Finite Automata

Theory Of Automata

Finite Automata

- So far, we have studied Different ways of defining languages i.e., Descriptive, Recursive, Regular Expression.
- Now we move towards Finite Automata (FA).
- In Finite Automata we can represent language in form of diagram or graph.
- Finite Automata is also known as:
  - Finite Machine (FM)
  - Finite Automatic Machine (FAM)
  - Finite State Machine (FSM)

Finite Automata

Method 4 (Finite Automaton)

Definition:

A **Finite automaton (FA)**, is a collection of the followings

1. Finite Automata is simplest model of computation.
2. It has limited memory.

Finite Automata

Method 4 (Finite Automaton)

Definition:

A **Finite automaton (FA)**, is a collection of the followings

1. Finite number of states, having one initial and some (maybe none) final states.

2. Finite set of input letters (Σ) from which input strings are formed.

3. Finite set of transitions *i.e.*, for each state and for each input letter there is a transition showing how to move from one state to another.

Finite Automata (FA)

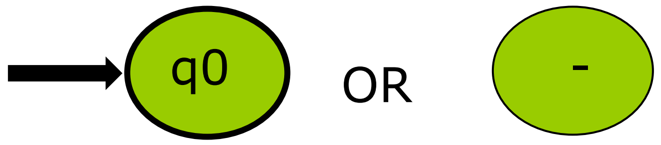
- Finite Automata (FA) has 5 tuples i.e.,

1. Q = set of all states
2. Σ = set of inputs symbols
3. q_0 = start state
4. F = final state
5. δ = Transition function that maps Q x Σ $\longrightarrow$ Q

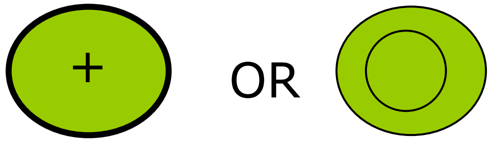
Note: transition function describes moment of states from one to another state.

Finite Automata

1. q0 (start state or initial state) of Finite Automata (FA)
2. In every FA, there will be only one initial state
3. It is represented by represent input.



4. At final state, machine may stop or may not stop depend on your language (input)
5. It is represented by:

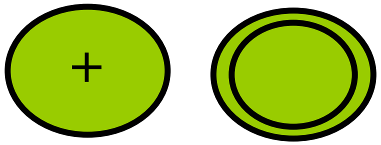


Finite Automata

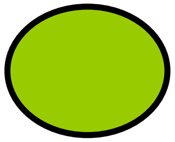
Final State

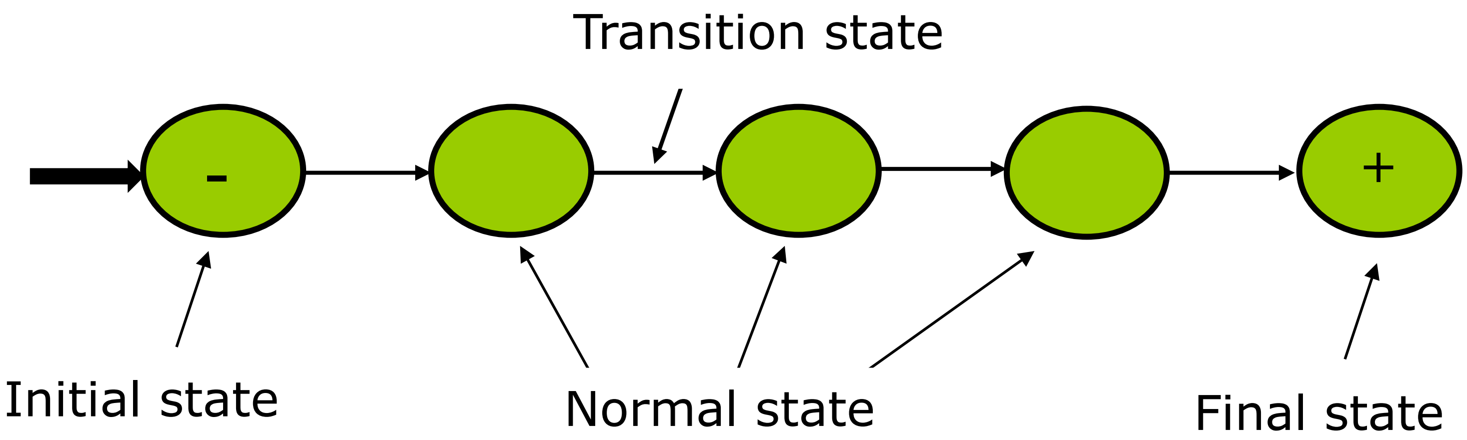
- In FA the minimum number of final state should be 1 and maximum number of final state can be more than 1.

- It means we may have more than q final state in FA.
- At final state, machine may stop or may not stop depending upon language (input).
- It is represented by



Finite Automata

1. Q (states/normal states) these are the states other then initial and final states.
2. These are states from which our FA neither start nor finish.
3. It is represented by: 
4. δ: transition / movement : transition or movement is represented with $\longrightarrow$ sign.



Types of Finite Automata (FA)

- Finite Automata without output
  - Deterministic Finite Automata (DFA).
  - Non-Deterministic Finite Automata (NFA or NDFA).
  - Non-Deterministic Finite Automata with epsilon moves (e-NFA).
- Finite Automata with Output
  - Moore machine.
  - Mealy machine.

Will be discussing in future lectures

Finite Automata Example

- $\Sigma = \{a,b\}$ // input

  States: x, y, where x is both initial and final state.

  Transitions:

  1. At state x reading a or b go to state y.
  2. At state y reading a or b go to state x.

Example Continued …

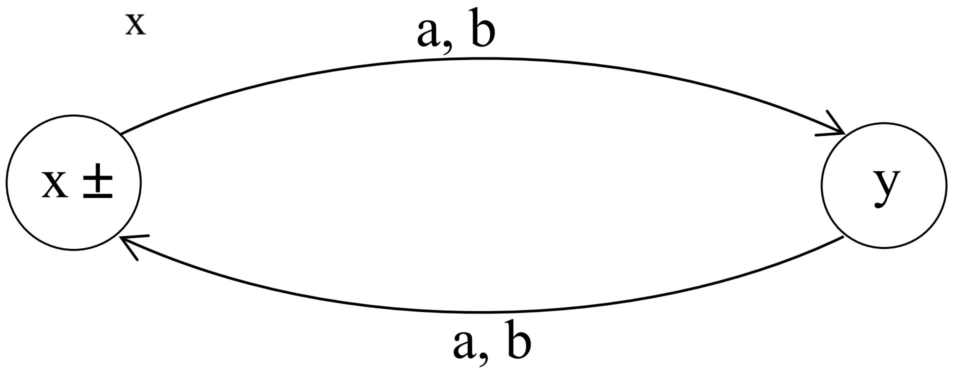
- These transitions can be expressed by the following transition table

Old States	New States	
	Reading a	Reading b
x ±	y	y
y	x	x

Example Continued …

- It may be noted that the transition table may be depicted by the following transition diagram.

Old States	New States	
	Reading a	Reading b
x ±	y	y
y	x	x

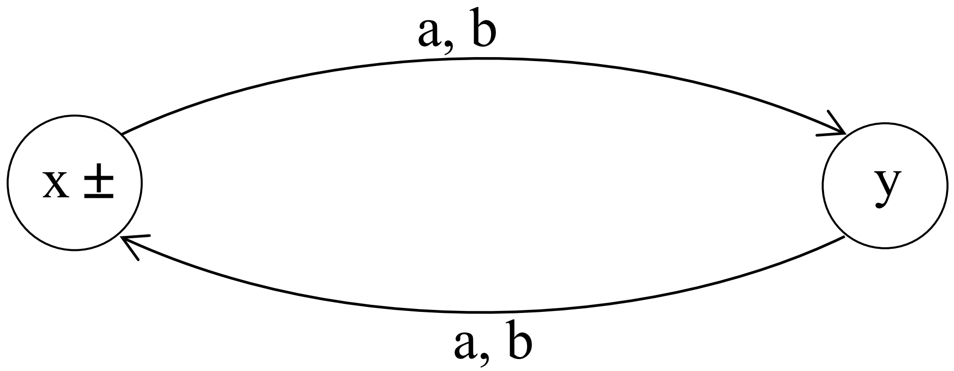


$$\begin{tikzcd} x\pm \arrow[r, bend left, "{a, b}"] & y \arrow[l, bend left, "{a, b}"] \end{tikzcd}$$

Example Continued …

- The transition diagram is an FA accepting the language of strings, defined over Σ={a, b} of **even length**. It may be noted that this language may be expressed by the regular expression

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



Note

- It may be noted that corresponding to a given language there may be more than one FA accepting that language, but for a given FA there is a unique language accepted by that FA.

Note

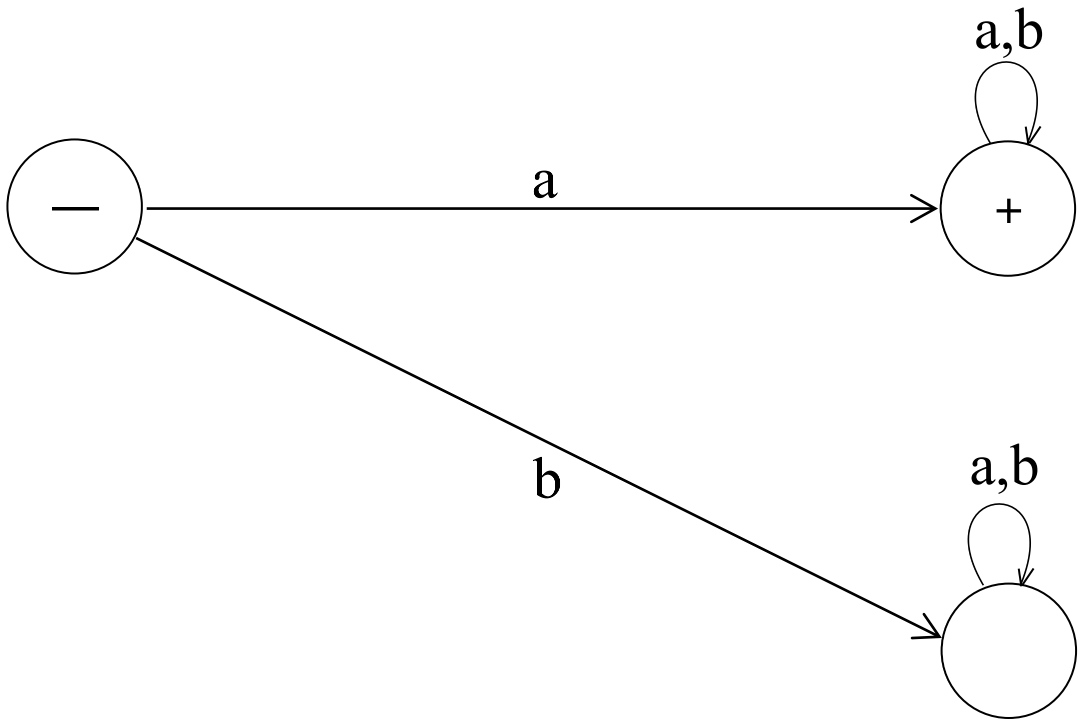
It is to be noted that for given the languages L_1 and L_2 ,where:

L_1 = The language of strings, defined over $\Sigma=\{a, b\}$, **beginning with a**

L_2 = The language of strings, defined over $\Sigma=\{a, b\}$, **not beginning with b**

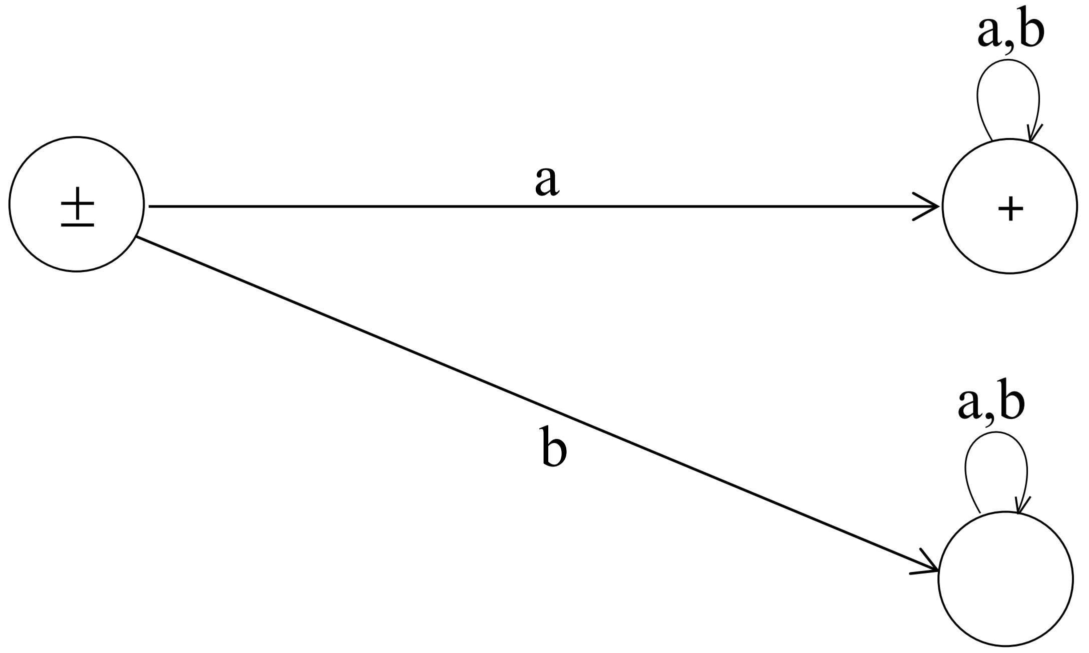
- The Λ does not belong to L_1 while it does belong to L_2 .
- This fact may be depicted by the corresponding transition diagrams of L_1 and L_2.

FA_1 Corresponding to L_1



- The language L_1 may be expressed by the regular expression $a(a + b)^*$

FA$_2$ Corresponding to L$_2$



- The language L$_2$ may be expressed by the regular expression $a\,(a + b)^* + \Lambda$

Example

- Consider the Language L of Strings of **length two or more**, defined over Σ = {a, b}, **beginning with and ending in same letters.**

  The language L may be expressed by the following regular expression

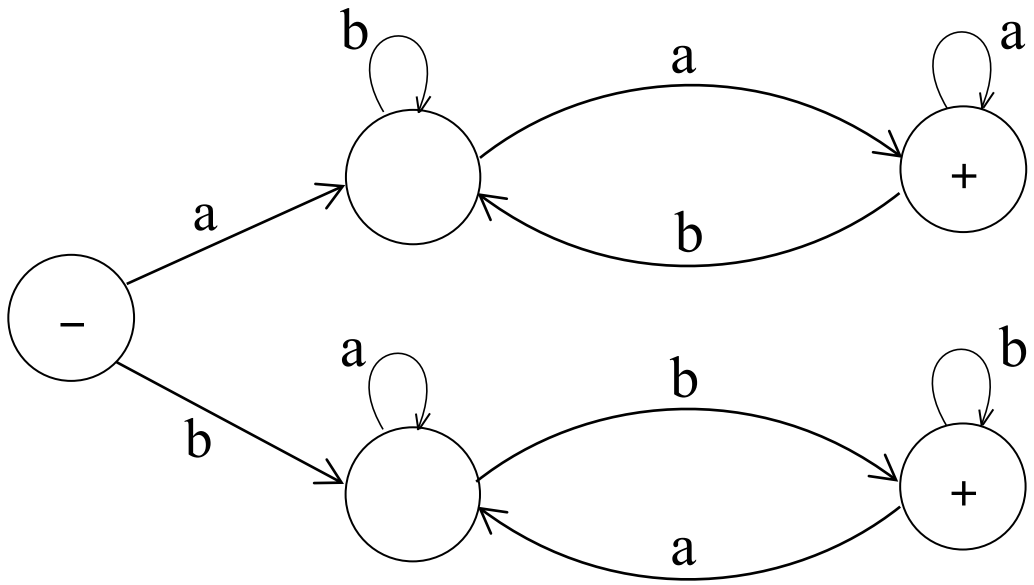
  $$a\ (a + b)^* a + b\ (a + b)^* b$$

  It is to be noted that if the condition on the length of string is not imposed in the above language, then **the strings a and b will then belong to the language.**

  This language L may be accepted by the following FA

Example Continued …

FA for the given language



Summing Up

- Different notations of transition diagrams, languages of strings of **even length, Odd length, starting with b, ending in a, beginning with b, not beginning with b, beginning and ending in same letters;**

Non-Deterministic Finite Automata (NFA)

Non-Deterministic Finite Automata (NFA)

- In NFA, for a particular input symbol, the machine can move to any combination of the states in the machine.
- In other words, the exact state to which the machine moves cannot be determined.
- Hence, it is called **Non-deterministic Automaton**.
- As it has finite number of states, the machine is called **Non-deterministic Finite Machine** or **Non-deterministic Finite Automaton**

Non-Deterministic Finite Automata (NDFA)

- Finite Automata (FA) has 5 tuples i.e.,

1. Q = set of all states
2. Σ = set of inputs symbols
3. q_0 = start state
4. F = final state
5. δ = Transition function that maps Q x Σ $\longrightarrow$ Q

Note: transition function describes moment of states from one to another state.

Example

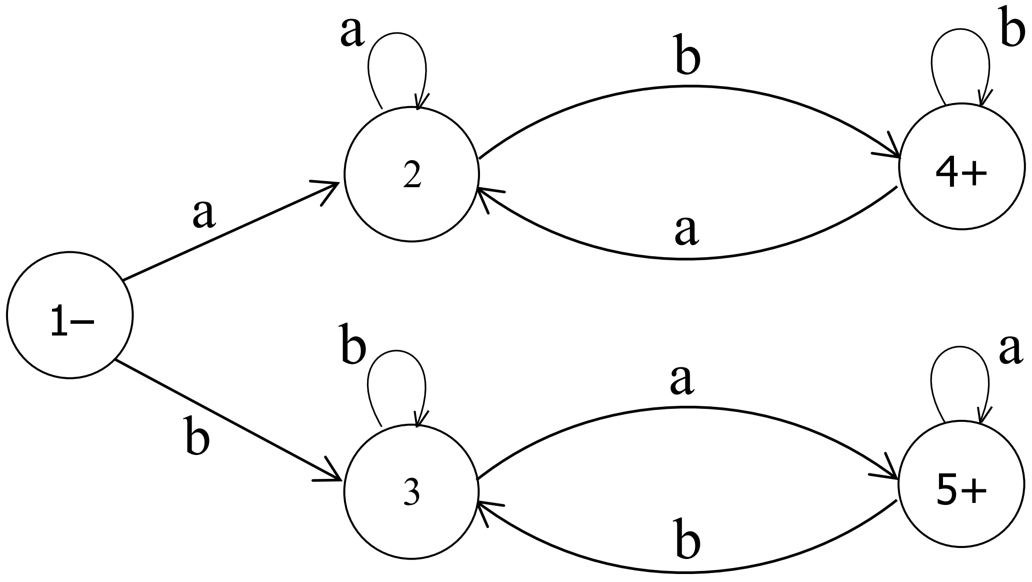
Consider the Language L of Strings , defined over Σ = {a, b}, **beginning with and ending in different letters.**

The language L may be expressed by the following regular expression

$$a\ (a + b)^{*}\ b + b\ (a + b)^{*}\ a$$

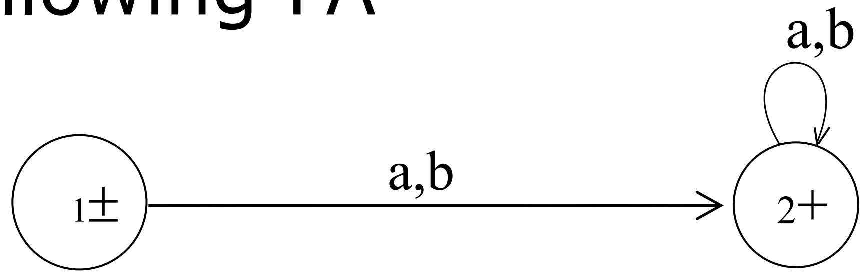
This language may be accepted by the following FA

Example Continued …



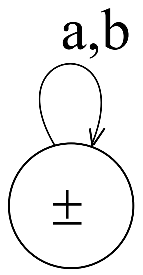
Example

- Consider the Language L , defined over $\Sigma = \{a, b\}$ of **all strings including Λ**, The language L may be accepted by the following FA



- The language L may also be accepted by the following FA

Example Continued …

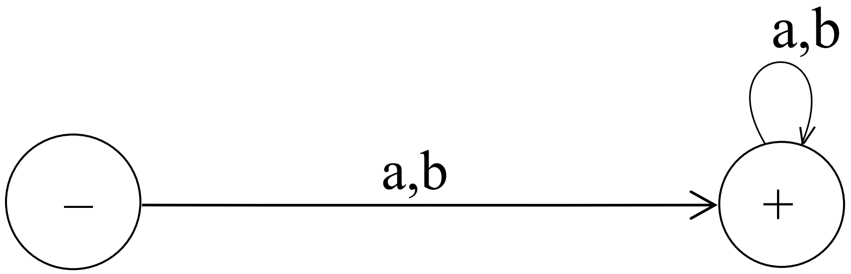


- The language L may be expressed by the following regular expression

$$(a + b)^*$$

Example

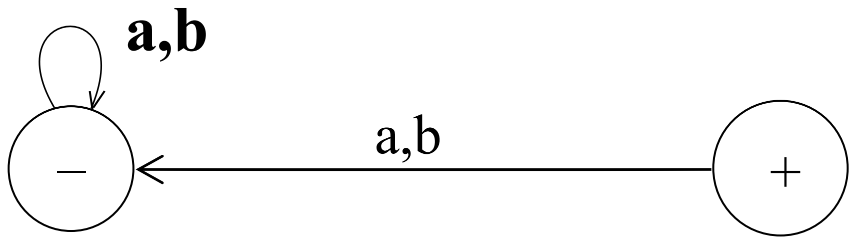
- Consider the Language L , defined over Σ = {a, b} of **all non empty strings**. The language L may be accepted by the following FA



The above language may be expressed by the following regular expression $(a + b)^+$

Example

- Consider the following FA, defined over
  Σ = {a, b}



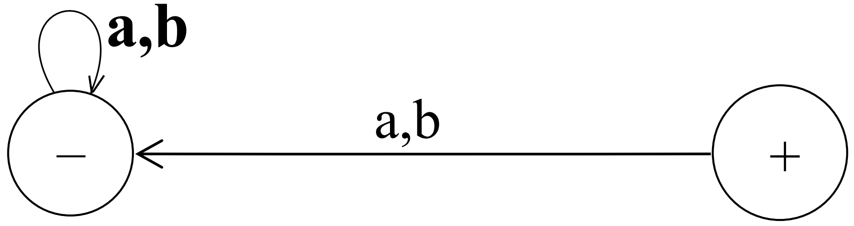
- It is to be noted that the above FA **does not accept any string**. Even it does not accept the null string. As there is no path starting from initial state and ending in final state.

Equivalent FAs

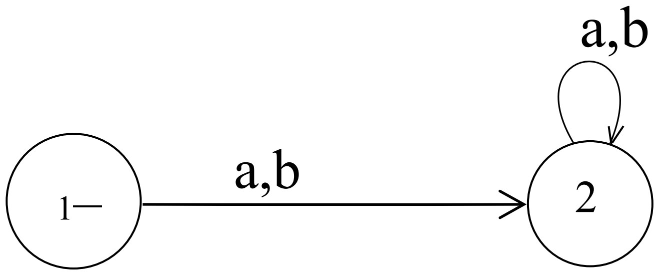
- It is to be noted that two FAs are said to be equivalent, if they accept the same language, as shown in the following FAs.

Equivalent FAs Continued …

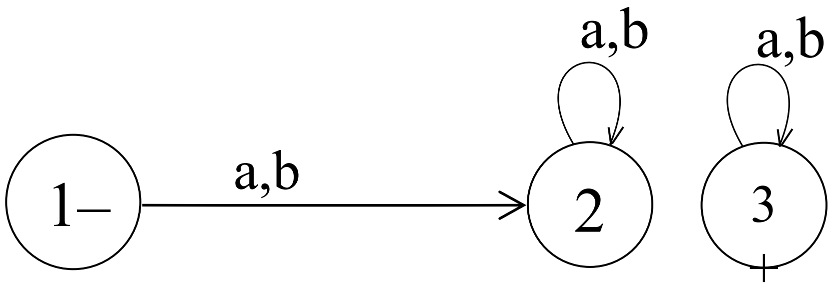
FA$_1$



FA$_2$



FA$_3$



Note (Equivalent FAs)

- FA_1 has already been discussed, while in FA_2, there is no final state and in FA_3, there is a final state but FA_3 is disconnected as the states 2 and 3 are disconnected.

  It may also be noted that the language of strings accepted by FA_1, FA_2 and FA_3 is denoted by the empty set *i.e.*

  { } OR Ø

Example

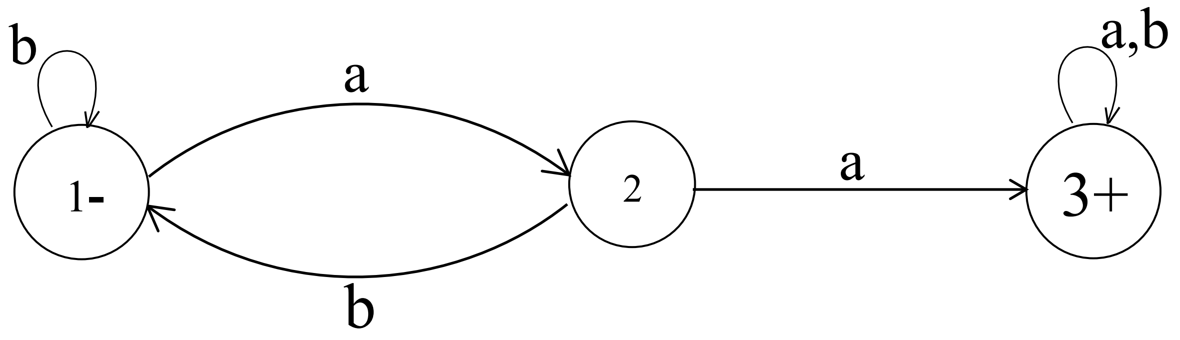
Consider the Language L of strings , defined over Σ = {a, b}, **containing double a.**

The language L may be expressed by the following regular expression

$$(a+b)^* (aa) (a+b)^*.$$

This language may be accepted by the following FA

Example Continued …



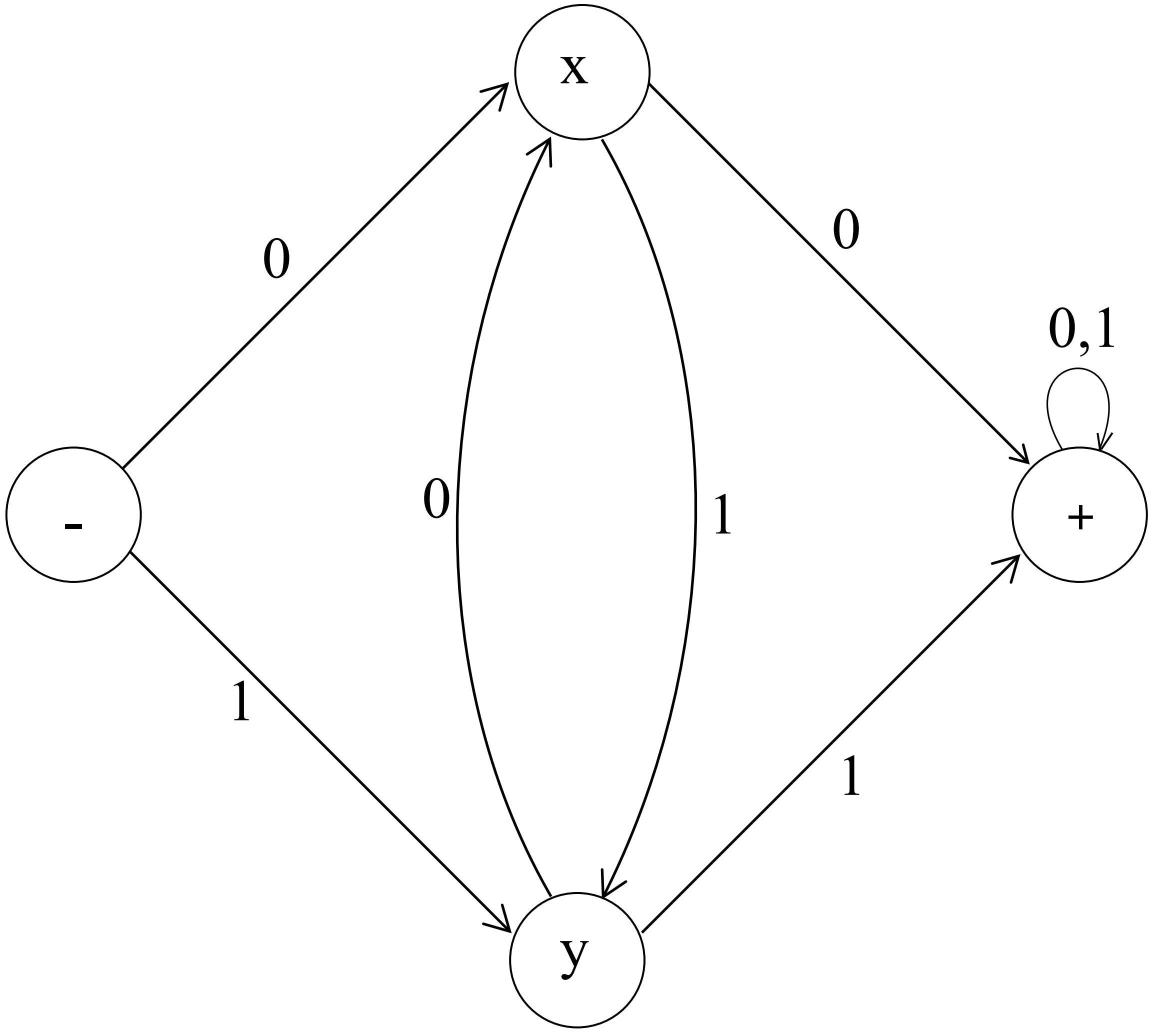
Example

Consider the language L of strings, defined over

$\Sigma=\{0, 1\}$, **having double 0’s or double 1’s,** The language L may be expressed by the regular expression
$(0+1)^* (00 + 11) (0+1)^*$

This language may be accepted by the following FA

Example Continued …



Example

Consider the language L of strings, defined over Σ={a, b}, **having triple a's or triple b's.** The language L may be expressed by RE

$$(a+b)^* (aaa + bbb) (a+b)^*$$

This language may be accepted by the following FA

Example Continued …

$$\begin{tikzcd}
& 2 \arrow[rr, "a"] \arrow[dd, bend left, "b"] & & 4 \arrow[dr, "a"] \arrow[ddll, "b"'] & \\
1- \arrow[ur, "a"] \arrow[dr, "b"'] & & & & 6+ \arrow[loop above, "{a,b}"] \\
& 3 \arrow[uu, bend left, "a"] \arrow[rr, "b"'] & & 5 \arrow[uull, "a"] \arrow[ur, "b"'] &
\end{tikzcd}$$

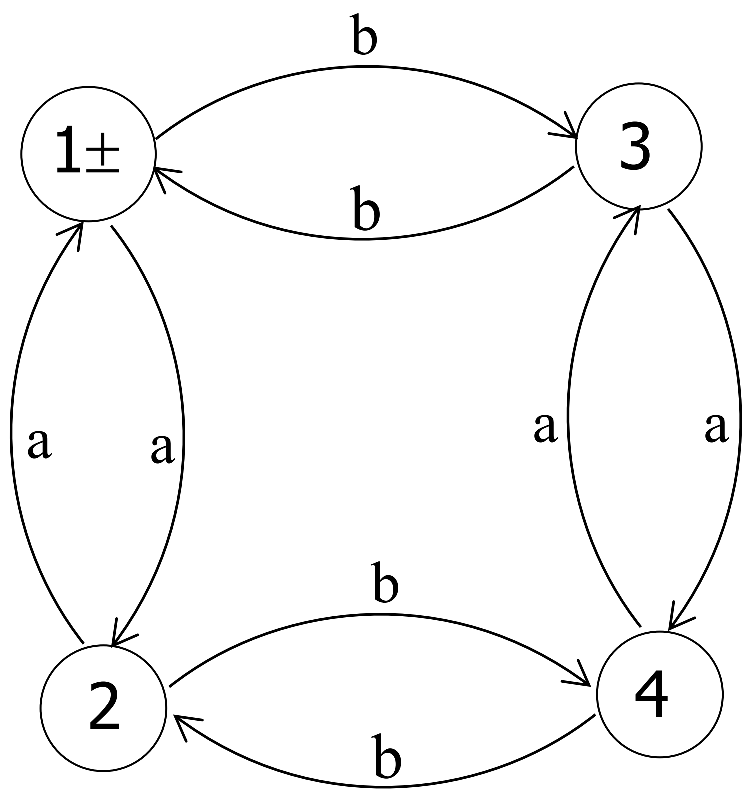
Example

- Consider the **EVEN-EVEN** language, defined over Σ={a, b}. As discussed earlier that **EVEN-EVEN** language can be expressed by the regular expression

$$(aa+bb+(ab+ba)(aa+bb)^*(ab+ba))^*$$

EVEN-EVEN language may be accepted by the following FA

Example Continued …



Summing Up

- Language of strings beginning with and ending in different letters, Accepting all strings, accepting non-empty strings, accepting no string, containing double a’s, having double 0’s or double 1’s, containing triple a’s or triple b’s, EVEN-EVEN

Non-Deterministic Finite Automata

- Different notations of transition diagrams, languages of strings of **even length, Odd length, starting with b, ending in a, beginning with b, not beginning with b, beginning and ending in same letters;**