\Rightarrow$ $\underset{R}{B} = \underset{Q}{Aa} + \underset{RP}{Bb}$ (P does not contain $\in$)

$B = Aab^*$

$\Rightarrow$ $A = \in + Ab + Ba$

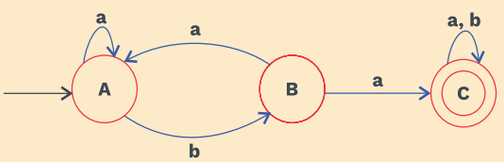
$A = \in + Ab + Aab^*a$

$\underset{R}{A} = \underset{Q}{\in} + \underbrace{A(b + ab^*a)}_{RP}$ (P does not contain $\in$)

$\boxed{A = \in (b + ab^*a)^*}$

$\boxed{B = (b + ab^*a)^* ab^*}$

Q 4 **Calculate regular expression for A , B and C?**



Sol: Write equations based on given Finite Automata:

$A = \in + Aa + Ba$

$B = Ab$

$C = Ba + C\ (a + b)$

$\Rightarrow \quad A = \in + Aa + Ba$

$A = \in + Aa + Aba$

$\underset{R}{A} = \underset{Q}{\in} + \underbrace{A(a + ba)}_{RP}$ (P does not contain $\in$)

$\boxed{A = \in (a + ba)^*}$

$\Rightarrow \quad B = Ab$

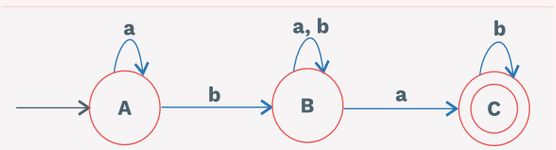
$\boxed{B = (a + ba)^* b}$

$\Rightarrow \quad C = Ba + C\ (a+b)$

$\underset{R}{C} = \underbrace{(a + ba) * ba}_{Q} + \underbrace{C(a + b)}_{RP}$

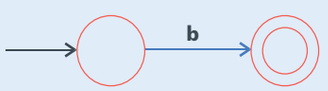
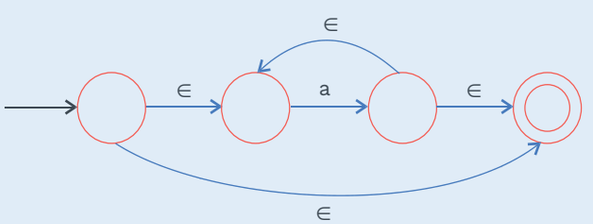
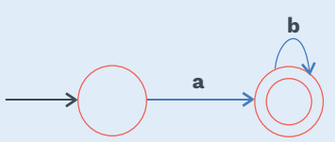
$\boxed{C = (a + ba)^* ba(a + b)^*}$

Rack Your Brain



Calculate regular expression for A, B, C using Arden's Theorem?

Conversion of finite automata to regular expression and vice-versa:
Conversion from regular expression to finite automata:

	Regular Expression	Finite Automata
i)	ϕ	
ii)	a	
iii)	b	
iv)	ab	
v)	a^*	
vi)	ab^*	

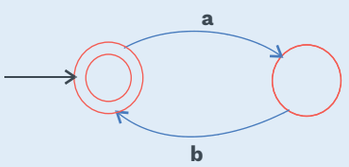
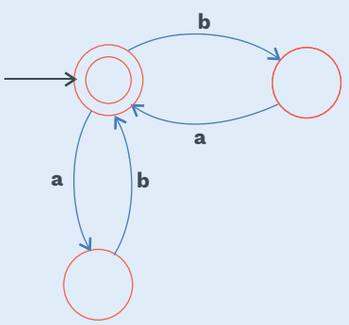
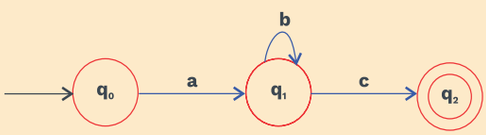
	Regular Expression	Finite Automata
vii)	$(ab)^*$	
viii)	$(ab + ba)^*$	

Table 3.1

Conversion from Finite Automata to Regular Expression:

SOLVED EXAMPLES

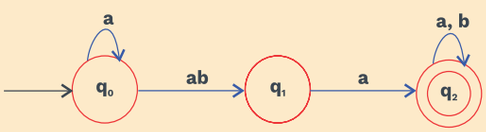
Q1 **Using State Elimination Method, find the regular expression for the given Finite Automata:**



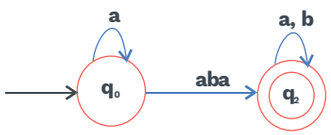
Sol: 

$RE = ab^*c$

Q2 **Using State Elimination Method, find the regular expression for the given Finite Automata:**

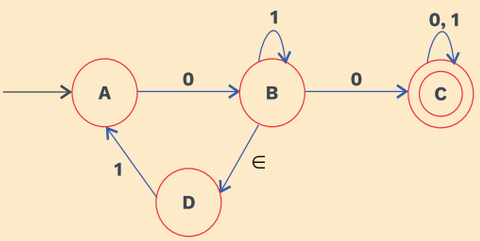


Sol:

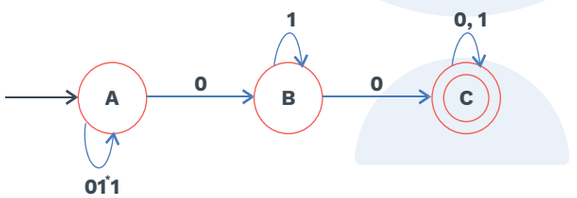


$$\boxed{RE = a^*aba(a+b)^*}$$

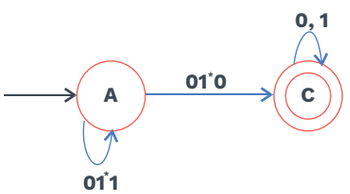
Q3 **Using State Elimination Method, find the regular expression for the given Finite Automata:**



Sol:

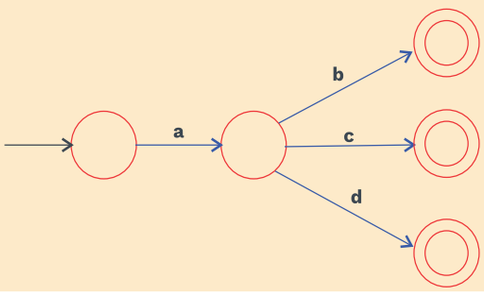


$\Rightarrow$

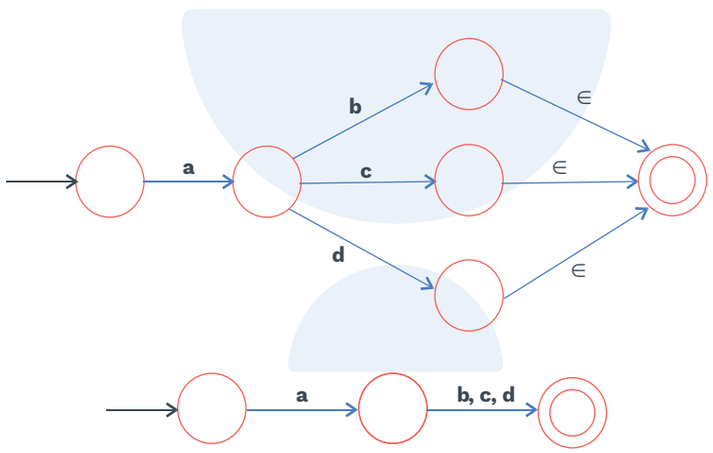


$\Rightarrow$ $\boxed{RE = (01^*1)^*01^*0(0+1)^*}$

Q4 **Using State Elimination Method, find the regular expression for the given Finite Automata:**

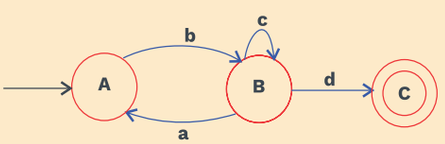


Sol:

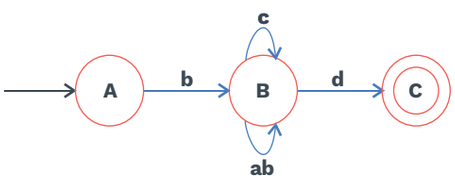


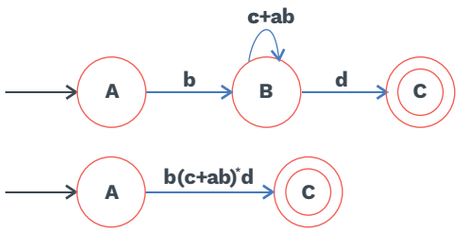
RE = a(b + c + d)

Q5 **Using State Elimination Method, find the regular expression for the given Finite Automata:**



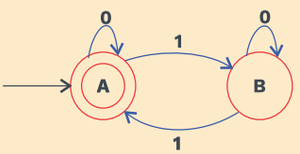
Sol:



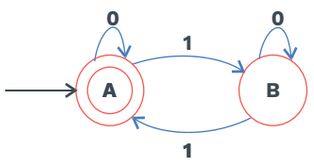


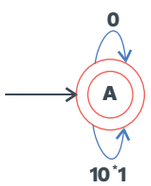
$RE = b(c + ab)^* d$

Q6 **Using State Elimination Method, find the regular expression for the given Finite Automata:**



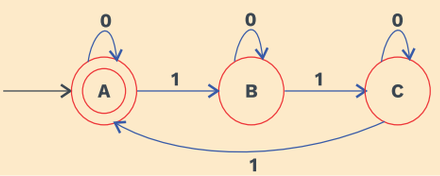
Sol:



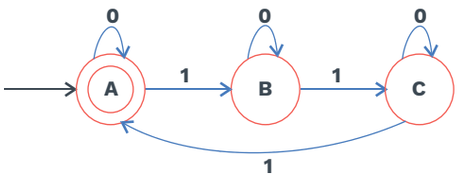


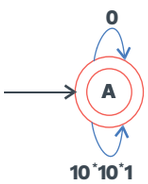
$RE = (0 + 10^*1)^*$

Q7 **Using State Elimination Method, find the regular expression for the given Finite Automata:**



Sol:





$\Rightarrow \quad (0+10^*10^*1)^*$

$$RE = \left(0 + 10^*10^*1\right)^*$$

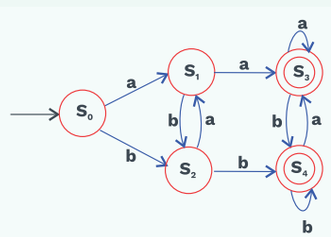
Previous Years' Question

Consider the regular expression:

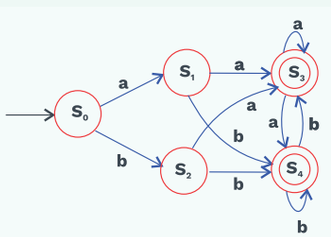
$$R = (a + b)^* (aa+bb) (a + b)^*$$

which deterministic Finite Automaton accepts the language represented by the regular expression R? **(GATE (IT) 2007)**

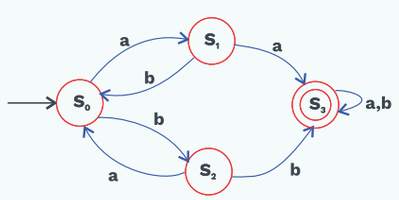
1)



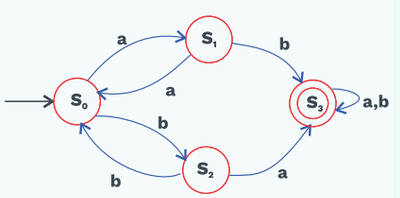
2)



3)



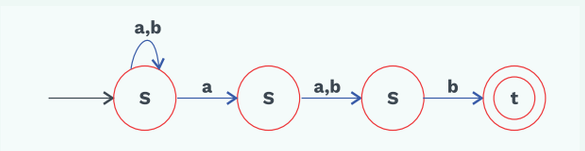
4)



Sol: 1)

Previous Years' Question

Which regular expression best describes the languages accepted by the non-deterministic automaton below? **(GATE (IT) 2006)**



1) (a + b)* a(a + b) b
2) (abb)*
3) (a+b)*a(a + b)*b (a+b)*
4) (a+b)*

Sol: 1)

3.3 REGULAR GRAMMAR

- Regular grammar and Regular languages are equal in power i.e., for every Regular language there exists a Regular grammar and vice–versa.
- **Types of regular grammar:**

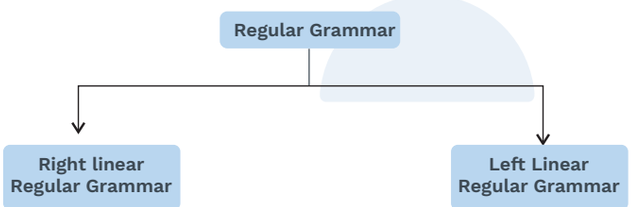


Fig. 3.2

Right and left linear regular grammar

Right linear regular grammar:

"A Grammar G = (V, T, S, P) is said to be right linear if all productions are in the form of:

$$A \rightarrow xB$$

$$A \rightarrow x$$

where A, B $\in$ V (Variables) and x $\in T^*$ (T = terminals)"

Left linear regular grammar:

"A Grammar G = (V, T, S, P) is said to be Left Linear Grammar if all productions are in the form of:

$$A \rightarrow Bx$$

$$A \rightarrow x$$

where A, B $\in$ V (Variables) and x $\in T^*$ (T = terminals)"

Note:

A regular grammar is one that is either right linear or left linear.

Example 1: Grammar $G_1 = (\{S\}, \{a, b\}, S, P_1)$ with P_1 given as:
$S \rightarrow abS/a$ is right linear
Grammar $G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$ with P_2 given as:
$S \rightarrow S_1ab$
$S_1 \rightarrow S_1ab|S_2$
$S_2 \rightarrow a$
is left linear.
Both G_1 and G_2 are regular grammar.

Example 2: The Grammar G = ({S, A, B}, {a, b}, S, P) with productions:
$S \rightarrow A$
$A \rightarrow aB/\in$
$B \rightarrow Ab$
Is not regular grammar.

Reason:
Even though here every production is in right or left linear form, but the grammar itself is neither right linear nor left linear. So, it is not Regular grammar.

Note:

The above example is linear grammar.

Linear grammar:
"A linear grammar is a grammar in which at most one variable can occur on the right side of any production, without restriction on the position of this variable."

Grey Matter Alert!

A regular grammar is always linear, but not all linear grammars are regular

Note:

The language generated by the right linear grammar is always regular.

Equivalence of regular grammar and regular language:

Theorem:

"A language L is regular if and only if there exists a regular grammar G such that L = L(G)"

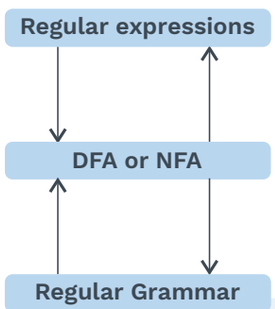


Fig. 3.3 Conversion of Regular expressions, Regular Grammar and Finite automata

Previous Years' Question

Language L_1 is defined by the grammar: **(GATE 2016 (SET-2))**

$S_1 \rightarrow aS_1b|\in$

Language L_2 is defined by the grammar:

$S_2 \rightarrow abS_2|\in$

Consider the following statements:

P : L_1 is regular.

Q : L_2 is regular.

Which one of the following is "TRUE"?

1) Both P and Q are true **2)** P is true and Q is false

2) P is false and Q is true **4)** Both P and Q are false

Sol: 3)

Rack Your Brain

Find a Regular Grammar that generates the language L $(aa^* (ab + a)^*)$

Rack Your Brain

Find a Regular Grammar for the language $L = \{a^n b^m \mid n + m \text{ is even}\}$

SOLVED EXAMPLES

Q1 **Construct the regular grammar for the following languages :**

I) $L = a^*$ **ii)** $L = a^+$

Sol:

i) $L = a^*$

$S \rightarrow Sa \mid \in$

(OR)

$S \rightarrow aS \mid \in$

ii) $L = a^+$

$S \rightarrow Sa \mid a$

(OR)

$S \rightarrow aS \mid a$

Q2 **Construct the regular grammar for the language that contains set of all the strings of a's and b's including $\in$.**

Sol: $\sum = \{a, b\}$

Regular expression = $(a + b)^*$, $L = \{\in, a, b, aa, ab, ...\}$

$S \rightarrow aS \mid bS \mid \in$

Q3 **Construct the regular grammar for the language that contains set of all the strings of a's and b's excluding $\in$.**

Sol: $\sum = \{a, b\}$

Regular expression = $(a + b)^+$, $L = \{a, b, aa, ab, ...\}$

$S \rightarrow aS \mid bS \mid a \mid b$

Q4 **Construct the regular grammar for the language that generates set of all the strings of a's and b's where every string starts with 'a'.**

Sol: Regular expression = a (a + b)*
L = {a, aa, ab, ...}

Grammar:
S → aA
A → aA|bA|∈

Conversion from finite automata to regular grammar:

Regular Grammar Corresponding to the given Finite Automata:

A → aB (There will be one production for every outgoing edge.
B → aB | bB |∈ For the final state, we will add null production in the grammar.)

It produces the Right Linear Grammar.
If we reverse the Right-Hand Side of the production of Right Linear Grammar, we will get Left Linear Grammar.

A → Ba
B → Ba | Bb |∈ (Left Linear Grammar)

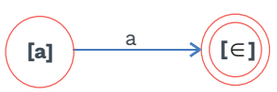
Right linear grammar to ∈-NFA:

1) If S → α is a production, then [S] is a start/initial state and all the suffix of RHS of the production will be the other states of ∈–NFA.

2) If S → α is a production, then δ (S, ∈) = [α]

3) If S → α is a production, then δ (aα, a) = [α]

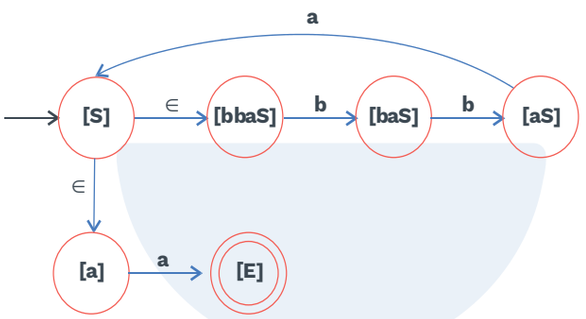
4) If S → α is a production, then for every terminal, a ∈ T
δ (a, a) = ∈ which is final state.



SOLVED EXAMPLES

Q1 **S ⟶ bbaS|a**

Sol: Q(States) = {[S], [bbaS], [baS], [aS], [a], [E]}
Initial State = q_0 = [S]
Final state = F = [E]



$\delta(S, \in) = [bbaS]$

$\delta(S, \in) = [a]$

$\delta(bbaS, b) = [baS]$

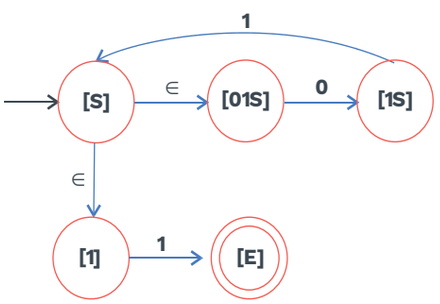
$\delta(baS, b) = [aS]$

$\delta(aS, a) = [S]$

$\delta(a, a) = [E]$

Q2 **S ⟶ 01S|1**

Sol: Q(States) = {[S], [01S], [E], [1], [1S]}
Initial State = [S]
Final state = [E] (End State)



$\delta(S, \in) = [01S]$

$\delta(S, \in) = [1]$

$\delta(01S, 0) = [1S]$

$\delta(1S, 1) = [S]$

$\delta(1, 1) = [E]$

Left linear grammar to $\in$-NFA:

Step 1: Reverse the Right-hand side of every production.
Step 2: obtain $\in$–NFA.
Step 3: Interchange initial and final state.
Step 4: Change the direction of the edges.

SOLVED EXAMPLES

Q1 **S $\longrightarrow$ S110|0**

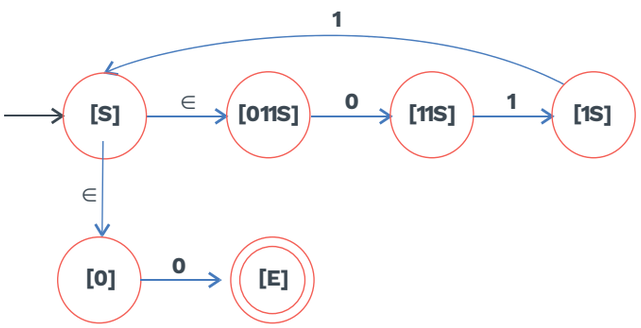
Sol:

Step-1: Reverse the right-hand side of every production and get the right linear Grammar.

S $\rightarrow$ 011S|0

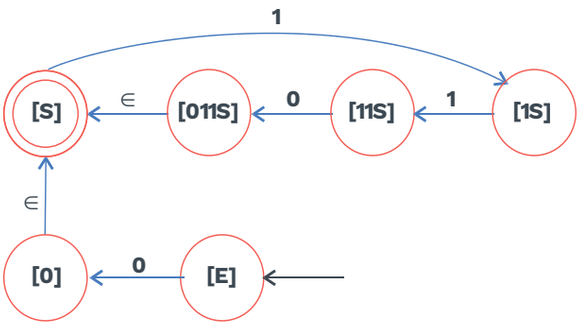
Step-2: Obtain $\in$–NFA

Q(States) = {[S], [011S], [11S], [1S], [0], [E]}



Step-3: Make the initial state as final state and the final state as initial state.

Step-4: Change the direction of the edges.



Q2 S ⟶ S100|1

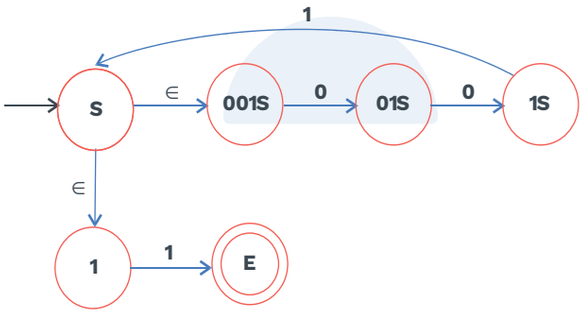
Sol:

Step-1 : Reverse the right-hand side of every production and get Right Linear Grammar.

S → 001S|1

Step-2: Convert to ∈–NFA

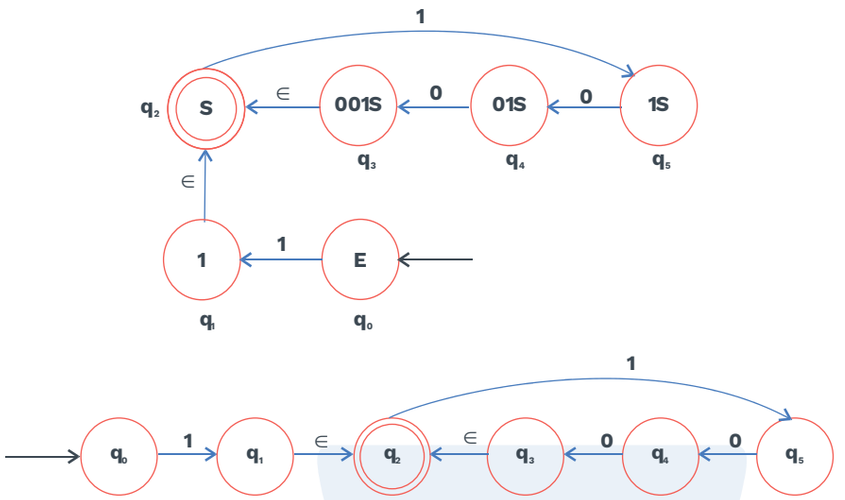
Q(States) = {[S], [001S], [01S], [1S], [1], [E]}



δ (S, ∈) = [001S]
δ (S, ∈) = [1]
δ (001S, 0) = [01S]
δ (01S, 0) = [1S]
δ (1S, 1) = [S]
δ (1, 1) = [E]

Step-3: Interchange initial and final states.

Step-4: Change the direction of edges:



3.4 REGULAR LANGUAGES

The language which is accepted by Finite Automata (or) the language which is generated by regular expression (or) the language which is generated by Regular grammar is called Regular Language.

Definition

A language L is called regular if and only if there exists some deterministic finite acceptor M such that L = L(M).

Note:

- Every formal languages must be either Regular (or) Non regular language.
- The language which is not accepted by Finite Automata is known as Non-Regular Language.
- Every finite language is Regular.
  Example : L = {b, ba, bab} is finite language then Regular .
- $L = \phi$ (Empty Language) is Regular language.
- $L = \sum^*$ (Universal Language) is Regular language.
- Regular Language can be finite (or) infinite.
- Every Non-Regular Language is Infinite.
- Every subset of a Regular language need not to be Regular.
  Example : $L_1 = \{a^m b^n | m, n \geq 1\}$ is Regular language.
  But $L_2 = \{a^m b^n | m = n\}$ which is a subset of L_1 i.e non Regular Language.

Example : $L_1 = \{a^m b^n | m, n \geq 1\}$ is Regular language.

But $L_2 = \{a^m b^n | m = n\}$ which is a subset of L_1 i.e non Regular Language.

- Every Finite subset of Regular language is a Regular Language.
- Every Finite subset of Non-Regular language is a Regular Language.
- Every subset of Non-Regular Language need not be Non-Regular.

  $L = \{a^n b^n | n \geq 0\}$ is Non-Regular Language.

  $L_1 = \{\in, ab, aabb\}$ is Regular Language.
- Superset of Regular Language can be Regular (or) cannot be Regular.

  Example : $L_1 = \{\in\}$ Regular Language.

  $L_2 = \{\in, ab\}$ is Regular Language.

  $L_3 = \{a^m b^m | m \geq 0\}$ is Non-Regular Language.
- Superset of Non-Regular Language need not be Non-Regular.

  Example : $L = \{a^n b^n | n \geq 1\}$ is Non-Regular Language.

  $L_1 \supset L$

  $L_1 = \{a^m b^n | m, n \geq 1\}$ is Regular Language.
- Every language defined over alphabet $\sum$ has a regular subset.

Closure properties of regular language

Closure properties of regular languages are defined as the certain operation on regular language, which when applied to the language, results into the language of the same type.

i) Closure under union:

Theorem:

"If L_1 and L_2 are regular languages, then $L_1 \cup L_2$ will also be a regular language only."

Proof:

Since L_1 and L_2 are regular, they have regular expressions, say $L_1 = L(R)$ and $L_2 = L(S)$, then $L_1 \cup L_2 = L(R + S)$ by the definition of the + operator for regular expressions.

ii) Closure under complementation:

Theorem:

"If L is a regular language over alphabet $\sum$, then $\overline{L} = \sum^* - L$ is also a regular language."

Proof:

Let L = L(A) for some DFA A = $(Q, \sum, \delta, q_0, F)$, then $\overline{L} = L(B)$, where B is the DFA $(Q, \sum, \delta, q_0, Q - F)$ i.e., B is exactly like A, but the accepting states of A become non-accepting states of B and vice-versa.

iii) Closure under intersection:

Theorem:

"If L_1 and L_2 are regular languages, then so is $L_1 \cap L_2$."

Proof:

By Demorgan's law:

$L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$

If L_1 and L_2 are regular, then by closure under complementation, so are $\overline{L_1}$ and $\overline{L_2}$.

Using closure under union, $\overline{L_1} \cup \overline{L_2}$ is regular.

Using closure under complementation once more

$\overline{(\overline{L_1} \cup \overline{L_2})} = L_1 \cap L_2$ is regular.

iv) Closure under difference:

Theorem:

"If L_1 and L_2 are regular languages, then so is $L_1 - L_2$."

Proof:

$L_1 - L_2 = L_1 \cap \overline{L_2}$

As we know, $\overline{L_2}$ is regular, and using the closure of intersection $L_1 \cap \overline{L_2}$ is also closed. Thus $L_1 - L_2$ is also regular.

v) Closure under reversal:

Theorem:

"If L is a regular language, so is L^R."

Proof:

Let L is Regular Language. We can construct a Non-Deterministic Finite Automata with a single final state for it.

In the transition graph for this Non-Deterministic Finite Automata,

a) Make the initial state as the final state.

b) Make the final state as the initial state.

c) Reverse the direction of all the edges.

Now the modified Non-Deterministic Finite Automata accepts w^R if and only if the original Non-Deterministic Finite Automata accepts w.

vi) Closure under homomorphism:

Homomorphism:

Definition

A string homomorphism is a function on strings that works by substituting a particular string for each symbol.

Example: The function h given by h(0) = abb and h(1) = ba is a homomorphism which replaces each 0 by abb and each 1 by ba. Thus, h(1011) = baabbbaba.

Theorem:

"If L is a regular language over alphabet $\sum$, and h is a homomorphism on $\sum$, then h(L) is also regular."

vii) Closure under Inverse homomorphism:

Inverse homomorphism:

We can apply homomorphism backwards also.

Regular languages are closed under this.

The inverse of homomorphic image of a language L is h^{-1} (L) = set of strings w in $\sum^*$ such that h(w) is in L; if h is a homomorphism from some alphabet $\sum$ to strings in another alphabet T, and L be a language over alphabet T.

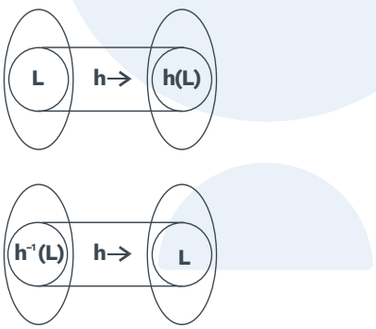


Fig. 3.4 A Homomorphism Applied in the Forward and Inverse Direction

Example 1. Let $\sum$ = {0, 1, 2} and Δ = {a, b}.

Defined h by h(0) = a, h(1) = ab, h(2) = ba.

Let L_1 = {ababa}

$h^{-1}(L_1)$ = {110, 022, 102}

Example 2. Let $\sum$ = {0, 1} and Δ = {a, b}.

Defined h by h(0) = aa, h(1) = aba.

Let L = $(ab + ba)^*a$

= {a, aba, baa, ababa, ...}

- Inverse homophoric image for a, is not possible.
- Inverse homophoric image for aba is possible.

h^{-1} (aba) = {1}

Rack Your Brain

Consider the homomorphism h from the alphabet {0, 1, 2} to {a, b} defined by : h(0) = ab, h(1)=b, h(2)=aa

a) What is h(2201)?

b) If L is the language 1*02, what is h(L)?

viii) Closure under quotient operation:

Regular languages are closed under Quotient operation.

Right quotient:

Let L_1 and L_2 be language on the same alphabet, then the right quotient of L_1 with L_2 is defined as:

$L_1/L_2 = \{x : xy \in L_1 \text{ for some } y \in L_2\}$

Example 1. $L_1 = \{01, 001, 101, 0001, 1101\}$

$L_2 = \{01\}$

$L_1/L_2 = \{\in, 0, 1, 00, 11\}$

Example 2. $L_1 = 10^*1$; $L_2 = 0^*1$

$L_1 = \{11, 101, 1001, 10001, ...\}$

$L_2 = \{1, 01, 001, 0001, ...\}$

$L_1/L_2 = \{1, 10, 100, 1000, ...\}$

$= 10^*$

Note:

Few more examples of Right Quotient:

$a/abc = \Phi$

$abc/c = ab$

$abc/abc = \in$

ix) Closure under INIT operation:

Regular sets are closed under INIT (all the prefixes of given language L) operation.

Example: L = {a, ab, aabb}

INIT(L) = {∈, a, ∈, a, ab, ∈, a, aa, aab, aabb}

= {∈, a, aa, ab, aab, aabb}

x) Closure under substitution:

Substitution: Let ∑, Δ two alphabets, then substitution is a mapping defined over the alphabet '∑' such that symbols of ∑ can be replaced

by regular language of another alphabet 'Δ'
Regular sets are closed under substitution.

Example: $\sum = \{a, b\}$

$f(a) = 0^*$

$f(b) = 01^*$

$L = a + b^*$

$f(L) = 0^* + (01^*)^*$

which is again a regular expression, which will accepted by some finite Automata.

Note:

- Finite Union of Regular languages is Regular language.
- Infinite Union of Regular language need not be a Regular language i.e. Regular language is not closed under Infinite Union.

Example: $L_1 = \{ab\}$

$L_2 = \{a^2b^2\}$

$L_3 = \{a^3b^3\}$ .....so on

$L = L_1 \cup L_2 \cup L_3 \cup$ .....

$= \{a^nb^n \mid n \geq 1\}$ which is not regular.

Note:

- Finite Intersection of Regular Language is Regular Language.
- Infinite Intersection of Regular Language need not be Regular Language.

xi) Closure under symmetric difference:

$L_1 \oplus L_2 = L_1 \Delta L_2$

$= (L_1 \cup L_2) - (L_1 \cap L_2)$

$= (L_1 - L_2) \cup (L_2 - L_1)$

As we know, Regular language is closed under union, intersection and set-difference.

Therefore, It is also closed under symmetric difference.

Rack Your Brain

The nor of two languages is :

$$\text{nor}(L_1, L_2) = \{w : w \notin L_1 \text{ and } w \notin L_2\}$$

Find whether the family of regular language is closed under the nor-operation.

Regular language (L) is closed under:

Operations	Yes/No
1) Union	Yes
2) Substitution	Yes
3) Complementation	Yes
4) Set-difference	Yes
5) Concatenation	Yes
6) Reversal	Yes
7) Kleene Closure	Yes
8) Positive Closure	Yes
9) Subset (L)	No
10) Prefix (L)	Yes
11) Suffix (L)	Yes
12) Substring (L)	Yes
13) Substitution	Yes
14) Homomorphism	Yes
15) Symmetric difference	Yes

Table 3.2 Closure Properties of Regular languages

Decision properties of regular language:

A decision property for a class of languages is an algorithm which takes a formal description of a language and says whether some property holds or not.

Example: Is language L empty?

Example: Is a particular string w in the described language?

Example: Do two descriptions of a language actually describe the same language?

i) Testing emptiness of regular languages:

Emptiness problem of Regular Language is decidable.

E_{DFA} = {<D>| D is a DFA and L(D) = ϕ}

Algorithm:

Step 1: Select the states which are not reachable from the initial state and delete those states and corresponding transitions.

Step 2: In the remaining finite Automata, if we find at least one final state, then the language accepted by the given finite Automata is non-empty otherwise, empty.

ii) Testing finiteness of regular language:

The finiteness problem of Regular language is decidable.

Algorithm:

Step 1: Remove the state which is not reachable from initial state and corresponding transitions.

Step 2: Delete the states and corresponding transitions from which we cannot reach the final state.

Step 3: In the remaining Finite Automata, if we find at least one loop and one final state, which means it accepts infinite language.

Note:

Finite Automata will accept a infinite language only when there is a loop exists and loop should be reachable from initial state, and from the loop, able to reach final state.

iii) Testing equality problem of regular languages:

The equality problem of Regular language is decidable.

EQ_{DFA} = {<D_1, D_2 >| D_1 and D_2 are DFA and L(D_1) = L(D_2)}

Is $L_1 = L_2$?

FA_1 FA_2

For regular language L_1, we can make a Finite Automata FA_1 and similarly for regular language L_2. We can make finite Automata FA_2. So, corresponding DFA will exist for both languages, which can be further minimised.

Thus, whether two regular languages are equal or not is decidable.

Note:

Consider G is a regular grammar.

$w \in L(G) \rightarrow$ Membership Problem

$L(G) = \Phi \rightarrow$ Emptiness Problem

L(G) is finite? ->Finiteness Problem

$L(G) = \Sigma^* \rightarrow$ Completeness(Totality) Problem

$L(G1) = L(G2) \rightarrow$ Equivalence of two regular language Problem

All these problems are decidable for regular language.

Pumping lemma:

- It is evident from the many examples that Regular languages can be infinite.
- However, the regular language associated with automata having finite memory imposes restrictions on the structure of Regular language.
- Pumping lemma is a theorem about regular languages which is used to prove that some of the languages are not regular.

Theorem:

"If L is an infinite regular language. Then there exists some positive integer n such that any $w \in L$, $|w| \geq n$ can be decomposed as:

w = xyz such that

i) $|y| \geq 1$

ii) $|xy| \leq n$

iii) for all $i \geq 0$, the string xy^iz is also in L."

Note:

- Every regular language satisfies the pumping lemma property.
- If a language 'L' does not satisfy the pumping lemma property, then 'L' is non-regular.

Example: $L = \{a^nb^n | n \geq 1\}$

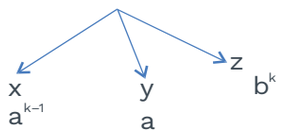
Sol: Let's assume L is a regular language (proof by contradiction)

$w \in L$

$w \in a^kb^k$

Choose a 'k' such that $|a^kb^k|$. Here $2k \geq n$

$w = a^kb^k$



$|xy| = |a^{k-1}a| = |a^k|$ (Here $k \leq 2k$)

$|y| \neq 0$

Hence, $xy^iz \in L$; $\forall\ i \geq 0$

For $i = 2$, $a^{k-1}a^2b^k = a^{k+1}b^k \notin L$

$\therefore$ L is Non-Regular.

Note:

Special Case of pumping Lemma.

When $\sum = \{a\}$ i.e. singleton set. Then lengths of string must follow arithmetic progression for a language to be a regular language.

Example 1 : $L = \{a^{2n+1} \mid n \geq 0\}$
$= \{a, aaa, aaaaa, ...\}$
Regular language. **(Following arithmetic progression)**

Example 2 : $L = \{a^{3n} \mid n \geq 1\}$
$= \{aaa, aaaaaa, ...\}$
Regular language. **(Following arithmetic progression)**

Example 3 : $L = \{a^{2^n} \mid n \geq 0\}$
$= \{a, aa, aaaa, ...\}$ **(Not following arithmetic progression)**
Not Regular language.

Which of the following languages are regular?

1) $L = \{a, ab, aba\} \rightarrow$ Finite set
It is Regular Language.

2) $L = \{w \in \{a, b\}^*, |w| = 20\} \rightarrow$ Regular Language

3) $L = \{w \in \{a, b\}^*, |w| = r \pmod n\} \rightarrow$ Regular Language

4) $L = \{w \in \{a, b\}^*, |w| \geq 20\} \rightarrow$ Regular Language

5) $L = \{a^mb^n \mid m, n \geq 0\} \rightarrow$ Regular Language

6) $L = \{a^mb^n \mid m \geq 2020, n \geq 2021\} \rightarrow$ Regular Language

7) $L = \{a^mb^n \mid 1 \leq m \leq 200, 20 \leq n \leq 400\} \rightarrow$ Regular Language

8) $L = \{a^mb^n \mid m + n = 20\} \rightarrow$ Finite $\rightarrow$ Regular Language

9) $L = \{a^mb^n \mid mn = \text{Finite}\} \rightarrow$ Regular Language

10) $L = \{a^mb^n \mid 1 \leq m \leq n \leq 40\} \rightarrow$ Regular Language

11) $L = \{a^nb^n \mid n \geq 1\} \rightarrow$ Non-Regular Language

12) $L = \{a^m b^n | \ m = n, m, n \geq 0\} \rightarrow$ Non-Regular Language
13) $L = \{a^m b^n | \ m < n\} \rightarrow$ Non-Regular Language
14) $L = \{a^m b^n | \ m \neq n, m, n \geq 0\} \rightarrow$ Non-Regular Language
15) $L = \{a^m b^n | \ m = n^2\} \rightarrow$ Non-Regular Language
16) $L = \{a^m b^n | \text{gcd}(m, n) = 1\} \rightarrow$ Non-Regular Language
17) $L = \{wcw^R | \ w \in (a, b)^*\} \rightarrow$ Non-Regular Language
18) $L = \{wxw^R | \ w, x \in (0, 1)^+\} \rightarrow$ Regular Language
19) $L = \{a^n b^n c^n | \ n \geq 1\} \rightarrow$ Non-Regular Language
20) $L = \{wxw^R | \ w, x \in (0, 1)^+, |x| = 5\} \rightarrow$ Non-Regular Language

Rack Your Brain

Is the language $L = \{a^n | \ n \text{ is a perfect square}\}$ Regular or Non-Regular?

Previous Years' Question

Which of the following language is/are Regular? **(GATE 2015 SET-2)**

L_1 : $\{wxw^R | \ w, x, \in \{a, b\}^*$ and $|w|, |x| > 0\}$, w^R is the reverse of string w.

L_2 : $\{a^n b^m | \ m \neq n$ and $m, n \geq 0\}$

L_3 : $\{a^p b^q c^r | \ p, q, r \geq 0\}$

1) L_1 and L_3 only
2) L_2 only
3) L_2 and L_3 only
4) L_3 only

Sol: 1)

Rack Your Brain

Find whether $L = \{a^p | \ p \text{ is a prime number}\}$ is Regular or Non-Regular?

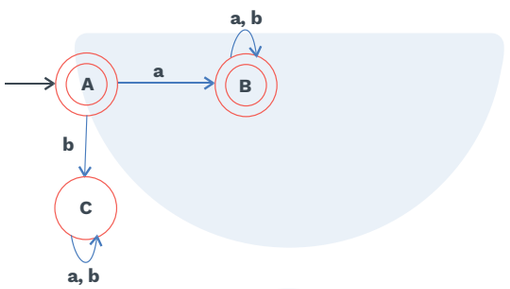
Myhill-nerode theorem

- Myhill-Nerode Theorem provides a necessary and sufficient condition for a language to be a regular.
- The Myhill-Nerode Theorem states that L is regular if and only if R_L has a finite number of equivalence classes and moreover that the number

of states in the minimal DFA recognizing L is equal to the number of equivalences classes in R_L.

- The language L defines over the alphabet '$\sum$' is Regular iff the number of equivalence classes with respect to 'L' is finite.
- In minimal finite automata, some language can be defined at each and every state. These languages are mutual disjoint and they are known as equivalence classes.
- Union of all these equivalence classes = $\sum^*$
- Union of some of the equivalence classes = Language 'L' accepted by finite automata.

Example 1:



$A = \{\in\}$
$B = a(a+b)^*$ } Equivalence Classes
$C = b(a+b)^*$

Union of all these equivalence classes = $A \cup B \cup C = \sum^*$

Union of some of the equivalence classes = $A \cup B$

= language accepted by finite automata

- The language 'L' is Regular if and only if 'L' is accepted by finite automata.

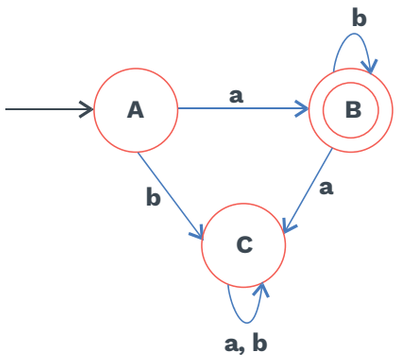
$\Leftrightarrow$ The number of states is finite.

$\Leftrightarrow$ The number of equivalence classes with respect to 'L' is finite.

- The language 'L' is not regular $\Leftrightarrow$ The number of equivalence classes with respect to 'L' is infinite.

Example 2: Let $L = ab^*$

The minimal DFA for the $L = ab^*$ is:



The equivalence classes are:

$A = \{\in\}$

$B = ab^*$ (State B refers to the strings in the language)

$C = b(a+b)^* + ab^*a(a+b)^*$

- L is regular (There are 3 equivalence class that and has a minimal DFA with 3 states).
- L itself is contained entirely in one equivalence class that corresponds to the fact that the minimal DFA has only one accepting state.

Chapter Summary

- **Regular expression:** Regular expression is one way to describe the regular languages via the notation.
- **Operators of regular expression:** Union, Concatenation, Closure (Kleene Closure/ Star Closure).
- **Order of precedence of regular expression operators:**

$$* > \bullet > +$$

- **Equivalence between regular language and regular expression:** For every regular language, there exists a regular expression and for every regular expression, there is a regular language.
- **Properties of regular expression:** Commutativity, Associativity, Distributive, Identity, Idempotent, Closure.
- **Equivalence between regular expression and finite automata:** Every language defined by a regular expression and a regular expression is defined by Finite Automata.
- **Conversion of finite automata to regular expression:** i) State Elimination Method and ii) Arden's Method.
- **Regular grammar:** Another way of describing the regular languages. It is also known as Type 3 grammar in Chomsky Hierarchy.
- The Grammar 'G' is said to be type 3 or Regular if every production is in the form: $A \rightarrow xB \mid x$ or $A \rightarrow Bx \mid x$, where $A, B \in V, x \in T^*$
- **Types of regular grammar:** Right linear Grammar and left linear Grammar.
- **Regular Language:** The language which is accepted by finite Automata, generated by Regular expression and generated by Regular Grammar is known as Regular language.
- **Myhill-nerode theorem:** "Myhill-Nerode Theorem provides a necessary and sufficient condition for a language to be a regular.