FINITE AUTOMATA

- Finite automata are a mechanism to recognize a set of valid inputs before carrying out an action.
- Finite automata are a mathematical model of a system with inputs, outputs, finite number of states and a transition from state to state on input symbol Σ.
- A recognizer for a language is a program that takes a string x, and answers "yes" if x is a sentence of that language, and "no" otherwise.
- We call the recognizer of the tokens as a finite automaton. A finite automaton can be: deterministic (DFA) or non-deterministic (NFA). Both deterministic and nondeterministic finite automaton recognize regular sets.

Non Deterministic Finite Automata:

NFA (Non-deterministic Finite Automaton) is a 5-tuple (S, Σ, δ, S0, F):

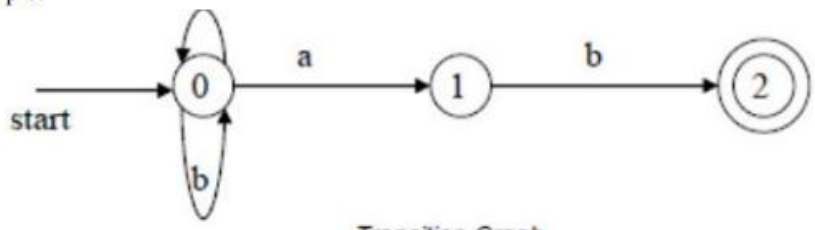
- S: a set of states
- Σ: the symbols of the input alphabet
  δ is a set of transition function
- S0: s0 ∈S, the start state
- F: F ⊆ S, a set of final or accepting states.

Transition table

A transition table is a good way to implement a FSA

- One row for each state, S
- One column for each symbol, A
- Entry in cell (S,A) gives the state or set of states can be reached from state S on input A.

Example:



Transition Graph

0 is the start state s0
{2} is the set of final states F
Σ = {a,b}
S = {0,1,2}

Transition Function:

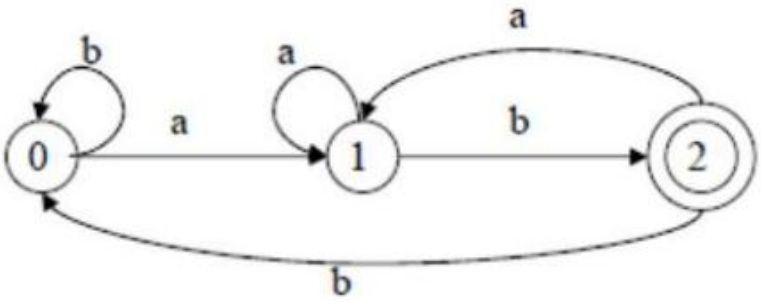
	a	b
0	{0,1}	{0}
1	{}	{2}
2	{}	{}

The language recognized by this NFA is (a|b)*ab

Deterministic Finite Automaton (DFA):

- A Deterministic Finite Automaton (DFA) is a special form of a NFA.
- No state has ε- transition
- For each symbol a and state s, there is at most one labeled edge a leaving s. i.e. transition function is from pair of state-symbol to state (not set of states)

The DFA to recognize the language (a|b)* ab is as follows.



Transition Graph

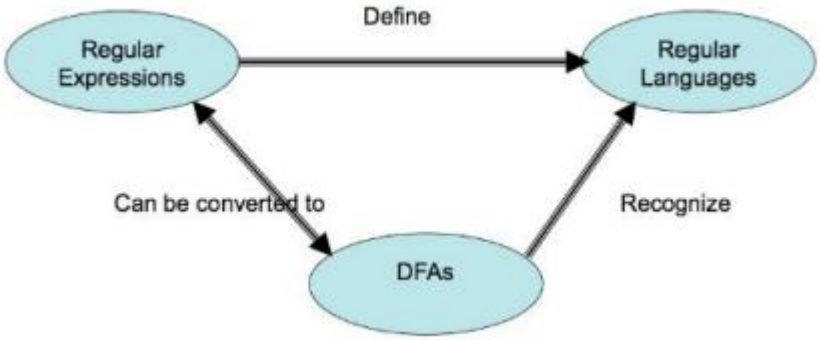
0 is the start state s0
{2} is the set of final states F
Σ = {a,b}
S = {0,1,2}

Transition Function:

	a	b
0	1	0
1	1	2
2	1	0

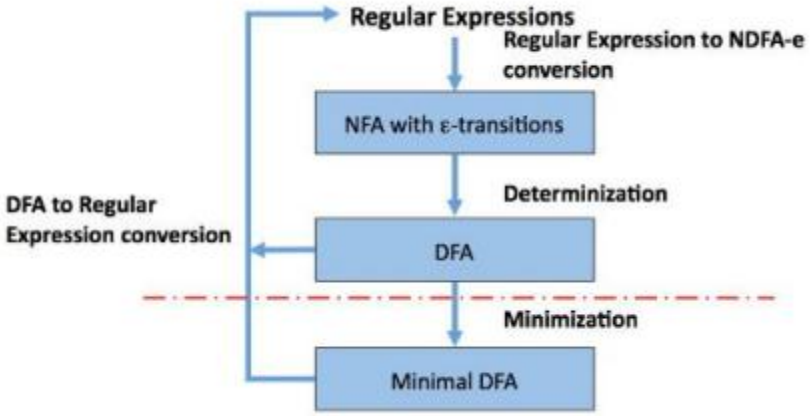
Construction of an NFA and DFA from a Regular Expression:

- To convert a regular expression to a NFA using McNaughton-Yamada-Thompson algorithm
- syntax-directed: It works recursively up the parse tree of the regular expression
- For each sub expression a NFA with a single accepting state is built



Conversion of a NFA to a DFA:

- Subset construction: Each state of DFA corresponds to a set of NFA states.
- DFA states may be exponential in number of NFA states.



Operations on NFA states:

Operation	Description
ε-closure(s)	set of NFA states reachable from NFA state *s* on ε- transition alone
ε-closure(T)	set of NFA states reachable from some NFA state *s* in set T on ε-transitions alone
move(T,a)	set of NFA states to which there is a transition on input symbol a from some state s in T