Finite State Automata

What is Finite State Automata?

- It is a graph like structure also known as directed graph.
- Finite state automata (FSAs) sound complicated, but the basic idea is as simple as drawing a map.
- A finite-state automaton is a device that can be in one of a finite number of *states*.
- In certain conditions, it can switch to another state.
    - This is called a *transition*.
- When the automaton starts working (when it is switched on), it can be in one of its initial states.
- There is also another important subset of states of the automaton: the final states.
    - If the automaton is in a final state when it stops working, it is said to *accept* its input. The input is a sequence of symbols.
- The interpretation of the symbols depends on the application; they usually represent events, or can be interpreted as ``the event that a particular data became available".
- The symbols must come from a finite set of symbols, called the *alphabet*.
- If a particular symbol in a particular state triggers a transition from that state to another one, that transition is *labeled* with that symbol.
    - The labels of transitions can contain one particular symbol that is in the alphabet.
    - A transition is labeled with ε (not present in the alphabet) if it can be traversed with no input symbol.
- It is convenient to present automata as directed graphs.
- The vertices denote states.
- They are portrayed as small circles.
- The transitions form the edges - arcs with arrows pointing from the source state (the state where the transition originates) to the target state.
- They are labeled with symbols. Unless it is clear from the context, the initial states have short arrows that point to them from ``nowhere".
- The final states are represented as two concentric circles.

Why to Study Finite State Automata?

- FSA is machine to model the regular languages.
- So, to model the problem belonging to regular language needs finite state automata

Applications

Finite Automata are widely used in:

- **Lexical analyzers** (compilers)
- **Text pattern searching** (like regex)
- **Network protocol design**
- **Control systems**
- **Robotics and AI behavior modeling**

Formal A **Finite State Automaton (FSA)** is a 5-tuple:

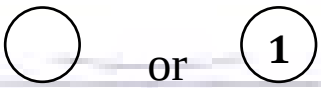
M=(Q, Σ, δ, q0, F)

where:

Symbol	Meaning
Q	Finite set of states
Σ (Sigma)	Finite set of input symbols (alphabet)
δ (delta)	Transition function → δ:Q×Σ→Q
q_0	Start state (initial state), where the machine begins
F	Set of final or accepting states, F⊆QF

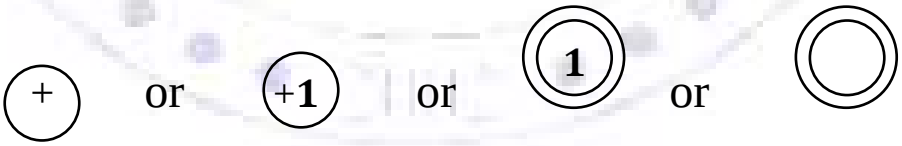
Notations Used to build FSA

- Sates are represented by only circles or circles having name inside it.



- Initial Sate is represented by circle having negative sign inside it or arrow symbol like

- Transitions having label from ∑ o Transitions omits from states are called outgoing transitions for that particular state.
    - o Transitions comes to states are called incoming transitions.
- Final state or acceptance state is represented with double circle or positive sign inside.



How It Works

- The automaton **starts** in the start state q_0.
- It **reads input symbols** one by one.
- For each symbol, it moves to the **next state** based on the transition function δδδ.
- After all input is read, if it ends in a **final (accepting) state**, the input is **accepted**; otherwise, it is **rejected**.

Manual Conversion Steps

If you're drawing by hand or for an exam:

1. **Start with smallest parts:**
   - a → single transition on 'a'
   - b → single transition on 'b'
2. **Apply rules:**
   - **Concatenation (AB):** connect end of A to start of B
   - **Union (A|B):** add a new start and end with ε-transitions
   - *Kleene star (A):** add loops using ε-transitions
3. **Combine** until full expression is represented.

(An **ε-transition** (epsilon transition) is a **move from one state to another without reading any input symbol**.)

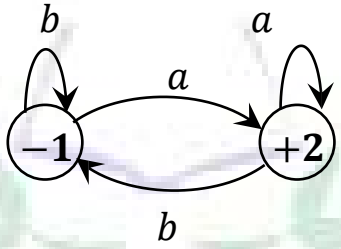
Example1: The language end on "a". $\Sigma = \{a, b\}$

R.E= (a + b)*a

Valid= {a, aa, ba, aba, baba….}

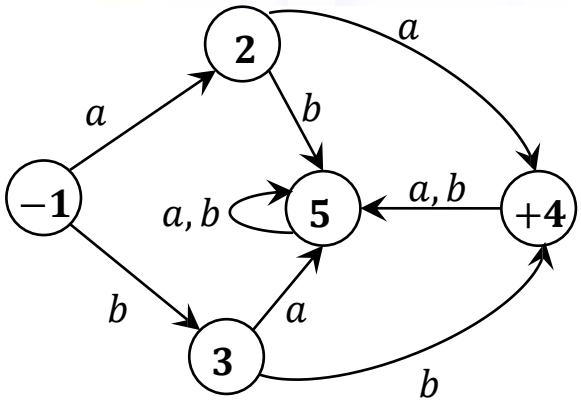
Transition Table

State	Inputs	
	a	b
-1	2	1
+2	2	1



Example2: Language having words {aa, bb} over ∑ = {a, b}

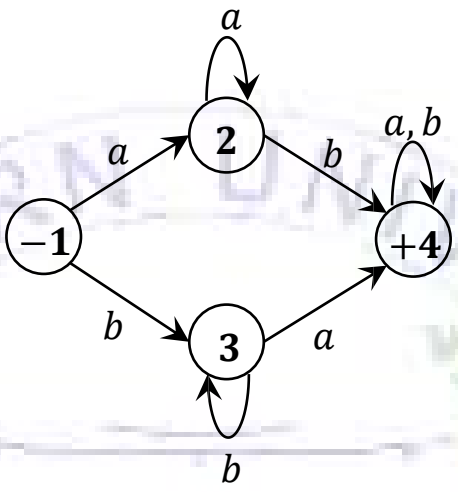
R.E= aa + bb



Example3: The language of all words that have at least one a and at least one b is somewhat trickier.

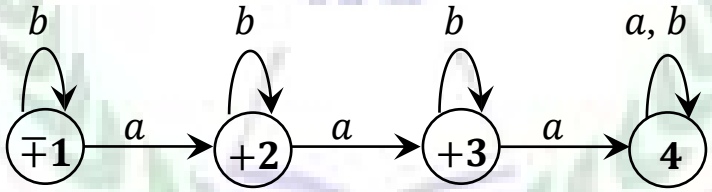
R.E= (a+b)*a(a+b)*b(a+b)* + (a+b)*b(a+b)*a(a+b)*

Valid={ab,ba,aab,bba,aba,bab,…}



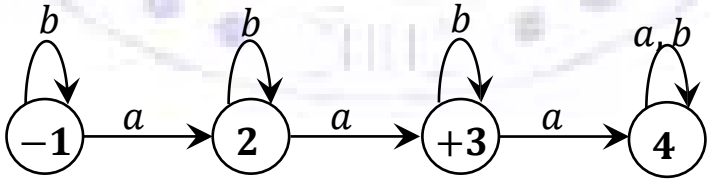
Example4: The Language contain at most two a's defined over, ∑={a, b}

R.E= b*+b*a b*+ b*a b*ab*



Example5: The Language contain exact two a's defined over, ∑={a,b}

RE=b*ab*ab*

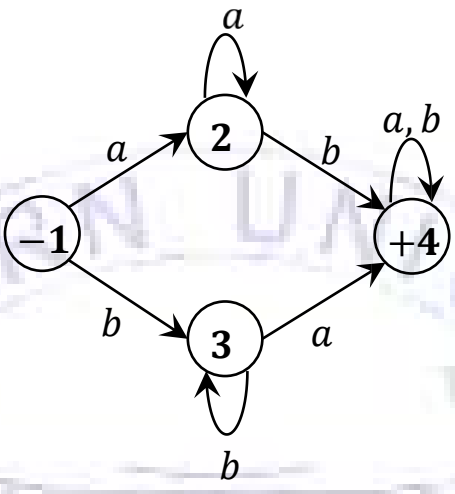


Example6: The language contain substring ab or ba defined over, ∑= {a, b}

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

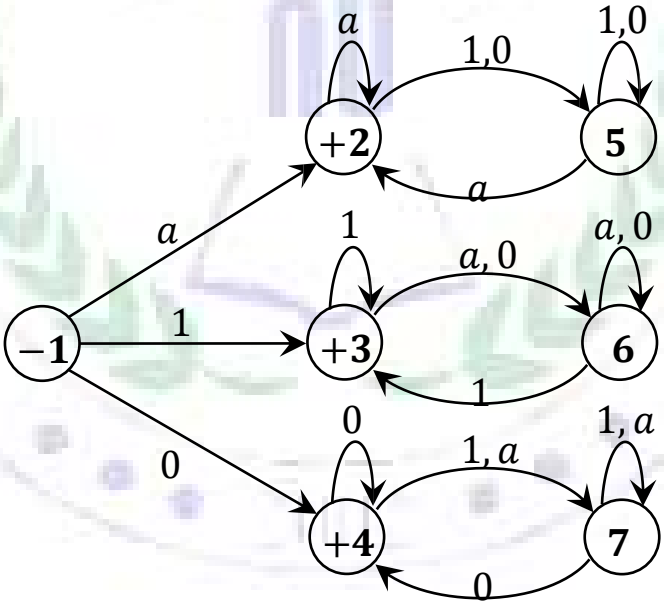
Or

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



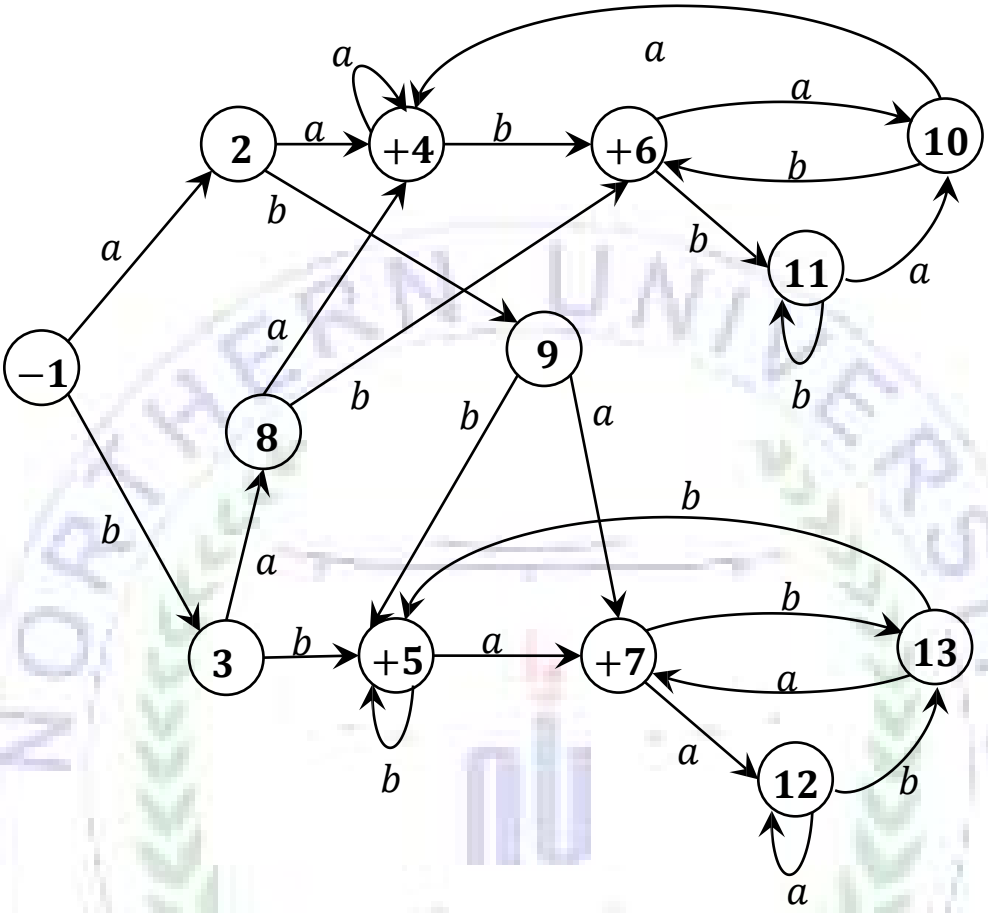
Example7: The language start and ends on same letter defined over ∑= {a, 1, 0}

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

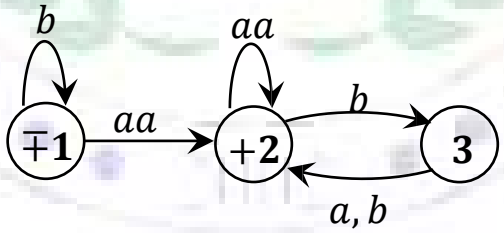


Example8: **The language whose second and second last letter are same defined over** $\Sigma = \{a, b\}$.

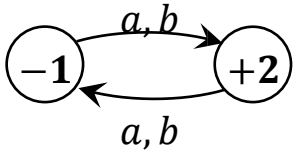
R.E= (a+b)a(a+b)* a(a+b) + (a+b)b(a+b)* b(a+b)



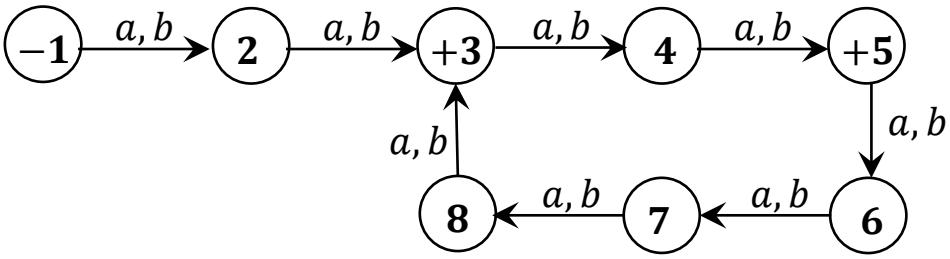
Example9: **The language cannot end on 'ab' defined over,** $\sum$**= {a, b}**



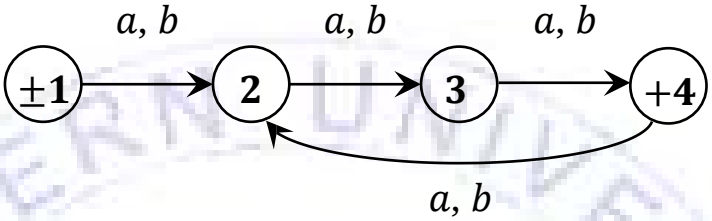
Example10: The language cannot contain even length of word defined over, $\sum$**= {0, 1, 2}**



Example11: The Language contain even length of word but not multiple of 3 defined over, $\sum$**={a,b}.**



Example12: The language contain word of length multiple of 3 defined over, ∑={a,b}



Assignment No. 3

Note : All the languages are based on set of alphabets Σ = $\{a,b\}$

Question 1: Language of all words having at least two a's and 3 b's

Question 2: Language of all words which start with 'ab' and end with 'bb'

Question 3: Language of all words having even length

Question 4: Language of all words which contain substring 'ab.

Question 5: Language which has odd length but start with 'b'

Question 6: Language of all strings which has exact 1 'aaa' and must start with 'b'

Question 7: Language which accept length >=2 and not end with 'aa' or 'bb'