Finite Automata

is a device consisting of a tape and a control circuit

which satisfy the following conditions:

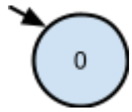
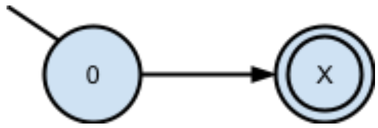
1. The tape starts from the left end and extends to the right without an end.
1. The tape is divided into squares in each symbol.
2. The tape has a read only head.
3. The head moves to the right one square every time it reads a symbol. It never moves to the left. When it sees no symbol, it stops and the Finite automata terminates its operation.
4. There is a control determines the state of the automaton and also controls the movement of the he

A finite automaton consists of a finite set of states, a set of transitions (moves), one start state, and a set of final states (accepting states). In addition, a DFA has a unique transition for every state combination. It is a set of states, and its “control” moves from state to state in response to external “inputs” .A finite automaton, FA, provides the simplest model of a computing device. It has a central processor of finite capacity and it is based on the concept of state.

finite state machine is a M = (Q, A, T, S, F) , where

- Q — set of states = {q1, q2, q3, … }
- A — set of input symbols ={a,b, …, 0, 1, …}
- T — set of transitions or rules
- S — an initial state
- F — the final state -- could be more than one final state

Designing (drawing) FA

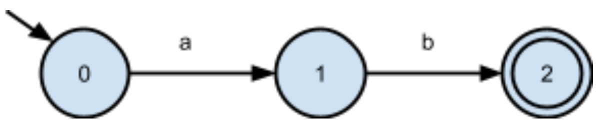
States	
start state	
end state	
transportations	

String is accepted by a FA if and only if the FA starts at the initial state and ends in an accepting state after reading the string.

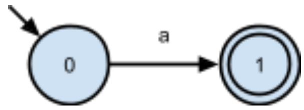
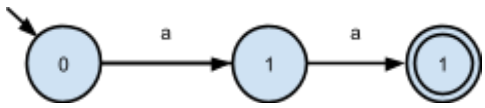
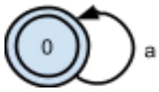
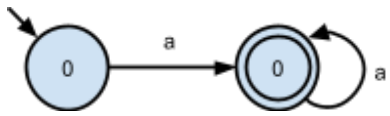
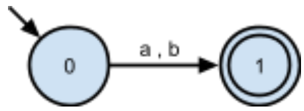
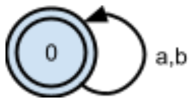
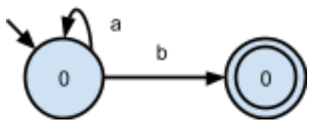
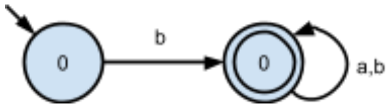
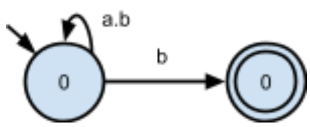
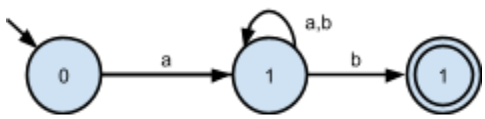
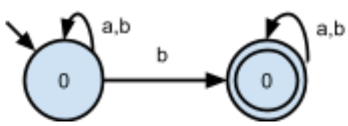
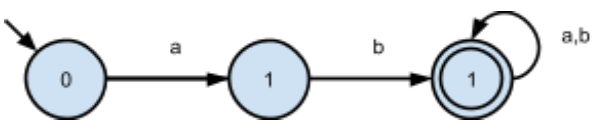
A FA represents a finite state machine that recognizes a RE.

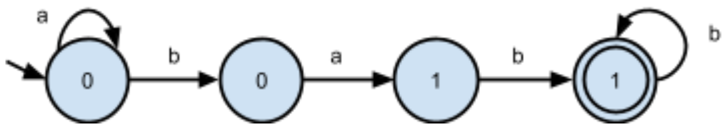
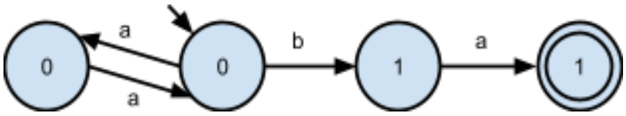
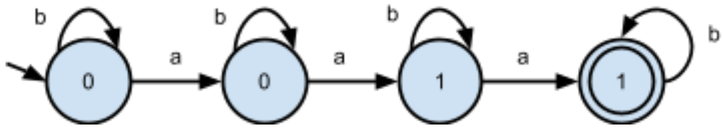
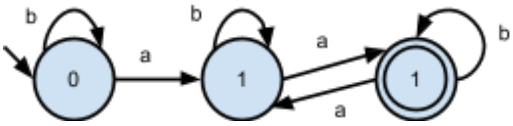
Example:

the following FA:



recognize (accept) string ab.

<table>
<tr><th>RE</th><th>FA</th></tr>
<tr><td>a</td><td></td></tr>
<tr><td>aa</td><td></td></tr>
<tr><td>a*</td><td></td></tr>
<tr><td>$a^+ = aa^*$</td><td></td></tr>
<tr><td>(a + b)</td><td></td></tr>
<tr><td>(a + b)*</td><td></td></tr>
<tr><td>a*b</td><td></td></tr>
<tr><td>b (a + b)*</td><td></td></tr>
<tr><td>(a + b)* b</td><td></td></tr>
<tr><td>a (a + b)* b</td><td></td></tr>
<tr><td>(a+b)* b (a+b)*</td><td></td></tr>
<tr><td>ab (a+b)*</td><td></td></tr>
</table>

a*babb*	
(aa)* ba	
b*ab*ab*ab* contains 3 a’s	
(b*ab*ab*)$^+$ contains even number of a , at least 2 a	

Exercise:

Design a FA

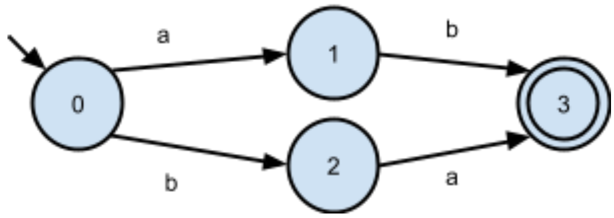
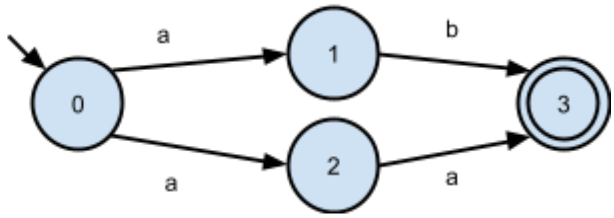
- All the Strings that begin with a and end with b
- All the Strings whose number of a’s is a multiple of 5
- All the Strings with a’s more than b’s
- All the strings with 3 a only.
- All the strings with even number of a
- **(aba)***
- **ab*a**
- **a*b***
- **(ab + ba)***
- **(a + b) a* (b + a)$^+$**
- **Aa$^+$**
- **(ab*)$^+$.**

Deterministic Finite Automata DFA and Non Deterministic Finite Automata NFA

There are two kinds of FA :

1- Deterministic Finite Automata (DFA)

2- Non Deterministic Finite Automata (NFA

DFA	
NFA	

<u>Converting NFA into DFA</u>

Three steps :

1- find the transition table for the NFA

2- create a transition table for the DFA

3- design a DFA

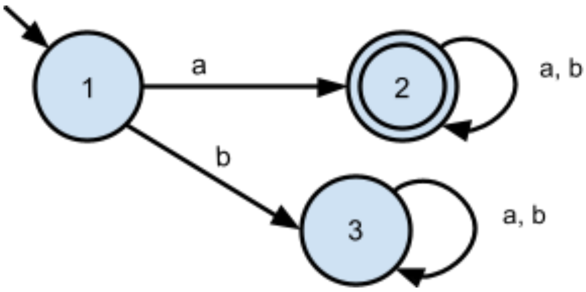
Example 1 : convert the following NDFA into DFA

NFA Transaction table

state	a	b
1	2	#
2	2	2

DFA Transition table **Let # = 3**

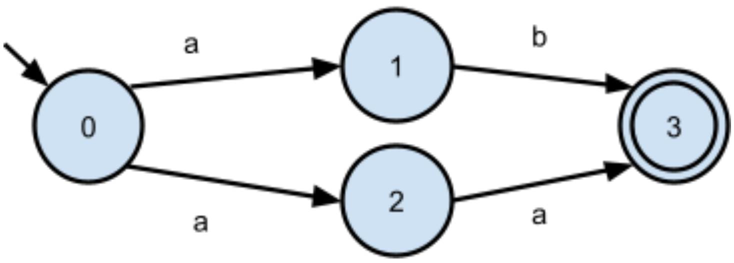
state	a	b
1	2	3
2	2	2
3	3	3



Example 2 : convert the following NDFA into DFA

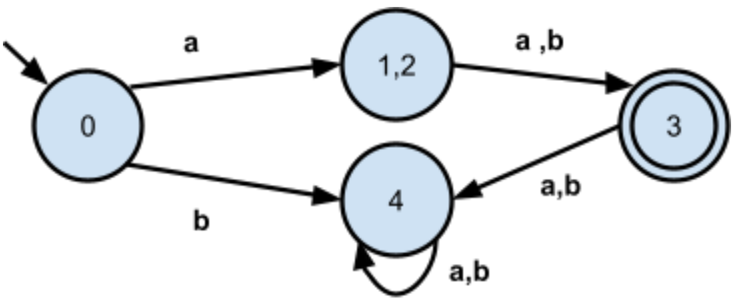
NFA Transaction table

state	a	b
0	1	2
1	#	3
2	3	#
3	#	#



DFA Transition table **Let # = 4**

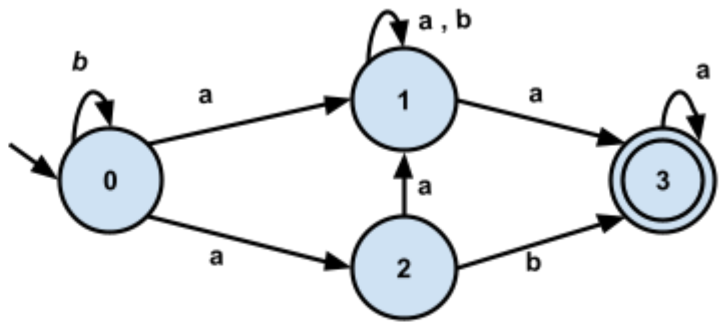
state	a	b
0	1 , 2	4
1 , 2	3	3
3	4	4
4	4	4



Example 3 : convert the following NDFA into DFA

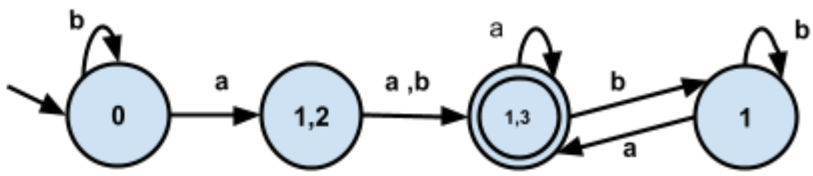
NFA Transaction table

state	a	b
0	1 , 2	0
1	1 , 3	1
2	1	3
3	3	#



DFA Transition table

state	a	b
0	1 , 2	0
1 , 2	1 , 3	1 , 3
1 , 3	1 , 3	1
1	1 , 3	1



Exercise:

Convert the following NFA into DFA