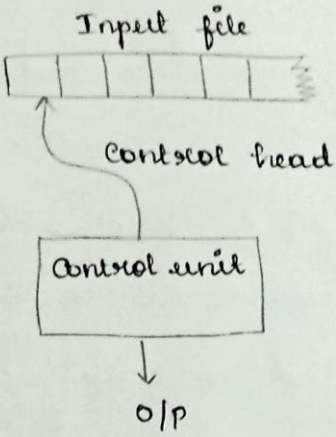
Finite Automata

Finite Automata involves states and transition among states in response to inputs. Finite Automata is a mathematical model of a system with discrete inputs and outputs. The system can be in any one of finite number of states and the state summarizes the history of past inputes and determines the behavior of the system for subsequent input.

It receives input as a string from an input file and it gives as output whether the input string is acceptable or not.

A finite automaton has a mechanism to read input, which is a string over a given alphabet. This input is actually written on an "input file", which can be read by the automaton but cannot change it.

Automaton :



(Accepted / Not accepted)

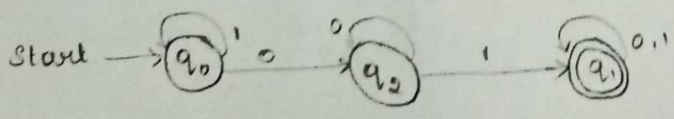
Types:

It can be of 3 types as follows,

1. Deterministic Finite Automata (DFA)
2. Non-Deterministic Finite Automata (NFA)
3. Non-Deterministic Finite Automata with ε (ε-NFA)

Problem :

Consider the DFA accepting all strings with a substring 01.



Solution :

The specification of the DFA accepts the language L of strings that have 01 substring

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

where δ is the transition function described, the table below.

Transition table :

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

$\delta(q_0, 0) = q_2$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_1$

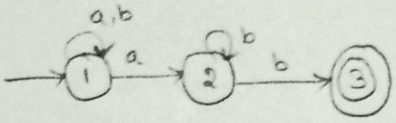
$\delta(q_1, 1) = q_1$

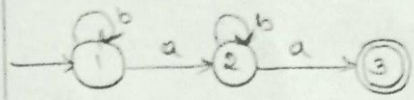
$\delta(q_2, 0) = q_2$

$\delta(q_2, 0) = q_1$

Compare NFA and DFA :

	NFA	DFA
1.	The automata is which for any input symbol from a given state those are many possible transitions	The automata in which for each input symbol from a given state there is an exactly one transition.





2.	It maps $Q \times \Sigma \rightarrow 2^Q$	It maps $Q \times \Sigma \rightarrow Q$
3.	Time complexity: The time needed for executing an input string is more.	less
4.	Supremacy: All NFAs are DFAs.	All DFAs are not NFAs.
5.	The language of an NFA. If $A = (Q, \Sigma, \delta, q_0, F)$ is an NFA, then $L(A) = \{w \mid \delta(q_0, w) \cap F \neq \phi\}$ i.e) $L(A)$ is the set of strings w in Σ^* such that $\delta(q_0, w)$ contains atleast one accepting state.	The language of a DFA. If $A = (Q, \Sigma, \delta, q_0, F)$ is a DFA then, $L(A) = \{w \mid \delta(q_0, w)$ is in $F\}$ i.e) the language of A is the set of strings w that then start state q_0 to one of the accepting states.

Regular expressions and languages:

Definition:

The rules that define the regular expressions over alphabet Σ are as follows:

1) ϕ is a regular expression that denotes the null set.

2) ϵ is a regular expression that denotes empty set $\{\}$ that is the set containing the empty string.

3) If 'a' is a symbol present in Σ, then 'a' is a regular expression that denotes $\{a\}$ that is set containing the string "a".

4) If 'r' and 's' are the regular expressions denoting the regular languages $L(r)$ and $L(s)$ then

   i) $r+s$ denoting $L(r) \cup L(s)$ union

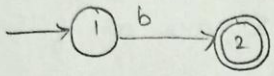
   rs denoting $L(r).L(s)$ concatenation

   r^* denoting $(L(r))^*$ closure

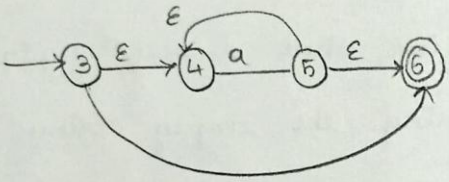
Construction of NFA - ε from regular expression

1) $b + ba^*$

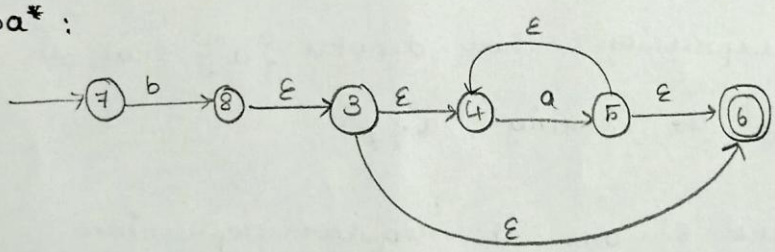
b :



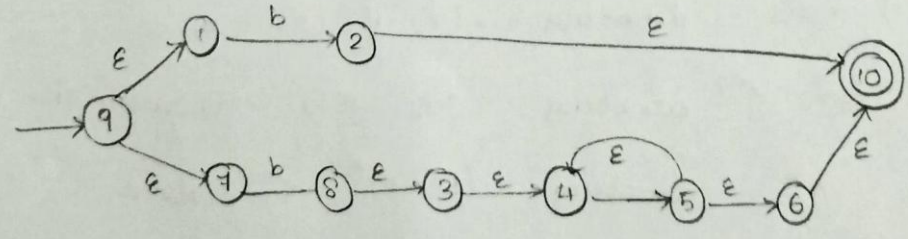
a^* :



ba^* :



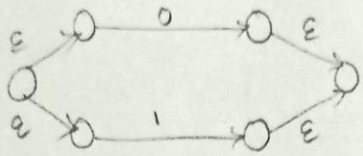
$b + ba^*$:



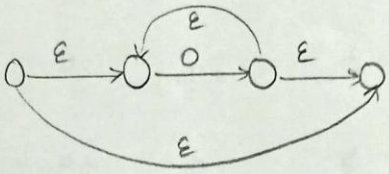
Regular Expressions (RE):

↳ set of strings

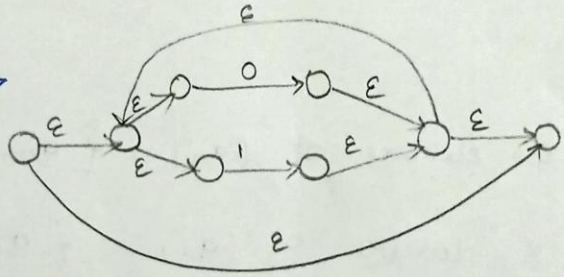
0 + 1 ⇒



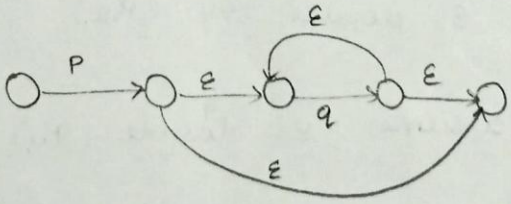
0^* ⇒



$(0+1)^*$ ⇒



pq^* ⇒



Conversion of NFA - ε to DFA

1) Convert NFA - ε to DFA

Solution :

Step 1 :

Transition table for NFA - ε

state	0	1	2	ε
q_0	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_0\}$	ϕ

Step 2 :

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

ε - closure of $(q_1) = \{q_1, q_2\}$

ε - closure of $(q_2) = \{q_2\}$

Let us assume ε - closure (q_0) is (A)

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

$\delta(A,0)$ = ε - closure $(\delta(A,0))$

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

= ε - closure $(\delta(q_0,0) \cup (q_1,0), \cup (q_2,0))$

= ε - closure (q_0)

$= (q_0, q_1, q_2) \longrightarrow (A)$

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

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

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

$= (q_1, q_2) \longrightarrow (B)$

$\delta(A,2) = \varepsilon\text{-closure}(\delta(q_0, q_1, q_2\}, 1)$

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

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

$= q_0 \longrightarrow (C)$

$\delta(B,0) = \varepsilon\text{-closure of}(\delta\{q_1, q_2\}, 0)$

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

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

$= \phi$

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

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

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

$= (q_1, q_2) \longrightarrow (B)$

$\delta(B,2) = \varepsilon\text{-closure}\ (\delta(q_1,2) \cup (q_2,2))$

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

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

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

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

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

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

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

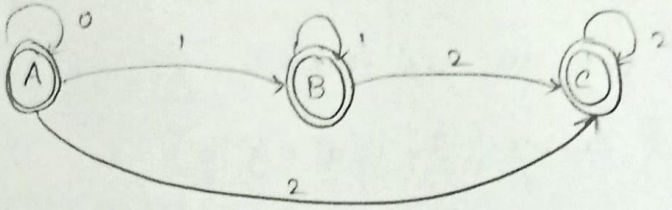
Step 3 :

Transition table for DFA

State	0	1	2
A	A	B	C
B	ϕ	B	C
C	ϕ	ϕ	C

Step 4:

Transition diagram for DFA

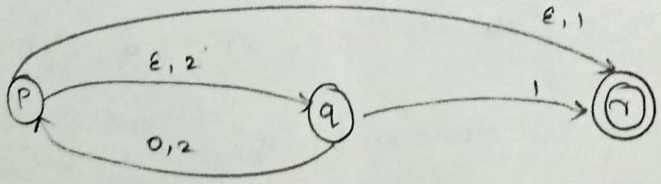


2) Given NFA - ε transition table to find the equivalent DFA.

State	ε	0	1	2
p	{q, r}	ϕ	{r}	{q}
q	ϕ	{p}	{r}	{p, q}
r	ϕ	ϕ	ϕ	ϕ

Solution:

Step 1: Transition diagram for NFA - ε



Step 2:

ε - closure of state:

ε - closure of p = {p, q, r}

ε - closure of q = {q}

ε - closure of r = {r}

Let's assume ε - closure of (P) as (A)

$A = \{p, q, r\}$

Transition for all the states :

$\delta(A,0) = \varepsilon\text{-closure}(\delta\{p,q,r\},0)$

$= \varepsilon\text{-closure}(\delta(p,0) \cup (q,0) \cup (r,0))$

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

$= \{p,q,r\} \longrightarrow (A)$

$\delta(A,1) = \varepsilon\text{-closure}(\delta\{p,q,r\},1)$

$= \varepsilon\text{-closure}(\delta(p,1) \cup (q,1) \cup (r,1))$

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

$= \{r\} \longrightarrow (B)$

$\delta(A,2) = \varepsilon\text{-closure}(\delta\{p,q,r\},2)$

$= \varepsilon\text{-closure}(\delta(p,2) \cup (q,2) \cup (r,2))$

$= \varepsilon\text{-closure}(p,q)$

$= \{p,q,r\} \longrightarrow (A)$

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

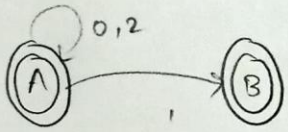
$\delta(B,1) = \varepsilon\text{-closure}(\delta(r,1)) = \phi$

$\delta(B,2) = \varepsilon\text{-closure}((r,2)) = \phi$

Transition table for DFA

State	0	1	2
A	A	B	A
B	ϕ	ϕ	ϕ
C	ϕ	ϕ	ϕ

Transition diagram for DFA



RE to DFA:

Note: i) R.E → NFA-ε

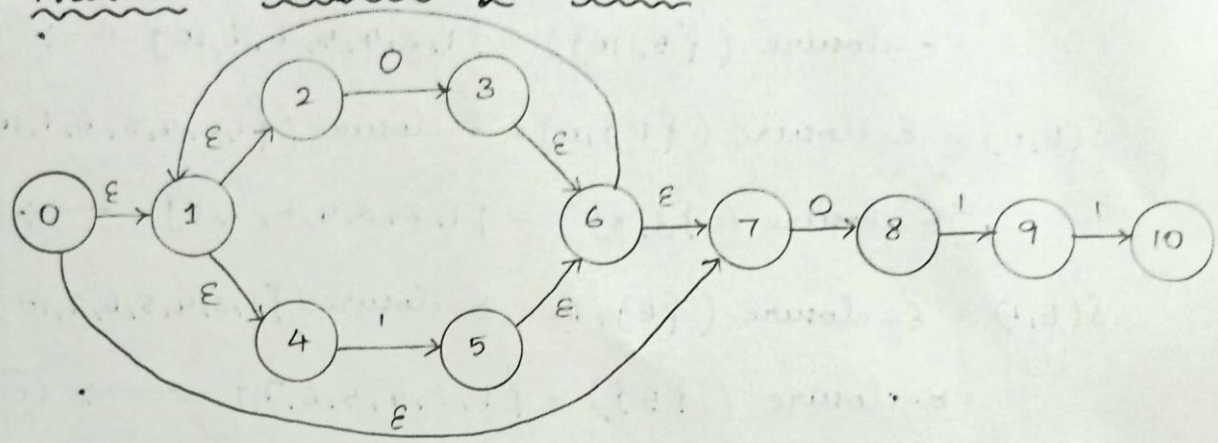
ii) NFA-ε → DFA

Ex:

Convert DFA from the following R.E $(0/1)^*011$.

Sol:

Transition diagram for NFA-ε:



Let us assume initial state {0}

ε-closure {0} = {0,1,2,4,7} → (A)

$\delta(A,0)$ = ε-closure ({A},0) = ε-closure({0,1,2,4,7},0)

= ε-closure ({3,8}) = {1,2,3,4,6,7,8} → (B)

$\delta(A,1)$ = ε-closure ({A},1) = ε-closure ({0,1,2,4,7},1)

= ε-closure {5} = {1,2,4,5,6,7} → (C)

$\delta(B,0)$ = ε-closure ({B},0) = ε-closure ({1,2,3,4,6,7,8},0)

= ε-closure ({3,8}) = {1,2,3,4,6,7,8} → (B)

$\delta(B,1)$ = ε-closure ({B},1) = ε-closure ({1,2,3,4,6,7,8},1)

= ε-closure ({5,9}) = {1,2,4,5,6,7,9} → (D)

$\delta(C,0)$ = ε-closure ({C},0) = ε-closure ({1,2,4,5,6,7},0)

= ε-closure ({3,8}) = {1,2,3,4,6,7,8} → (B)

15

$\delta(C,1) = \varepsilon\text{-closure}(\{C\},1) = \varepsilon\text{-closure}(\{1,2,4,5,6,7\},1)$

$= \varepsilon\text{-closure}(\{5\}) = \{1,2,4,5,6,7\} \longrightarrow (C)$

$\delta(D,0) = \varepsilon\text{-closure}(\{D\},0) = \varepsilon\text{-closure}(\{1,2,4,5,6,7,9\},0)$

$= \varepsilon\text{-closure}(\{3,8\}) = \{1,2,3,4,6,7,8\} \longrightarrow (B)$

$\delta(D,1) = \varepsilon\text{-closure}(\{D\},1) = \varepsilon\text{-closure}(\{1,2,4,5,6,7,9\},1)$

$= \varepsilon\text{-closure}(\{5,10\}) = \{1,2,4,5,6,7,10\} \longrightarrow (E)$

$\delta(E,0) = \varepsilon\text{-closure}(\{E\},0) = \varepsilon\text{-closure}(\{1,2,4,5,6,7,10\},0)$

$= \varepsilon\text{-closure}(\{3,8\}) = \{1,2,3,4,6,7,8\} \longrightarrow (B)$

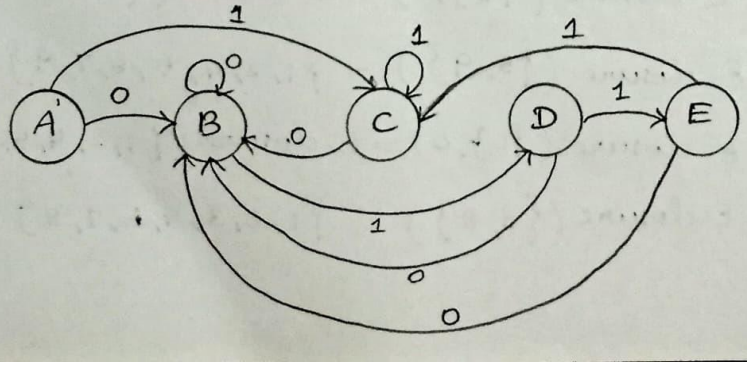
$\delta(E,1) = \varepsilon\text{-closure}(\{E\},1) = \varepsilon\text{-closure}(\{1,2,4,5,6,7,10\},1)$

$= \varepsilon\text{-closure}(\{5\}) = \{1,2,4,5,6,7\} \longrightarrow (C)$

Transition Table for DFA:

State	0	1
A	B	C
B	B	D
C	B	C
D	B	E
*E	B	C

Transition Diagram for DFA:



Nondeterministic Finite Automata : (NFA)

A nondeterministic finite automation NFA has the power to be in several states at once.

Definition:

An NFA, M is represented as,

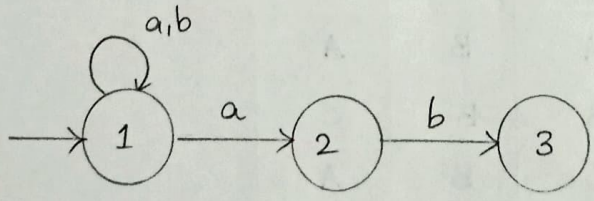
$$M = (Q, \Sigma, \delta, q_0, F)$$

where

1) Q is a finite set of states
2) Σ is a finite set of i/p symbols.
3) q_0 is a member of Q, is the start state
4) F a subset of Q, is the set of final or accepting states.
5) δ, the transition function of the form $Q \times \Sigma \to 2^Q$.

Ex:1:

8 mark **How to convert NFA to DFA:**



Given: NFA

Let us assume initial state of DFA is {1}

— (A) of DFA.

$\delta(A,a) = \delta(\{1\},a) = \{1,2\}$ — (B)

$\delta(A,b) = \delta(\{1\},b) = \{1\}$ — (A)

$\delta(B,a) = \delta(\{1,2\},a)$

$= \delta(1,a) \cup \delta(2,a)$

$= \{1,2\} \cup \phi = \{1,2\}$ — (B)

$\delta(B,b) = \delta(\{1,2\},b)$

$= \delta(1,b) \cup \delta(2,b)$

$= \{1\} \cup \{3\} = \{1,3\}$ — (C)

$\delta(C,a) = \delta(\{1,3\},a)$

$= \delta(1,a) \cup \delta(3,a)$

$= \{1,2\} \cup \phi = \{1,2\}$ — (B)

$\delta(C,b) = \delta(\{1,3\},b)$

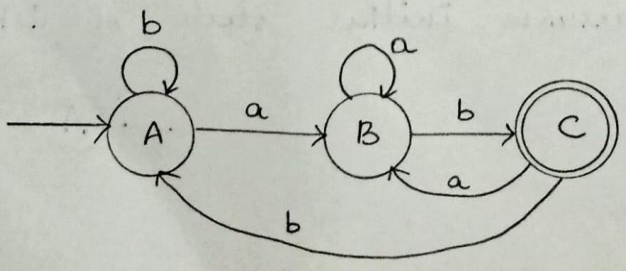
$= \delta(1,b) \cup \delta(3,b)$

$= \{1\} \cup \phi = \{1\}$ — (A)

Transition Table for DFA:

	States	a	b
→	A	B	A
	B	B	C
*	C	B	A

Transition Diagram for DFA:



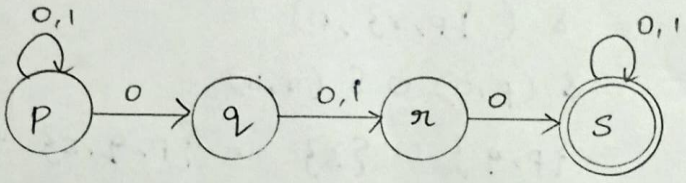
Ex: 2.

Construct a DFA equivalent to the NFA $M = (\{p, q, r, s\}, \{0, 1\}, \delta, p, \{s\})$ where δ is defined in the following table.

	State	0	1
→	p	{p, q}	{p}
	q	{r}	{r}
	r	{s}	ϕ
*	s	{s}	{s}

Solution:

Step 1 : Transition Diagram for NFA :



Let's assume that initial state p is (A) in DFA.

Step 2 : Processing of all states:

Processing of A :

$\delta(A, 0) = \delta(\{p\}, 0)$
$= \delta(p, 0) = \{p, q\}$ — (B)

$\delta(A, 1) = \delta(\{p\}, 1)$
$= \delta(p, 1) = \{p\}$ — (A)

Processing of B:

$\delta(B, 0) = \delta(\{p, q\}, 0)$
$= \delta(p, 0) \cup \delta(q, 0)$
$= \{p, q, r\}$ — (C)

$\delta(B,1) = \delta(\{p,q\},1)$

$= \delta(p,1) \cup \delta(q,1)$

$= \{p\} \cup \{r\} = \{p,r\}$ — (D)

Processing of C:

$\delta(C,0) = \delta(\{p,q,r\},0)$

$= \delta(p,0) \cup \delta(q,0) \cup \delta(r,0)$

$= \{p,q\} \cup \{r\} \cup \{s\} = \{p,q,r,s\}$ — (E)

$\delta(C,1) = \delta(\{p,q,r\},1)$

$= \delta(p,1) \cup \delta(q,1) \cup \delta(r,1)$

$= \{p\} \cup \{r\} \cup \phi = \{p,r\}$ — (D)

Processing of D:

$\delta(D,0) = \delta(\{p,r\},0)$

$= \delta(p,0) \cup \delta(r,0)$

$= \{p,q\} \cup \{s\} = \{p,q,s\}$ — (F)

$\delta(D,1) = \delta(\{p,r\},1)$

$= \delta(p,1) \cup \delta(r,1)$

$= \{p\} \cup \phi = \{p\}$ — (A)

Processing of E:

$\delta(E,0) = \delta(\{p,q,r,s\},0)$

$= \delta(p,0) \cup \delta(q,0) \cup \delta(r,0) \cup \delta(s,0)$

$= \{p,q\} \cup \{r\} \cup \{s\} \cup \{s\}$

$= \{p,q,r,s\}$ — (E)

$\delta(E,1) = \delta(\{p,q,r,s\},1)$

$= \delta(p,1) \cup \delta(q,1) \cup \delta(r,1) \cup \delta(s,1)$

$= \{p,r,s\}$ — (G)

Processing of F:

$\delta(F,0) = \delta(\{p,q,s\},0)$

$= \delta(p,0) \cup \delta(q,0) \cup \delta(s,0)$

$= \{p,q\} \cup \{r\} \cup \{s\}$

$= \{p,q,r,s\}$ — (E)

$\delta(F,1) = \delta(\{p,q,s\},1)$

$= \delta(p,1) \cup \delta(q,1) \cup \delta(s,1)$

$= \{p\} \cup \{r\} \cup \{s\}$

$= \{p,r,s\}$ — (G)

Processing of G:

$\delta(G,0) = \delta(\{p,r,s\},0)$

$= \delta(p,0) \cup \delta(r,0) \cup \delta(s,0)$

$= \{p,q\} \cup \{s\} \cup \{s\}$

$= \{p,q,s\}$ — (F)

$\delta(G,1) = \delta(\{p,r,s\},1)$

$= \delta(p,1) \cup \delta(r,1) \cup \delta(s,1)$

$= \{p\} \cup \phi \cup \{s\}$

$= \{p,s\}$ — (H)

Processing of H:

$\delta(H,0) = \delta(\{p,s\},0)$

$= \delta(p,0) \cup \delta(s,0)$

$= \{p,q\} \cup \{s\} = \{p,q,s\}$ — (F)

$\delta(H,1) = \delta(\{p,s\},1)$

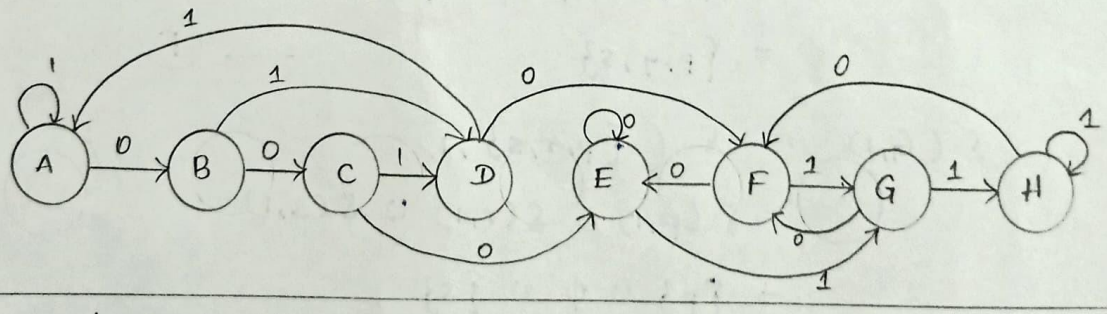
$= \delta(p,1) \cup \delta(s,1)$

$= \{p\} \cup \{s\} = \{p,s\}$ — (H)

Step 3: Transition Table for DFA:

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

Step 4: Transition Diagram for DFA:



Definition of NFA-ε:

NFA-ε is a 5 tuple of $M = (Q, \Sigma, \delta, q_0, F)$

where Q is the set of states

Σ is the set of i/p symbols

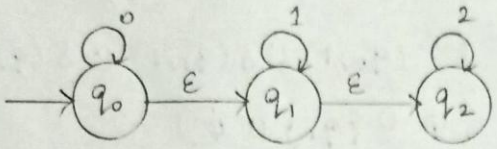
q_0 is the initial state

F is the set of accepting states and

$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow 2^Q$

Ex: 1:

Convert NFA-ε to NFA:



Solution:

Step 1: Transition Table for NFA-ε:

	State	0	1	2	ε
→	q_0	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
	q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
*	q_2	ϕ	ϕ	$\{q_2\}$	ϕ

Step 2: ε-closures for all states:

ε-closure $(q_0) = \{q_0, q_1, q_2\}$

ε-closure $(q_1) = \{q_1, q_2\}$

ε-closure $(q_2) = \{q_2\}$

Step 3: Processing of all states:

Processing of state q_0:

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

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

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

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

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

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

$\delta(q_0,1)$ = ε-closure (δ ($\hat{\delta}(q_0,\varepsilon)$, 1))

= ε-closure (δ ($\{q_0,q_1,q_2\}$, 1))

= ε-closure ($\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1)$)

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

= ε-closure (q_1)

= $\{q_1,q_2\}$

$\delta(q_0,2)$ = ε-closure (δ ($\hat{\delta}(q_0,\varepsilon)$, 2))

= ε-closure (δ ($\{q_0,q_1,q_2\}$, 2))

= ε-closure ($\delta(q_0,2) \cup \delta(q_1,2) \cup \delta(q_2,2)$)

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

= $\{q_2\}$.

Processing of state q_1 :

$\delta(q_1,0)$ = ε-closure (δ ($\hat{\delta}(q_1,\varepsilon)$, 0))

= ε-closure (δ ($\{q_1,q_2\}$, 0))

= ε-closure ($\delta(q_1,0) \cup \delta(q_2,0)$)

= ε-closure ($\phi \cup \phi$)

= ϕ

$\delta(q_1,1)$ = ε-closure (δ ($\hat{\delta}(q_1,\varepsilon)$, 1))

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

= ε-closure ($\delta(q_1,1) \cup \delta(q_2,1)$)

= ε-closure ($\{q_1\} \cup \phi$)

= ε-closure (q_1)

= $\{q_1,q_2\}$

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

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

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

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

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

$= \{q_2\}$

Processing of state q_2:

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

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

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

$= \phi$

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

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

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

$= \phi$

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

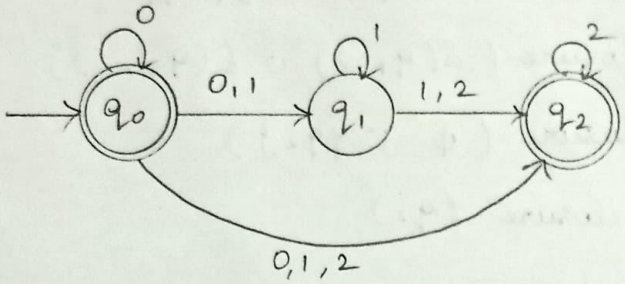
$= \varepsilon\text{-closure}(\delta(q_2,2))$

$= \{q_2\}$

Step 4: Transition Table for NFA:

	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	$\{q_2\}$

Step 5: Transition Diagram for NFA:



Note: If ε-closure (q_0) contains q_2, q_0 is also final state.

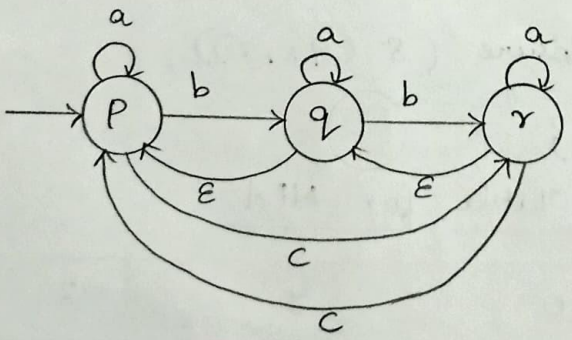
Ex: 2:

Convert NFA-ε to NFA.

	State	a	b	c	ε
→	p	{p}	{q}	{r}	ϕ
	q	{q}	{r}	ϕ	{p}
*	r	{r}	ϕ	{p}	{q}

Solution:

Step 1: Transition Diagram for NFA-ε



Step 2: ε-closures for all states:

ε-closure (p) = {p}

ε-closure (q) = {p, q}

ε-closure (r) = {p, q, r}

Step 3: Processing of all states:

Processing of state p:

$\delta(p,a) = \varepsilon\text{-closure}(\delta(\hat{\delta}(p,\varepsilon),a))$

$= \varepsilon\text{-closure}(\delta(\{p\},a))$

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

$= \{p\}$

$\delta(p,b) = \varepsilon\text{-closure}(\delta(\hat{\delta}(p,\varepsilon),b))$

$= \varepsilon\text{-closure}(\delta(\{p\},b))$

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

$= \{p,q\}$

$\delta(p,c) = \varepsilon\text{-closure}(\delta(\hat{\delta}(p,\varepsilon),c))$

$= \varepsilon\text{-closure}(\delta(\{p\},c))$

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

$= \{p,q,r\}$

Processing of state q:

$\delta(q,a) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q,\varepsilon),a))$

$= \varepsilon\text{-closure}(\delta(\{p,q\},a))$

$= \varepsilon\text{-closure}(\delta(p,a) \cup \delta(q,a))$

$= \varepsilon\text{-closure}(\{p\} \cup \{q\})$

$= \varepsilon\text{-closure}(p) \cup \varepsilon\text{-closure}(q)$

$= \{p,q\}$

$\delta(q,b) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q,\varepsilon),b))$

$= \varepsilon\text{-closure}(\delta(\{p,q\},b))$

$= \varepsilon\text{-closure}(\delta(p,b) \cup \delta(q,b))$

$= \varepsilon\text{-closure}(\{q\} \cup \{r\})$ = ε-closure(q) $\cup$ ε-closure(r)

$= \{p,q,r\}$

$\delta(q,c)$ = ε-closure $(\delta(\hat{\delta}(q,\varepsilon),c))$

= ε-closure $(\delta(\{p,q\},c))$

= ε-closure $(\delta(p,c) \cup \delta(q,c))$

= ε-closure (r)

= $\{p,q,r\}$

Processing of state r:

$\delta(r,a)$ = ε-closure $(\delta(\hat{\delta}(r,\varepsilon),a))$

= ε-closure $(\delta(\{p,q,r\},a))$

= ε-closure $(\delta(p,a) \cup \delta(q,a) \cup \delta(r,a))$

= ε-closure $(\{p,q,r\})$

= ε-closure $(p) \cup \varepsilon$-closure $(q) \cup \varepsilon$-closure (r)

= $\{p,q,r\}$

$\delta(r,b)$ = ε-closure $(\delta(\hat{\delta}(r,\varepsilon),b))$

= ε-closure $(\delta(\{p,q,r\},b))$

= ε-closure $(\delta(p,b) \cup \delta(q,b) \cup \delta(r,b))$

= ε-closure $(\{q,r\})$

= $\{p,q,r\}$

$\delta(r,c)$ = ε-closure $(\delta(\hat{\delta}(r,\varepsilon),c))$

= ε-closure $(\delta(\{p,q,r\},c))$

= ε-closure $(\delta(p,c) \cup \delta(q,c) \cup \delta(r,c))$

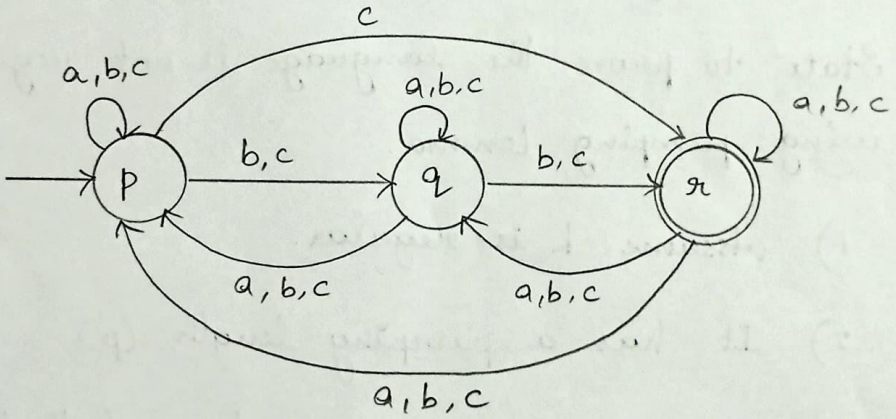
= ε-closure $(\{p,r\})$

= $\{p,q,r\}$

Step 4: Transition Table for NFA:

	State	a	b	c
→	p	{p}	{p,q}	{p,q,r}
	q	{p,q}	{p,q,r}	{p,q,r}
*	r	{p,q,r}	{p,q,r}	{p,q,r}

Step 5: Transition Diagram for NFA:



Accepting state of NFA = if ε-closure (q_0) does not contain r. So, only r is the final state of NFA.

Pumping Lemma:

* Pumping lemma is to be applied to show that certain languages are not regular.

* It should never show that a language is regular.

Application:

⇒ If L is regular, it satisfies the pumping lemma.

⇒ If L does not satisfy the pumping lemma, it is not regular.

State to prove the language is not regular by using pumping lemma.

1) Assume L is regular.
2) It has a pumping length (p)
3) Find the string w in L such that $|w| \geq p$
4) Divide w into xyz
5) $xy^iz \notin L$ for some i
6) $|xy| \leq p$
7) $|y| > 0$

Another method:

Proving languages not to be Regular:

Theorem: (The Pumping lemma for regular language)

Let L be a regular language. Then there exists a constant n (which depends on L) such that for every string w in L such that $|w| \geq n$, we can break 'w' into 3 strings, $w = xyz$, such that,

1) $y \neq \varepsilon$
2) $|xy| \leq n$
3) for all $k \geq 0$, the string xy^kz is also in L.

ie., repeating y, any number of times, or deleting it (the case $k = 0$), keeps the resulting string in language.

Ex: 1:

Convert the language using pumping lemma $A = \{a^n b^n \mid n \geq 0\}$ is not regular.

Proof:

Assume A is the regular language then we have pumping length 'p'.

$$A = a^n b^n$$

$$\Rightarrow A = a^p b^p$$

$p = 7$

$A = aaaaaaabbbbbbb$

Case 1 : The 'y' is in 'a' part.

$\underbrace{a\,a\,a}_{x}\,|\,\underbrace{a\,a\,a\,a}_{y}\,|\,\underbrace{b\,b\,b\,b\,b\,b}_{z}$

Case 2 : The 'y' is in 'b' part

$\underbrace{a\,a\,a\,a\,a\,a\,a\,b\,b}_{x}\,|\,\underbrace{b\,b\,b\,b}_{y}\,|\,\underbrace{b}_{z}$

Case 3 : The 'y' is in 'a' & 'b' part

$\underbrace{a\,a\,a\,a\,a}_{x}\,|\,\underbrace{a\,a\,b\,b}_{y}\,|\,\underbrace{b\,b\,b\,b}_{z}$

Condition $\Rightarrow xy^iz \notin A$

$\boxed{i = 2}$

$xy^iz = xy^2z$

Case 1 : $aaaaaaaaaabbbbbbb \Rightarrow 11 \neq 7$

Case 2 : $aaaaaaaabbbbbbbbb \Rightarrow 7 \neq 11$

Case 3 : $aaaaaaaabbaabbbbbb \Rightarrow$ Not Regular

Yes, by using pumping lemma, we proved that the conditions are not satisfied and so it is not a regular language.

Ex: 2:

Given language

$L = \{ a^n b^{2n} \mid n > 0 \}$ is not regular using pumping lemma.

Proof:

Assume L is regular, then we have pumping length 'p'.

$L = a^n b^{2n} \longrightarrow a^p b^{2p}$

$\boxed{p = 2} \Rightarrow$ a a b b b b

Case 1: The y is in part of 'a'

$\underbrace{a}_{x} \underbrace{a}_{y} \underbrace{b\ b\ b\ b}_{z}$

Case 2: The y is in part of 'b'

$\underbrace{a\ a}_{x} \underbrace{b\ b}_{y} \underbrace{b\ b}_{z}$

Case 3: The y is in part of 'a' and 'b'

$\underbrace{a}_{x} \underbrace{a\ b}_{y} \underbrace{b\ b\ b}_{z}$

Condition $\Rightarrow x y^i z \notin A$

$i = 2 \Rightarrow x y^2 z$

In Case 1: a a a b b b b $\Rightarrow$ $3 \neq 4$

In Case 2: a a b b b b b b $\Rightarrow$ $2 \neq 6$

In case 3 : a a b a b b b ⇒ Irregular

Yes, by using the conditions of pumping lemma, we can say that the language is not regular.

Applications of finite automata

Finite automata have several practical applications across various fields. Here are some notable applications.

Lexical Analysis:

* Finite automata are extensively used in compiler design for lexical analysis which involves tokenizing and scanning the source code of a programming language. Lexical analyzers employ finite automata to recognize and classify different lexical units such as keywords, identifiers, operators and literals.

Pattern recognition:

* Finite automata are employed in pattern-matching and recognition tasks. They can be used to search for specific patterns or sequences of characters within a given input. Applications include text processing, string matching and searching algorithms.

Network protocol Analysis:

Finite Automata are used in network protocol analysis and packet filtering. They can be

employed to define rules and match patterns in network traffic, enabling functionalities like intrusion detection, firewalls and network monitoring.

Digital circuit design:

* Finite Automata can model the behavior of digital circuits and aid in their design and testing. Sequential circuits, such as counters and state machines can be represented and analyzed using finite automata.

Natural language processing:

* Finite automata are used in natural language processing for tasks such as text tokenization, morphological analysis and part-of-speech tagging. They help in parsing and understanding the structure of natural language sentences.

Vending machines:

* Finite automata are an appropriate model for designing and implementing the control

logic of vending machines. They can manage the states and transitions required to process user inputs and dispense the appropriate products.

Regular Expression Processing :

* Finite automata are closely related to regular expressions, a powerfull tool for specifying patterns in strings. Regular expression engines often utilize finite automata to efficiently match and process regular expressions.

DNA Sequence Analysis :

* Finite automata find applications in bioinformatics for analyzing DNA sequences. They can be used to identify specific patterns or motifs within DNA sequences, aiding in gene identification, sequence alignment and genetic research.

What are the applications of finite automata to lexical analyzer ?

* Lexical analyzers employ finite automata to recognize and classify different lexical units such as keywords, identifiers, operators, and literals.

* Pattern recognition : Finite automata are employed in pattern-matching and recognition tasks.

* In lexical analysis, the source code is read, one character at a time and tokenized. Each word in the source code needs to be recognized and categorized as an identifier, a specific keyword, integer, floating point number, string, left or right parenthesis based on the pattern.

Lex Tools:

Lexical Analysis

⇒ it takes the input as a stream of characters and gives the output as tokens also known as tokenization. The tokens can be classified into identifiers, seperators, Keywords, Operators, Constant and spcial characters.

Lexical Analyzer:

* Lexical analysis identifies the lexical units in the source statement.

* It then classifies them into different lexical classes. eg. id's constants etc. and enter them into different tables.

* Lexical analysis builds a descriptor, called a TOKEN, for each lexical unit.

* A Token contains two fields: class code and number in class.

* Token is depicted as: Code #no, e.g. Id #2.

It has three phases:

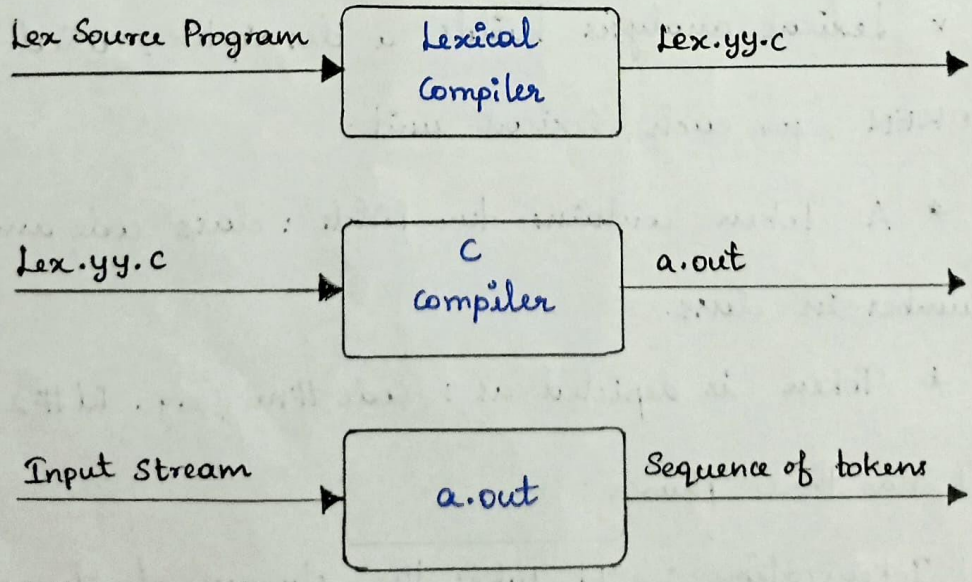
1) Tokenization: It takes the stream of characters and converts it into tokens.

2) Error Messages : It gives errors related to lexical analysis such as exceeding length, unmatched string etc.

3) Eliminate Comments : Eliminates all the spaces, blank spaces, new lines and indentations.

Lex :

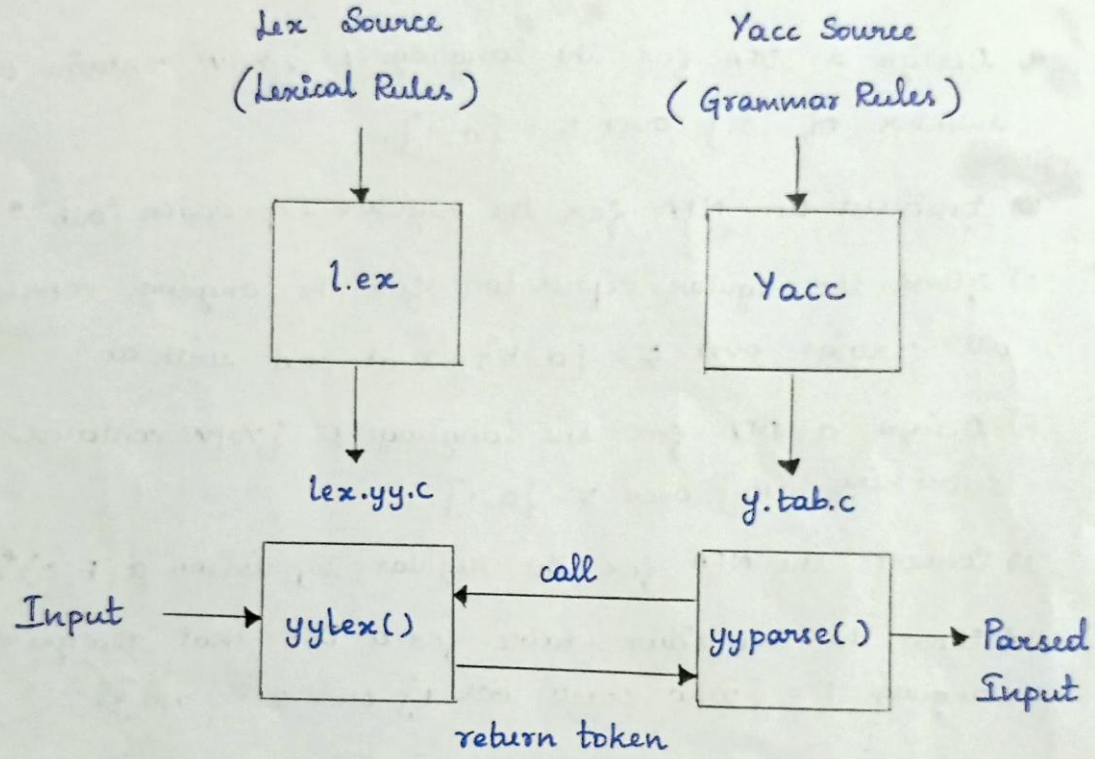
Lex is a tool or a computer program that generates Lexical Analyzers (converts the stream of characters into tokens). The Lex tool itself is a compiler. The Lex compiler takes the input and transforms that input into input patterns. It is commonly used with YACC (Yet Another Compiler Compiler). It was written by Mike Lesk and Eric Schmidt.



CHENNAI INSTITUTE OF TECHNOLOGY
(Autonomous)

40

Lex with YACC:



PART - A (2 marks)

1) Define a regular expression.
2) What is a finite automation?
3) Distinguish between DFA and NFA.
4) What is the pumping lemma for regular languages.
5) How is the epsilon transition represented in an NFA.
6) What is difference between an alphabet and a string in formal language theory?

7) Give an example of a regular expression and the corresponding language.

8) Define the term "lexical analysis" in compiler design

9) Design a DFA for the language $L = \{w | w$ contains an even number of 1s$\}$ over $\Sigma = \{0, 1\}$.

10) Construct an NFA for the regular expression $(a|b)^*ab$.

11) Given the regular expression for the language consisting of all strings over $\Sigma = \{a, b\}$ that end with ab.

12) Design a DFA for the language $L = \{w | w$ contains the substring $010\}$ over $\Sigma = \{0, 1\}$.

13) Convert the NFA for the regular expression $a(b|c)^*d$ to a DFA.

14) Show the transition table for a DFA that recognizes the language $L = \{w | w$ starts with $1\}$ over $\Sigma = \{0, 1\}$.

15) Design a DFA for the language $L = \{w | w$ contains no consecutive 1s$\}$ over $\Sigma = \{0, 1\}$.

16) Draw the transition diagram for an NFA that accepts strings over $\Sigma = \{a, b\}$ ending with ba.

17) Design a DFA for the language $L = \{w | w$ contains at most one 1$\}$ over $\Sigma = \{0, 1\}$.

18) What is the role of regular expressions in pattern matching ?

19) Define an epsilon closure in the context of NFA.

20) What is the function of lex tool ?