Module II
Regular Expression

Syllabus and Planner

Lecture	Topic	Book
1	Formal definition and Construction of Regular Expression of the given Language	T1,T2
2	Identities of Regular Expressions Construction of Regular Expression of the given Language	T1,T2
3	Construction of Language from the RE	T2
4	FA and RE, DFA to RE Using Arden's Theorem, Closure properties of RLs, Applications of Regular Expressions	T2,R3
5	RE to DFA (RE to e-NFA to DFA and RE to DFA Direct Method)	T2,R3
6	Pumping Lemma for RL , Interconversion between Left Linear and Right linear Grammar	T1,T2
7	Closure properties of RLs , Applications of Regular Expressions	T1,T2

Tutorial No	Topic
1	Construction of Regular Expression of the given Language, Construction of Language from the RE, DFA to RE conversion Using Arden's Theorem,

Text Books & Reference Books

- **Text Books**
  1. Michael Sipser "Introduction to the Theory of Computation" CENGAGE Learning, 3rd Edition ISBN-13:978-81-315-2529-6
  2. Vivek Kulkarni, "Theory of Computation", Oxford University Press, ISBN-13: 978-0-19-808458-7

- **Reference Books**
  1. Hopcroft Ulman, "Introduction To Automata Theory, Languages And Computations", Pearson Education Asia, 2nd Edition
  2. Daniel. A. Cohen, "Introduction to Computer Theory" Wiley-India, ISBN:978-81-265-1334-5
  3. K.L.P Mishra ,N. Chandrasekaran ,"Theory Of Computer Science (Automata, Languages and Computation)", Prentice Hall India,2nd Edition
  4. John C. Martin, "Introduction to Language and Theory of Computation", TMH, 3rd Edition ISBN: 978-0-07-066048-9
  5. Kavi Mahesh, "Theory of Computation: A Problem Solving Approach", Wiley-India, ISBN: 978-81-265-3311-4

What is Regular expression

- Regular expressions are short notations that can denote complex and infinite regular languages.

- In arithmetic, we can use the operations + and × to build up expressions such as

  $(5 + 3) \times 4$.
- The value of the arithmetic expression is the number 32.

- Similarly, we can use the regular operations to build up expressions describing languages, which are called **regular expressions**.

  Example : $\mathbf{(0 \cup 1)0^*}$ **.**

- The value of a regular expression is a language:

  language of consisting of all strings starting with a 0 or a 1 followed by zero or any number of 0s.

Definition of a Regular Expression

1. Regular expressions over **Σ**, include **letters, ∅(empty set) and ε(empty string of length zero).**

2. Every symbol **a € Σ** is a regular expression over Σ.

3. If R1 and R2 are regular expressions over Σ, then so are **(R1+R2), (R1.R2) and (R1)***

   [Where '+' indicates union,
   '.' indicates concatenation,
   '*' indicates closure or repetitive concatenation]

4. Regular expressions are only those that are obtained using rules1-3.

Operators of RE

Regular Expression operators:

1. The Star operator :Closure

   if r = a* then L(r) = (€,a,aa,aaa,aaaa,….)

2. The Dot operator :Concatenation

   if r = a.b then L(r) = (ab)

3. The Plus operator :Union

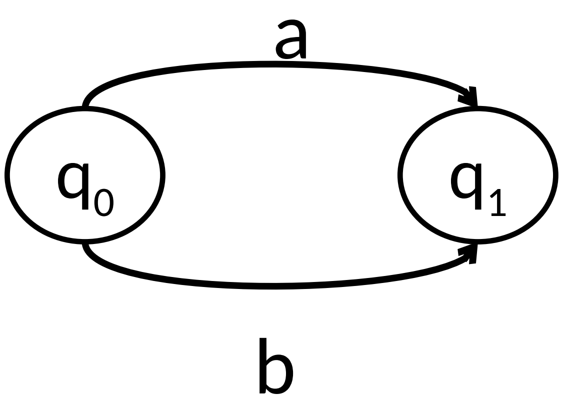
   if r = a+b then L(r) = (a,b)

Continued…

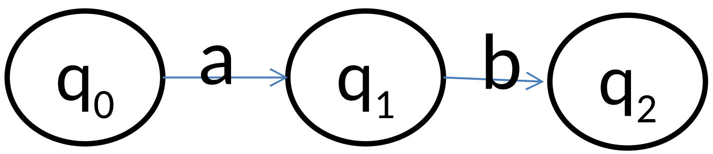
1. “a+b” stands for either a or b (parallel)

2. “a.b” stands for a followed by b (series)

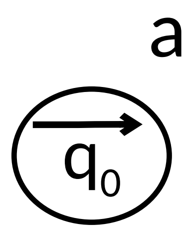
3. “a*” stands for any no of occurrences of a(Closure).



(1)



(2)



(3)

Continued…

Concatenation of 2 sets:

$U.V = \{x \mid x=uv,\ u \subseteq U \text{ and } v \subseteq V\}$

$UV \neq VU$

$U(VW)=(UV)W$

E.g. $U=\{000,111\}, V=\{101,010\}$

Closure of a set:

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

Where $S^0 = \{\epsilon\}$ and $S^i = S^{i-1}.S$ for $i>0$

e.g. $S=\{01,11\}$

$S^1 = S^0 .S = \{\epsilon\}.\{01,11\}$

$S^2 = S^1 .S = \{01,11\}\ \{01,11\}$

…..

$S^* = \{\epsilon\ , 01,11,0101,0111,1101,1111,\ldots\ldots\}$

Examples of RE

1. Using RE describe the language consisting if all strings over {0,1} with at least two consecutive 0's.
2. If L(r) =set of all strings over {0,1,2} such that at lease one 0 is followed by at least one 1,which is followed by at least one 2 ,find a RE r representing this language.
3. Using RE represent the language over {a , b} with all strings starting and ending with any number of b's in between.
4. If L(r) =set of all strings over {0,1 } ending with '011',then find r.
5. Describe the language represented by RE $r=(1 + 10)^*$
6. Represent the language over {0,1}containing all possible combinations of 0's and 1's but not having two consecutive 0's.
7. Show that $(a . b)^* \neq a^*b^*$

Precedence of Regular Expression operators

The Star operator :Closure
The Dot operator :Concatenation
The Plus operator :Union

E.g.
01*+1 is grouped as (0(1*))+1
Set of all strings over {0,1} consisting of 1 or a 0 followed by zero or more number of 1s.

(01)*+1
Set of all strings over {0,1} consisting of 1 or zero or more number of 01.

0(1*+1)
Set of all strings over {0,1} starting with 0 followed by single one or zero or more number of 1's.

Identities of regular expressions

1. **R U Φ=R**
   Adding the empty language to any other language will not change it.

2. **Φ.R = R.Φ = Φ**
   if R = 0, then L(R) = {0} but L(R ∘ ∅) = ∅.

3. **€.R =R.€ =R**
   Joining the empty string to any string will not change it

4. **€*= € and Φ*= €**
5. **R + R = R**
6. **R*R *= R***
7. **RR* = R*R**
8. **(R*)=R***
9. **€ +RR *= R*= € +R*R**
10. **(PQ)*P = P(QP)***
11. **(P + Q)* = (P*Q*)* = (P*+Q*)***
12. **(P+Q)R=PR+QR and R(P+Q)=RP+RQ**

Examples

True or False?

Let R and S be two regular expressions. Then:

1. $((R^*)^*)^* = R^*$?
2. $(R+S)^* = R^* + S^*$?

Construction of RE from regular Language

Write RE to represent L over $\sum^*$ where $\sum=\{0,1\}$:

1.Set of all strings of 0's and 1's over {0,1}

Ans: $\sum=\{0,1\}$,

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

$R=(0 + 1)^*$

2. Set of all strings in which 0 is followed by any number 1's

Ans: $\sum=\{0,1\}$,

$L=\{0,01,011,0111,\ldots.\}$

$R=01^*$

3. $L=\{01, 10\}$

$R=01 \cup 10$

4. The language of all binary strings over $\sum=\{0,1\}$ starting with 01

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

5. The language of all binary strings ending at 01

$R=(0+1)^*01.$

6. The language of all binary strings having substring 01

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

Continued…

7. If L(r) ={aaa,aab,aba,abb,baa,bab,bba,bbb} find r.
   R=(a+b)(a+b)(a+b)

8. If L(r) ={a,c,ab,cb,abb,cbb,abbb….} find r.
   R=(a+c).b*

9. If L(r) ={€,x,xx,xxx,xxxx,xxxxx} find r.
   R=(€+x) (€+x) (€+x) (€+x) (€+x)

10. The set of all strings of 0's and 1's such that the tenth symbol from the right is 1.
    R=$(0+1)^*.1.(0+1)^9$

Construction of Regular Language from RE

Describe the language represented by following R.E.

1) r=(0|1)*011

Ans: ∑={0,1},

L{r}=Set of all strings over {0,1} such that all strings end with 011

2) r= 0*1*2*

Ans: ∑={0,1,2},

L{r}=Set of strings over {0,1,2} with zero or more number of 0's,
followed by zero or more number of 1's,
followed by zero or more number of 2's

3) r= 00*11*22*

Ans:∑={0,1,2},

L{r}=Set of all strings over {0,1,2}such that every string will have at least one 0 followed by at least one 1 followed by at least one 2.

4) r= (0*1*)*000(0+1)*

L=Set of all strings over {0,1} with 3 consecutive 0's.

5)r= (0 ∪ ε)(1 ∪ ε)

L= {ε, 0, 1, 01}

Construction of Regular Language from RE

6) R= (∑∑)*

L={w| w∈∑ and is a string of even length}

7) R= (∑∑∑)*

L={w| w∈∑ and length of w is a multiple of 3}

8) Φ*= ?

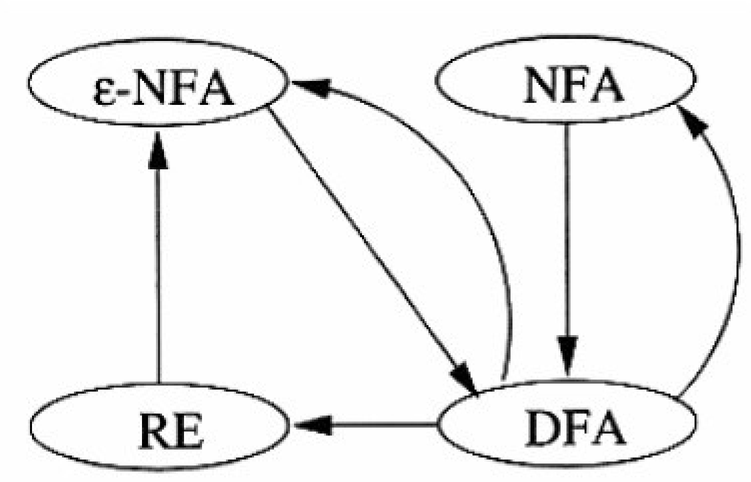
L= €

9) R . € =?

L= R

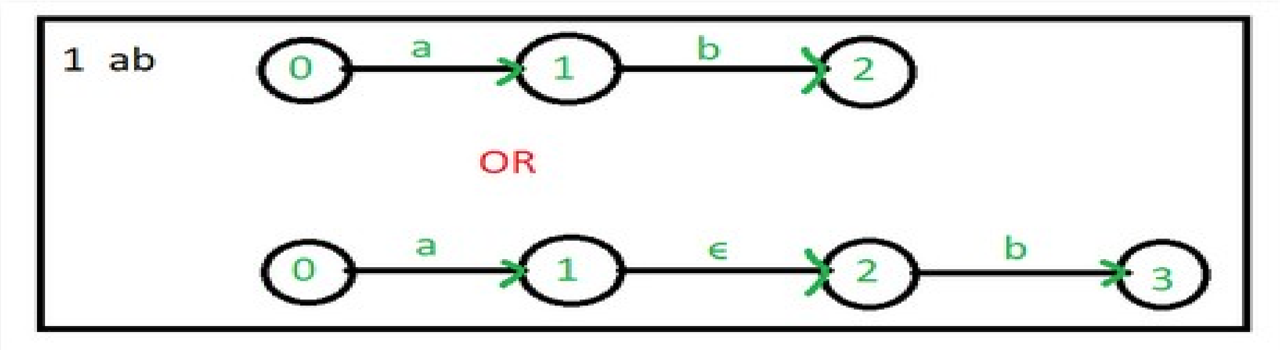
Equivalence of FA and RE

- Finite automata (DFA and NFA) and regular languages are equivalent.
- Every regular language can be recognized (accepted) by a finite automata and,
- conversely, for every finite automata there is a regular language which is the language accepted by the finite automata.

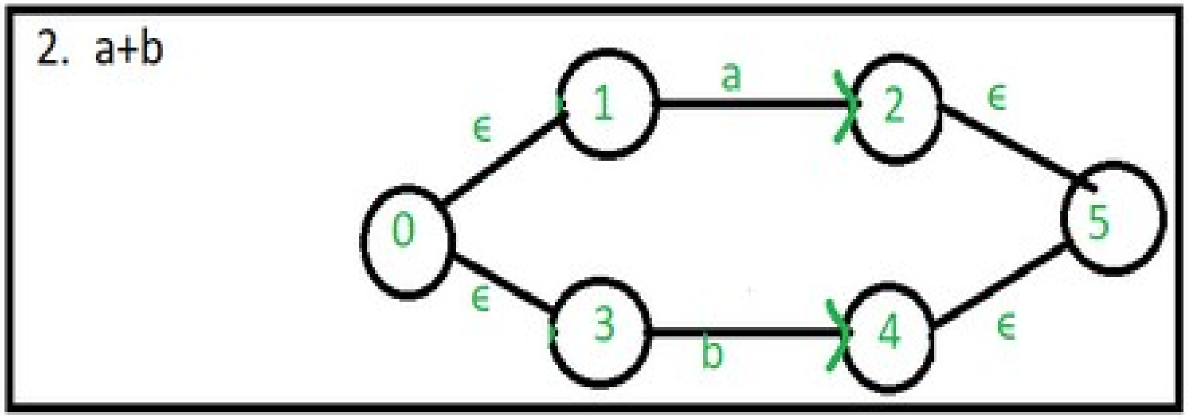


RE to NFA

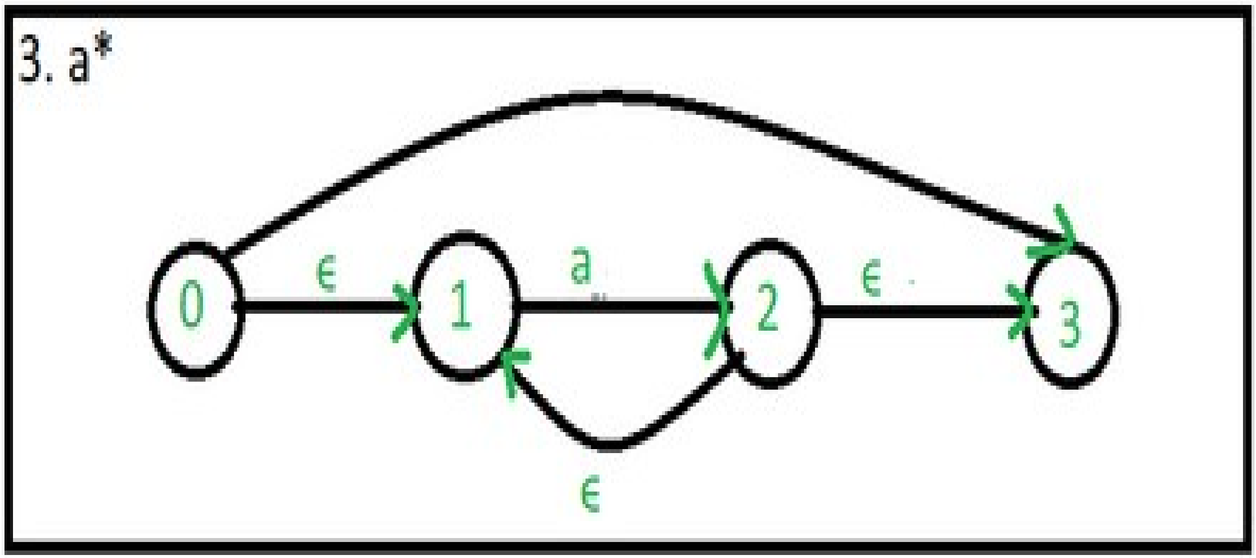
1) a.b



2) a+b



3) a*



Example

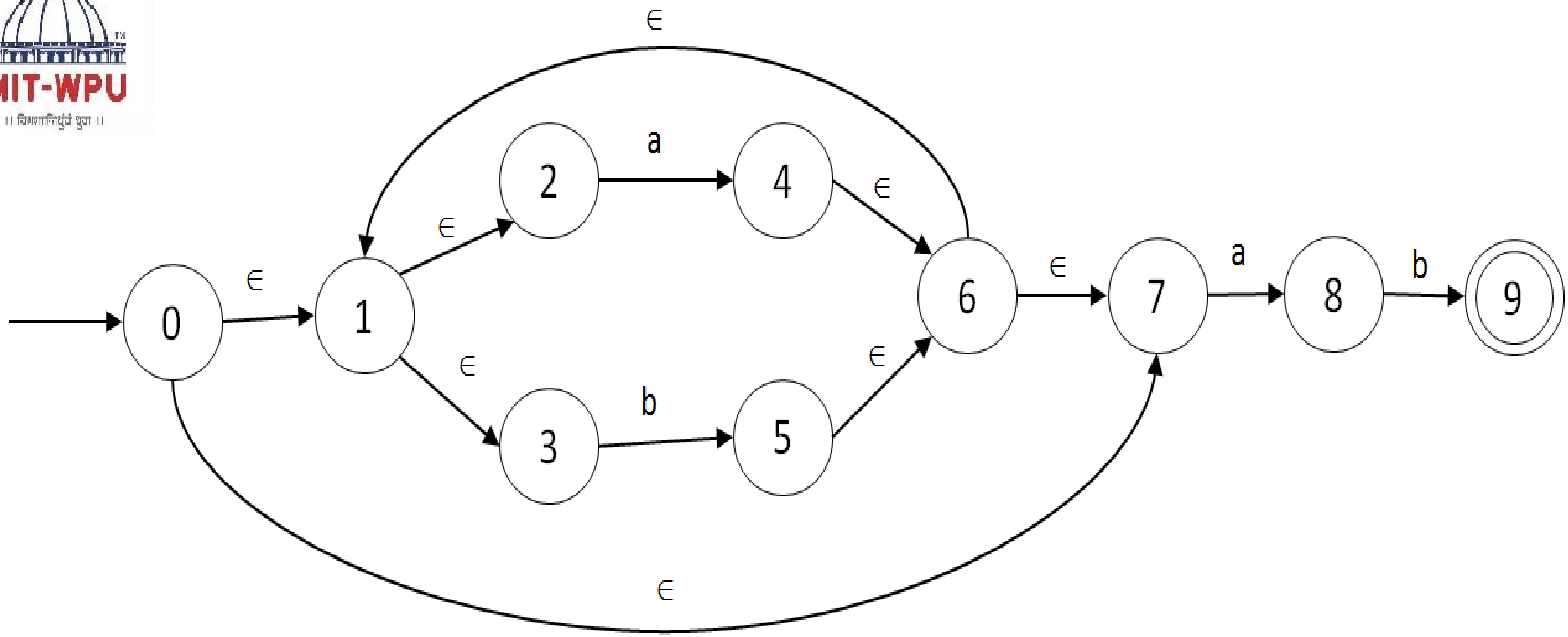
Construct an NFA for the regular expression, (a + b)* ab. Convert the NFA to its equivalent DFA .

Solution:

It is expected to construct a DFA that recognizes the regular set: $R = (a+b)\cdot a \cdot b$

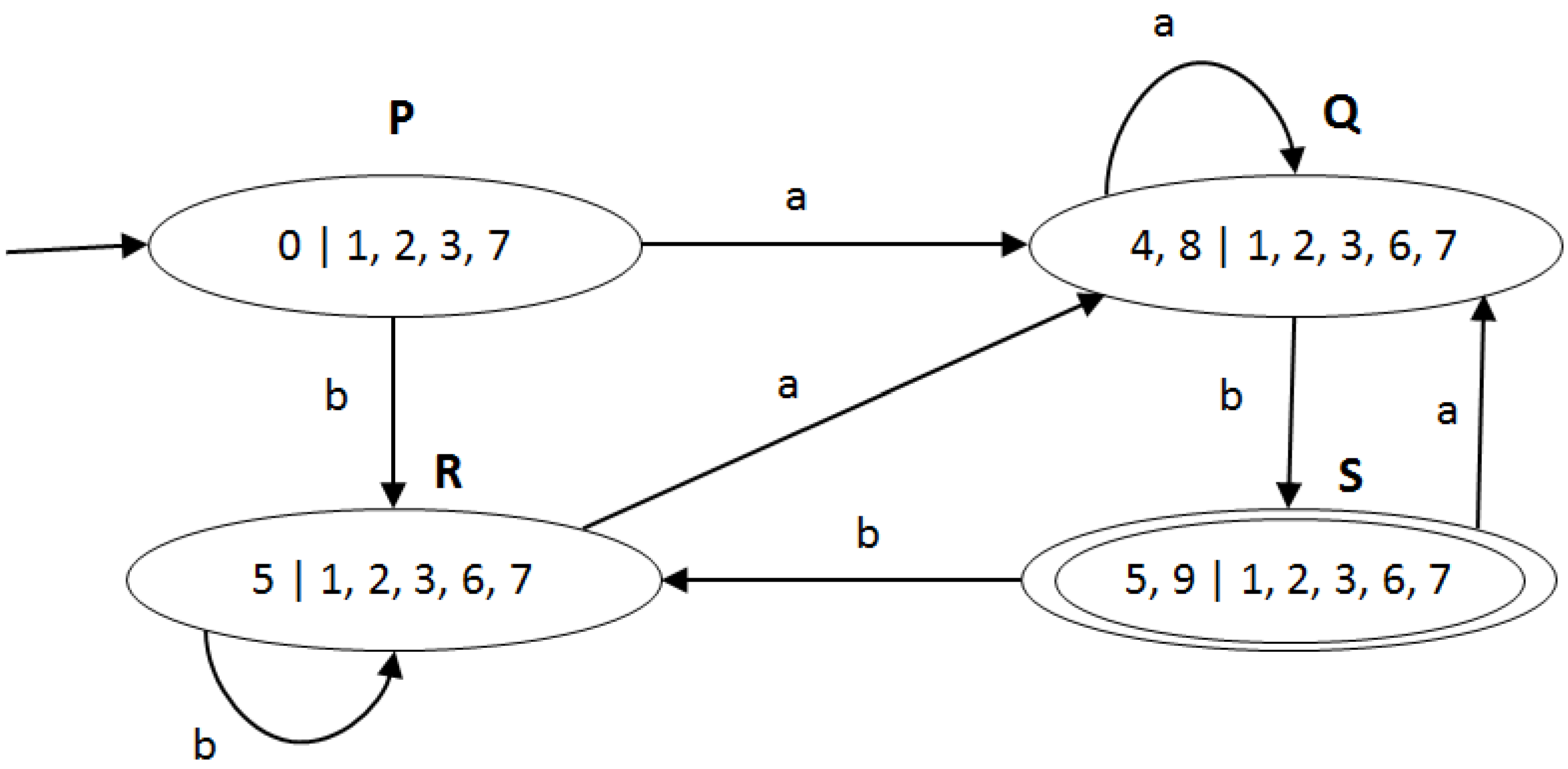
Let us first build the NFA with **ε** moves and the convert the same to DFA.

The TG for NFA with **ε** moves is as follows,



Let us convert this NFA with ε -moves to its equivalent DFA using a direct method.

RE to NFA



We have relabelled the states as well.
Let us see if we can minimize it.

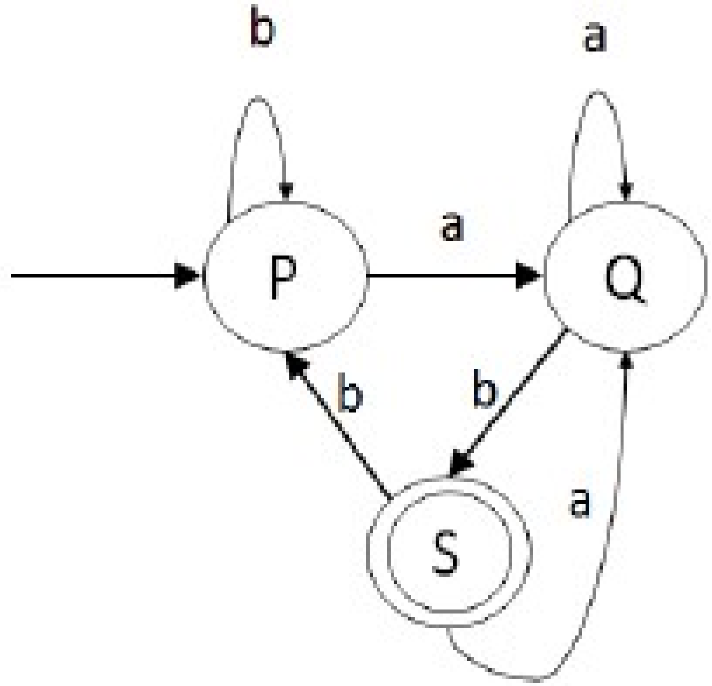
The STF for the DFA looks like,

Q\∑	a	b
P	Q	R
Q	Q	S
R	Q	R
* S	Q	R

We can see that states *P* and *R* are equivalent. Hence, we can replace *R* by *P* and get rid of *R*. The reduced STF is,

Q\∑	a	b
P	Q	P
Q	Q	S
* S	Q	P

The TG for the equivalent DFA is,



RE to FA

1. $R=(a+b).(a+b)^*$

2. $R=a.(a+b)^*$

Statement –

Let **P** and **Q** be two regular expressions.
If **P** does not contain null string(ε),
then **R = Q + RP** has a unique solution that is **R = QP***

Proof –

$R = Q + (Q + RP)P$ [After putting the value R = Q + RP]
$= Q + QP + RPP$

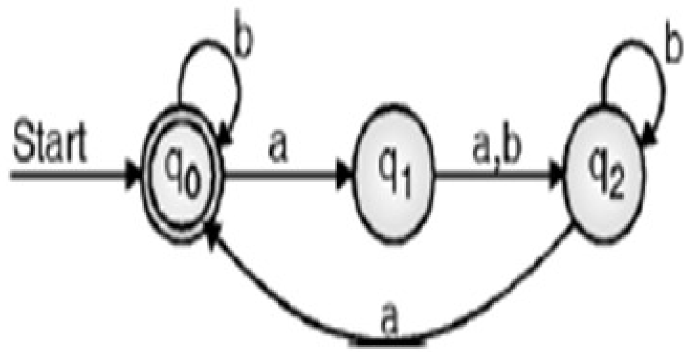
When we put the value of **R** recursively again and again, we get the following equation –

$R = Q + QP + QP^2 + QP^3\ldots..$
$R = Q\ (\varepsilon + P + P^2 + P^3 + \ldots.\ )$
$R = QP^*$ [As P* represents (ε + P + P2 + P3 + ….)]

Hence, proved.

Arden's Theorem Example 1

Construct a regular expression for the following DFA:



Solution:

The state equations for the given DFA are:

q0 = q0 b + q2 a + **ε**

q1 = q0 a

q2 = q1 a + q1 b + q2 b

Arden's Theorem

$q2 = q1\ a + q1\ b + q2\ b$

Substituting for q1 in q2,

$q2 = q0aa + q0ab + q2\ b$

$q2 = q0\ a\ (a + b) + q2\ b$

$q2 = q0\ a\ (a + b)b^*$ ... using Arden's Theorem(**R = Q + RP** has a unique solution that is **R = QP***)

Substituting for q2 in q0,

$q0 = q0\ b + q2\ a + \varepsilon$

$q0 = q0\ b + q0\ a\ (a + b)b^*\ a + \varepsilon$

$q0 = q0\ (b + a\ (a + b)b^*\ a\ ) + \varepsilon$

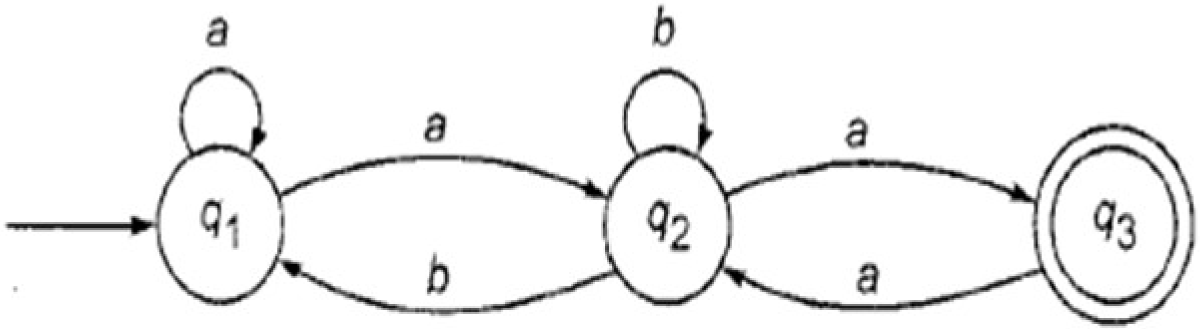
$q0 = \varepsilon\ (b + a\ (a + b)b^*\ a\ )^*$ ... using Arden's Theorem

Hence, $q0 = (b + a\ (a + b)b^*\ a\ )^*$

q0 being the only final state for the DFA,

$R = (b + a\ (a + b)b^*\ a\ )^*$

Example 2.



Construct RE for the given FA.

q1=q1.a+q2.b+є
q2=q1.a+q2.b+q3.a
q3=q2.a

Using q3in q2 (q3=q2.a)
q2 =q1.a+q2.b+q3.a
q2=q1.a+q2.b+q2.a.a
q2=q1.a+q2(b+aa) {**R = Q + RP** has a unique solution that is **R = QP*)**
Applying Arden's Theorem,
q2=q1.a(b+aa)*

Using q2 in q1 (q2=q1.a(b+aa)*)
q1=q1.a+q2.b+ϵ
q1=q1.a+q1.a(b+aa)*.b+ ϵ
q1= ϵ+q1[a+a(b+aa)*.b] {**R = Q + RP** has a unique solution that is **R = QP*)**
Applying Arden's Theorem,
q1= ϵ. [a+a(b+aa)*.b]*
q1= [a+a(b+aa)*.b]*

q2= [a+a(b+aa)*.b]*.a(b+aa)* {q2=q1.a(b+aa)*}
q3= [a+a(b+aa)*.b]*.a(b+aa)*.a {q3=q2.a}
Is the RE for given FA

Limitations of RE

1. FA does not have the capacity to remember large amount of information.

2. The head can not move in reverse direction.

3. It cannot recognize the languages which are not regular. e.g. Palindrome.

4. FA cannot multiply the numbers.

Nonregular languages

Consider the language $B = \{0^n1^n | n \geq 0\}$.

- If we attempt to find a DFA that recognizes B we discover that such a machine needs to remember how many 0s have been seen so far as it reads the input
- Because the number of 0s isn't limited, the machine needs to keep track of an unlimited number of possibilities
- This cannot be done with any finite number of states

Intuition may fail us

- Just because a language appears to require unbounded memory to be recognized, it doesn't mean that it is necessarily so
- Example:
  - $C = \{w \mid w$ has an equal number of 0s and 1s$\}$ not regular
  - $D = \{w \mid w$ has equal no of 01 and 10 substrings$\}$ regular

Language nonregularity

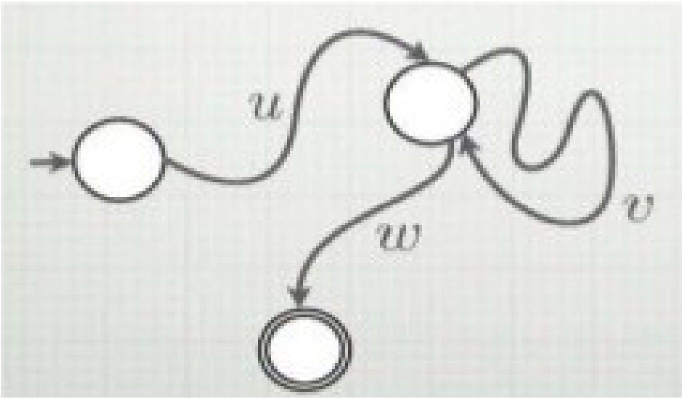
- The technique for proving nonregularity of some language is provided by a theorem about regular languages called pumping lemma
- Pumping lemma states that all regular languages have a special property
- If we can show that a language L does not have this property we are guaranteed that L is not regular.

$$\begin{array}{rcl} L \text{ regular} & \Longrightarrow & L \text{ satisfies P.L.} \\ L \text{ non-regular} & \Longrightarrow & ? \\ L \text{ non-regular} & \Longleftarrow & L \text{ doesn't satisfy P.L.} \end{array}$$

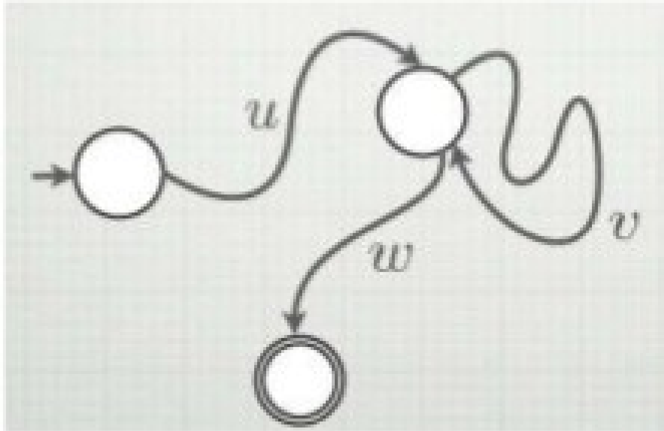
Pumping property

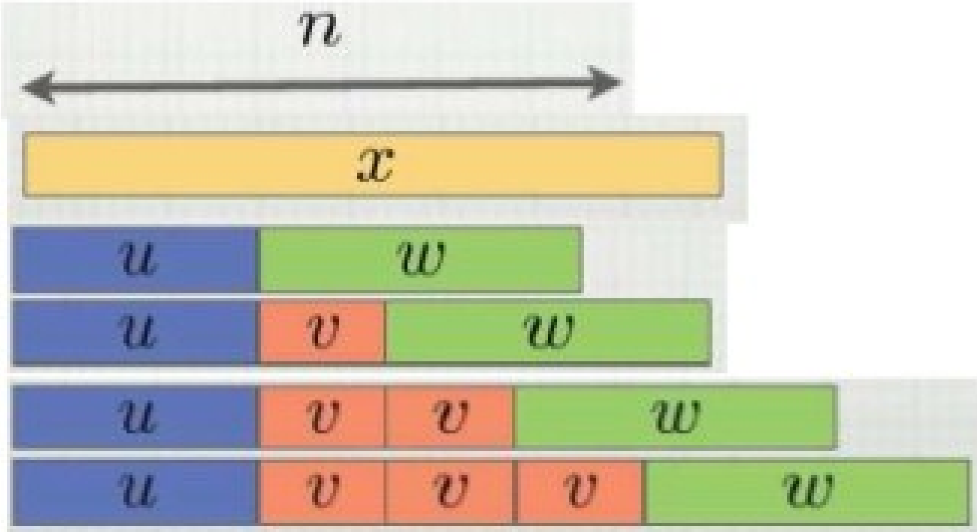
All strings in the language can be "pumped" if they are at least as long as a certain value, called the pumping length

Meaning: each such string in the language contains a section that can be repeated any number of times with the resulting string remaining in the language.



$$x = \begin{array}{l} uw \in L \\ uvw \in L \\ uvvw \in L \\ uvvvw \in L \\ \ldots \end{array}$$





$$
\begin{aligned}
& uw \in L \\
x = \; & uvw \in L \\
& uvvw \in L \\
& uvvvw \in L \\
& \dots
\end{aligned}
$$

Pumping lemma for Regular Languages

is used to prove that a language is not Regular

Let L be a regular language and $M(Q,\Sigma,\delta,q_1,F)$ is the finite automata with n states.

Let L is accepted by M.

Let $x \in L$ and $|x| \geq n$, then x can be written as uvw ,where

(i) $|v| > 0$

(ii) $|uv| \leq n$

(iii) $uv^iw \in L$ for all $i \geq 0$, where v^i denotes that v is repeated or pumped i times .

Pumping lemma Application

Example: Show that the given language

L={ $a^m b^m$ } is not regular

Ex 1. Show that the language L= $\{a^m b^m\}$ is not regular.

Answer:

Step 1:

Let us assume that L is regular and L is accepted by a FA with n states.

Step 2: Let us choose a string $x = a^m b^m$

$|x| = 2m >= n$

Let us write x as uvw, with $|v| > 0$ and $|uv| <= n$

Since, $|uv| <= n$, u must be the form of a^s.

Since, $|uv| <= n$, v must be of the form $a^r \mid r > 0$

Now, $a^m b^m$ can be written as $a^s\, a^r\, a^{m-s-r}\, b^m$

a^s is u , a^r is v , $a^{m-s-r} b^m$ is w

Step 3: Let us check whether $uv^i w$ for i=2 belongs to L.

$uv^2w = a^s\, (a^r)^2\, a^{m-s-r}\, b^m = a^s\, a^{2r}\, a^{m-s-r}\, b^m = a^{s+2r+m-s-r} b^m = a^{m+r}\, b^m$

Since r>0 , Number of a's in $a^{m+r} b^m$ is greater than number of b's .

Therefore, $uv^2w \not\subset$ L.

Hence by the contradiction we can say that the given language is not regular.

Q. Show that the language L= {xx|xϵ(a,b)*} is not regular.

Answer:

Step 1:

Let us assume that L is regular and L is accepted by a FA with n states.

Step 2: Let us choose a string $x= a^m b$, $xx= a^m b.a^m b$

$| xx | = m+1+m+1=2m+2 >= n$

Let us write x as uvw, with $|v| > 0$ and $|uv|<=n$

Since, $|uv| <=n$, u must be of the form a^s

Since, $|uv|<=n$, v must be the form of $a^r \mid r>0$

Now, $xx=a^m b.a^m b$ can be written as $a^s\, a^r\, a^{m-s-r} b.a^m b$

a^s is u , a^r is v , $a^{m-s-r} b.a^m b$ is w

Step 3: Let us check whether $uv^i w$ for i=2 belongs to L.

$uv^2 w = a^s\, (a^r)^2\, a^{m-s-r}\, b.a^m b = a^s\, a^{2r}\, a^{m-s-r}\, b.a^m b$

$= a^{s+2r+m-s-r} b.a^m b = a^{m+r}\, b.a^m b \neq L$

Therefore, $uv^2 w \not\subset L$.

Hence by the contradiction we can say that the given language is not regular.

Closure properties of Regular Languages

Regular Grammar : A grammar is regular if it has rules of form A -> a or A -> aB or A -> ε where ε is a special symbol called NULL.

Regular Languages : A language is regular if it can be expressed in terms of regular expression.

Closure properties of Regular Languages

Union : If L1 and If L2 are two regular languages, their union L1 ∪ L2 will also be regular.

For example,

$L1 = \{a^n \mid n \geq 0\}$ and $L2 = \{b^n \mid n \geq 0\}$

$L3 = L1 \cup L2 = \{a^n \cup b^n \mid n \geq 0\}$ is also regular.

Intersection : If L1 and If L2 are two regular languages, their intersection L1 ∩ L2 will also be regular.

For example,

$L1= \{a^m b^n \mid n \geq 0 \text{ and } m \geq 0\}$ and $L2= \{a^m b^n \cup b^n a^m \mid n \geq 0 \text{ and } m \geq 0\}$

$L3 = L1 \cap L2 = \{a^m b^n \mid n \geq 0 \text{ and } m \geq 0\}$ is also regular.

Closure properties of Regular Languages

Concatenation : If L1 and If L2 are two regular languages, their concatenation L1.L2 will also be regular.

For example,
$L1 = \{a^n \mid n \geq 0\}$ and $L2 = \{b^n \mid n \geq 0\}$
$L3 = L1.L2 = \{a^m . b^n \mid m \geq 0$ and $n \geq 0\}$ is also regular.

Kleene Closure : If L1 is a regular language, its Kleene closure L1* will also be regular.

For example,
$L1 = (a \cup b)$
$L1^* = (a \cup b)^*$

Closure properties of Regular Languages

Complement

If L(G) is regular language, its complement L'(G) will also be regular. Complement of a language can be found by subtracting strings which are in L(G) from all possible strings.

For example,

$L(G) = \{a^n \mid n > 3\}$

$L'(G) = \{a^n \mid n <= 3\}$

Application of Regular Expression

- Application in Linux

  Example : Text File Search , Unix tool: egrep

  Editing Commands , cw :Change word

- Application in Search Engine

- Web application

- Regular Expressions in Lexical Analysis

  Example : C programming language