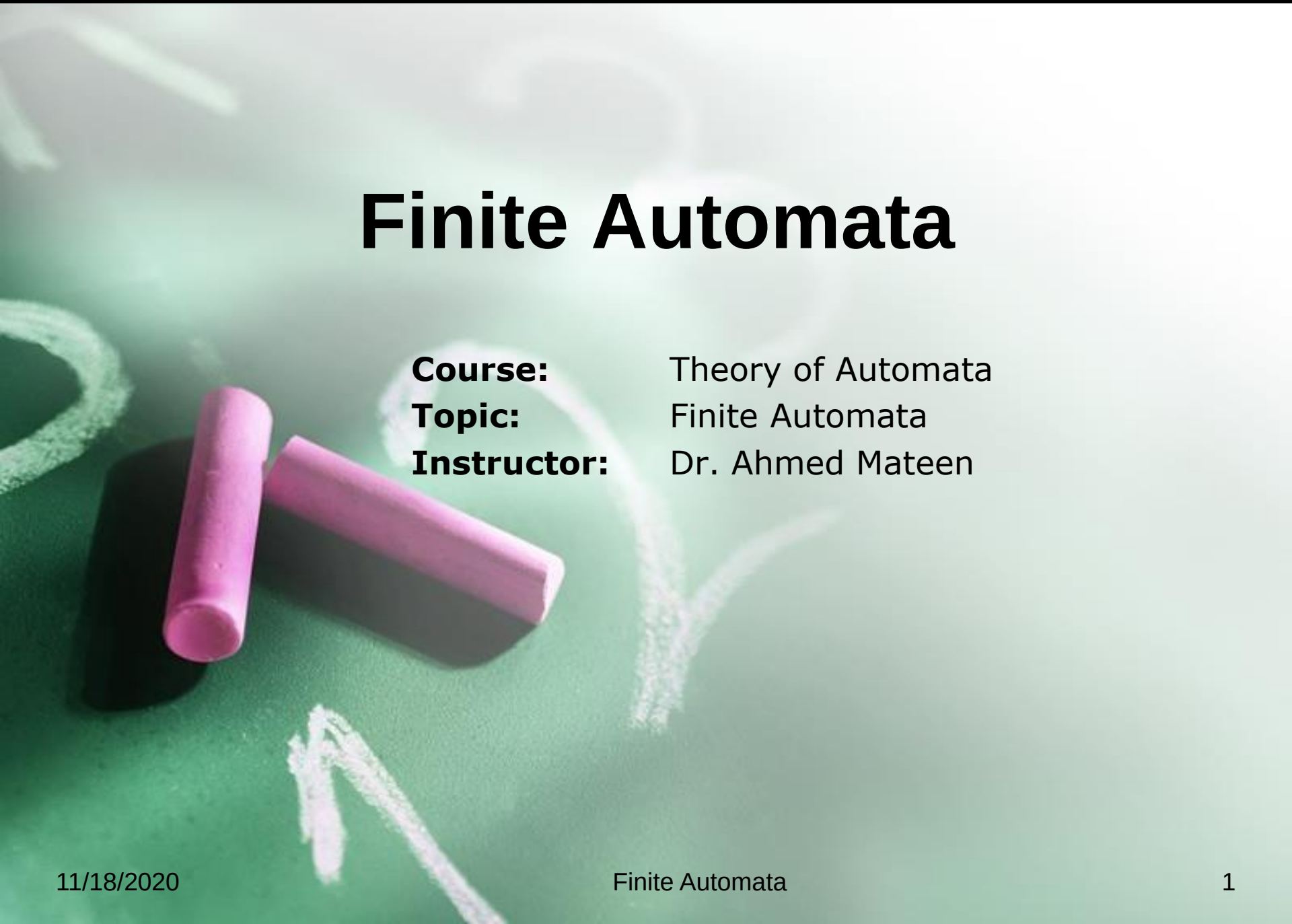


Finite Automata

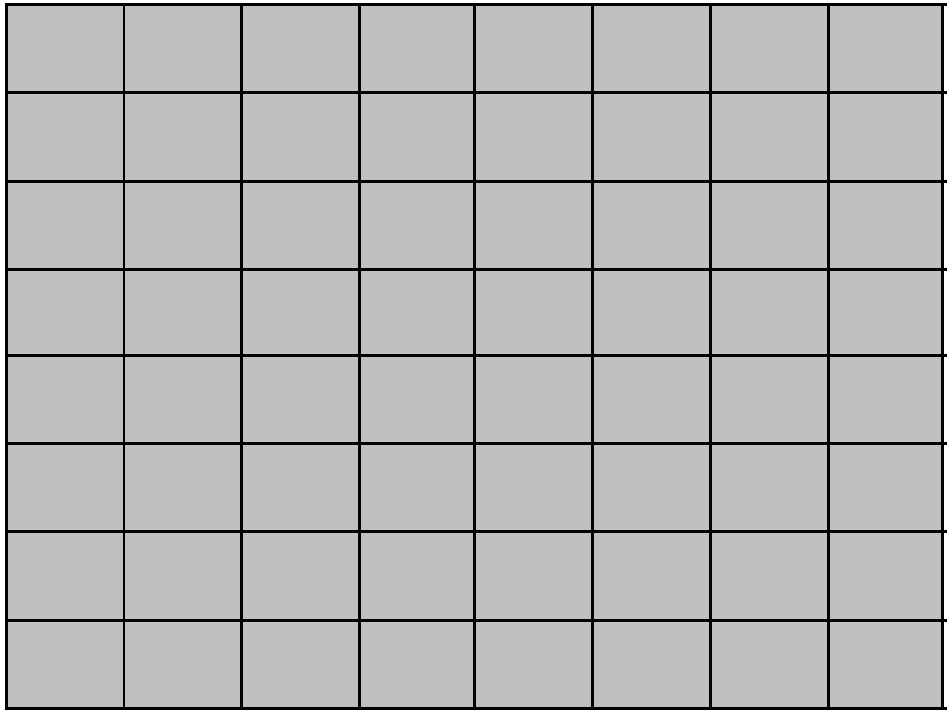
Course: Theory of Automata
Topic: Finite Automata
Instructor: Dr. Ahmed Mateen

Background

- Why need formal definitions of language
  - Define a precise, unambiguous and uniform interpretation
  - Communication with machines
- Formal Language notation/definition
  - Regular Expression
    - Tell how to generate words of that language
    - Tell which words belong to this language
    - Can this be automated

Introduction to Finite Automata

- Consider the following game board that consists of 64 boxes



Introduction to Finite Automata

- We are supposing that there are some pieces of paper. Some are of white colour while others are of black colour.
- The number of pieces of papers are 64 or less.
- The possible arrangements under which these pieces of paper can be placed are finite.
  - Because number of boxes are finite, number of pieces are finite. If we will keep placing the arrangements then there will be one stage where the previous stage will be repeated.

Introduction to Finite Automata

- There are pair of dice that can generate the numbers 2,3,4…..12
- To start the game, one of the arrangements is supposed to be initial arrangement.
- For each number generated, a unique arrangement is associated among the possible arrangements.
  - Means if we are at certain arrangement and there comes a number then what arrangement should be followed.

Introduction to Finite Automata

- It shows that the total number of transition rules are finite.
- One and more arrangements can be supposed to be the winning arrangement.
- It can be observed that the winning of the game depends on the sequence in which the numbers are generated.
- This structure of game can be considered to be a finite automaton.

Finite Automata

- Language Recognizers
- Machines embedded with grammatical rules that recognize a language
- Automated language recognition
- REs define a language and FAs accept (or reject) them
- FAs serve two purposes
  - Implicit language definition
  - Recognition

Finite Automata

- Visual notations (abstract machines)
- Sort of graphs consisting of nodes called states and edges called transitions
- States serve as memory locations that keep a track of last character read
- Transitions define where to go on reading a particular character

Finite Automaton

- A finite automaton is a collection of three things
  - A finite set of states
    - Exactly one initial state (start state)
    - One or more (may be none) final states that mark the acceptance of a word
    - Intermediate states that are neither start not final states
    - Finite set of input letters (Σ) from which input strings are formed
    - A finite set of transitions that tell for each state and for each letter of the input alphabet which state to go to next
  - An alphabet $\sum$ of possible input letters

Finite Automaton

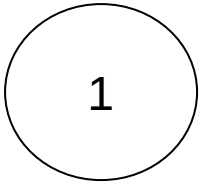
- The start state marks the beginning of reading every input
- Reading a character triggers a transition from that state which may transfer control to some other state and the reading mechanism advances to next character and the process continues
- When the input terminates, if the control is left with a final or accepting state, the input string is accepted otherwise it is rejected and the FA resets control to the initial state for next input

Finite Automaton

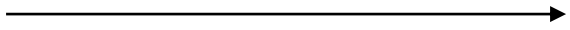
- The state to go to next on reading a letter of the input string is determined automatically and deterministically by the transitions and is fixed (for that particular state and input character)
- The transitions are fired automatically
- A single character is consumed for each transition

Finite Automaton

- Visual representations
  - States represented by circles labeled to identify each distinctly

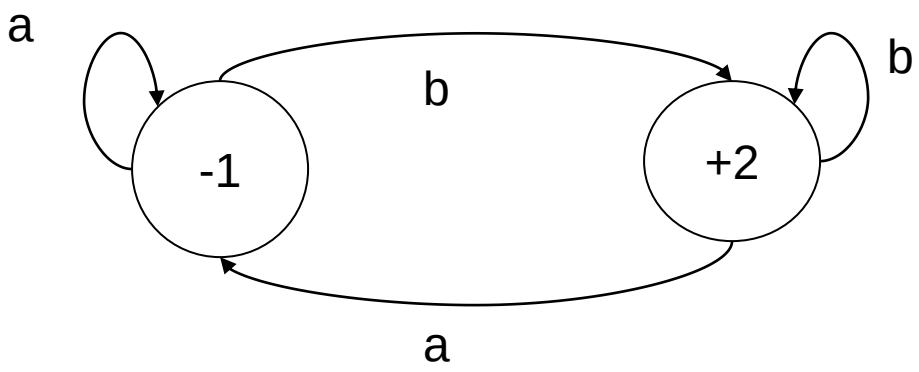


    - Initial (**-** sign) and Final states (**+** sign)
  - Transitions
    - Directed edges labeled with the characters of $\sum$



Finite Automata

- Example
  - Language of all words that end at *b*



Characteristics of FA

- Every FA must have exactly one start state
  - There may be multiple or may be no final states
  - In the latter case the FA doesn't accept any language
- Only a single character is read on a state at a time
- Every state define a transition for every character in the alphabet set or alternatively
- Every state has exactly as many outgoing transitions as the number of characters in $\sum$

Characteristics of FA

- Each transition labeled with a distinct letter from Σ
  - No duplicate edges
  - No missing edges
- An FA is built for a particular language and recognizes only that language
- It should accept all valid words of the language
  - Not even a single valid word should be rejected.
- It should reject all invalid words of the language
  - Not even a single valid word should be rejected.

Finite Automaton

- Example:
  - **Alphabet**
    - Σ = {a, b}
  - **States**
    - x, y, z where x is an initial state and z is final state.
  - Transitions
    - At state x reading a go to state z
    - At state x reading b go to state y
    - At state y reading a, b go to state y
    - At state z reading a, b go to state z

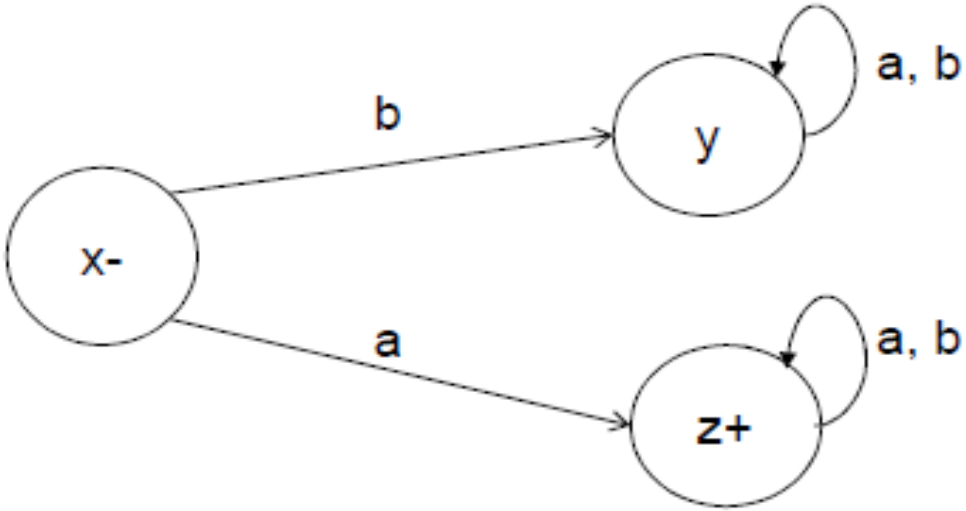
Finite Automaton

- These transitions can be expressed by the following table called *Transition Table:*

Old states	New States	
	Reading a	Reading b
x-	z	y
y	y	y
z+	z	z

Finite Automaton

- It may be noted that the information on an FA, given in the previous table, can also be depicted by the following diagram called the *Transition Diagram.*



Finite Automaton

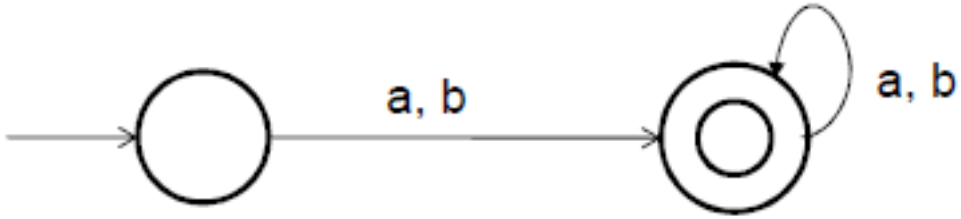
- The previous transition diagram is an FA *accepting the language of strings, defined over* Σ={a, b}, starting with a.
- ***Accepting the language*** – *any string which* starts from the initial state and ends at the final state while following the path is referred to as accepted string. The collection of all accepted strings will be named with specific language which can be associated with the FA.

Finite Automaton

- It may be noted that this language may be expressed by the regular expression:
  - a (a + b)*
- It may be noted that to indicate the initial state, an arrow head can also be placed before that state and that the final state with double circles.

Finite Automaton

- It is also to be noted that while expressing an FA by its transition diagram, the labels of the states are not necessary.



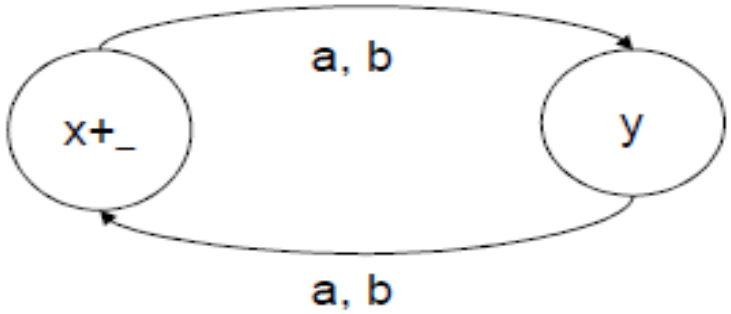
Finite Automaton

- Example:
  - Σ = {a, b}
  - States : x, y where x is both an initial and final state.
  - Transitions
    - At state x reading a or b go to state y
    - At state y reading a or b go to state x
- Transition Table:

Old states	New States	
	Reading a	Reading b
x +_	y	y
y	x	x

Finite Automaton

- Transition Diagram:



- Noted: There are 2 possibilities i.e. either the string will end at final state or it may end at non-final state.
  - If it ends at final state, it will be treated as accepted string otherwise rejected string

Finite Automaton

- Regular Expression:
  - Null String
    - We do no represent null string in case of transition diagram and we assume that null string will not take us to any other state.
    - What about null string? Accepted Or Rejected?
    - In this case, our initial state is also a final state, therefore, null string is an accepted string.
  - Single Length String
    - Neither a, nor b is accepted by the FA

Finite Automaton

- Double Length String:
  - If we think of aa, ab, ba, bb they will be accepted
- String of Length 3
  - Strings of length 3 will start from x, will visit y, then visit x and will end at y. y is not a final state therefore, any string of length 3 will be rejected.
- String of Length 4
  - Strings of length 4 will start from x, will visit y, then visit x, will visit y and will end at x. x is a final state therefore, any string of length 4 will be accepted.
  - Shows that all strings with even length will be accepted and with odd length will be rejected.

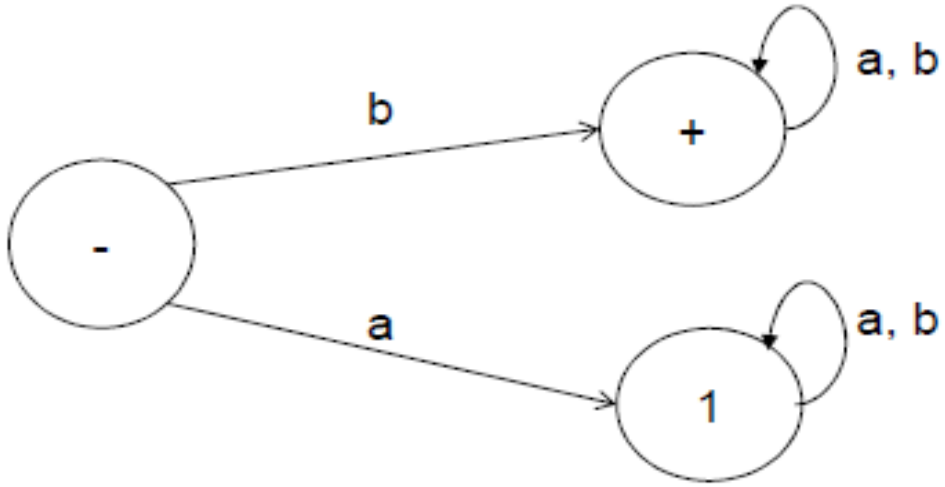
Finite Automaton

  - The previous 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: move systematically in order to find out the sequence or pattern while associating any language with FA.

Finite Automaton

- What we have discussed is, given an FA can we identify the language accepted that FA.
- Given a language, can we build FA?
- Yes/No?
- Example:
  - Consider the language L of strings, defined over Σ = {a, b}, starting with b. The language L may be expressed by RE b(a+b)*, may be accepted by the following FA.

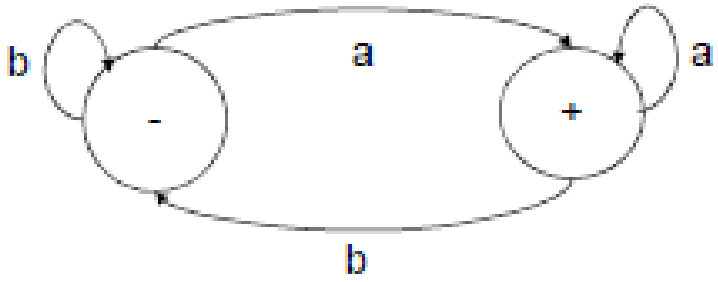
Finite Automaton



- There must be two way check:
  - Every string belonging to the given language should be accepted by the given FA and
  - Every string accepted by the FA should belong to the given language.

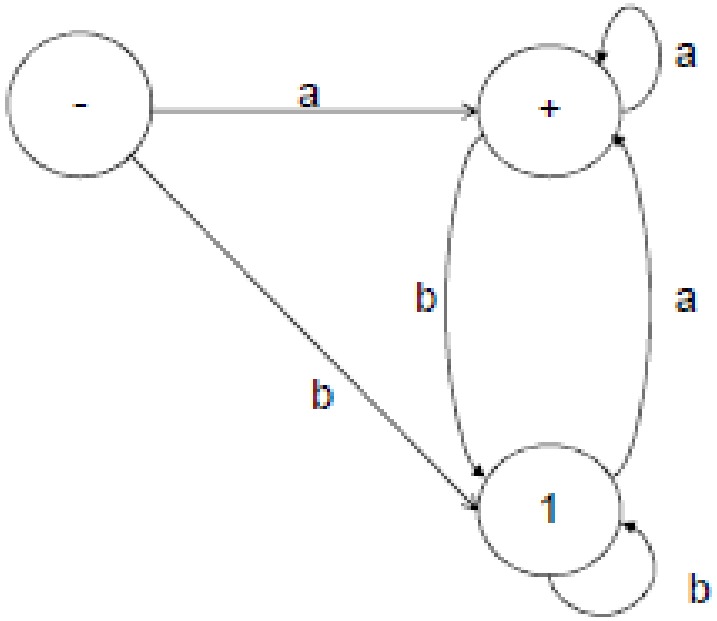
Finite Automaton

- Example:
  - Consider the language L of strings, defined over Σ = {a, b}, ending in a. The language L may be expressed by RE (a+b)*a, ***may be accepted*** by the following FA.



  - Try to reduce the number of states in FA

Finite Automaton



– FA for strings that ends in a.

Finite Automaton

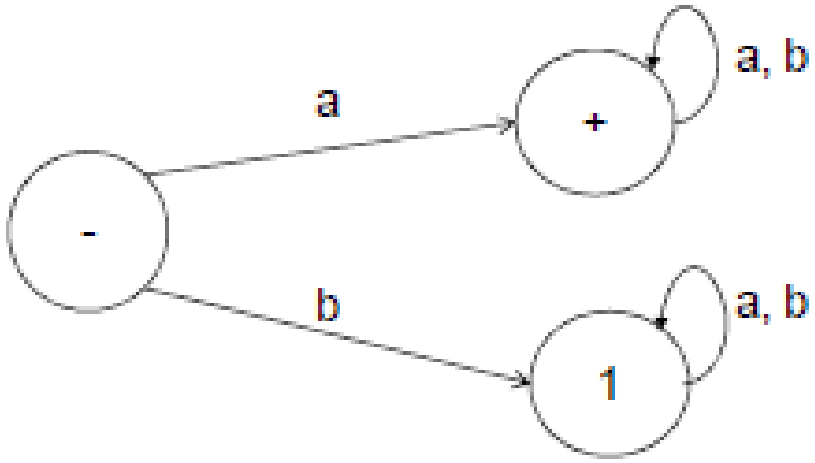
- Note:
  - It may be noted that corresponding to a given language there may be more than one FA accepting that language.
  - But for a given FA there is a unique language accepted by that FA.

Finite Automaton

- Note:
  - It is to be noted that given the languages L1 and L2, where
  - L1 = the language of strings, defined over Σ = {a, b}, beginning with a.
  - L2 = the language of strings, defined over Σ = {a, b}, not beginning with b.
  - The null string does not belong to L1, while it does belong to L2. This fact may be depicted by the corresponding transition diagrams of L1 and L2.

Finite Automaton

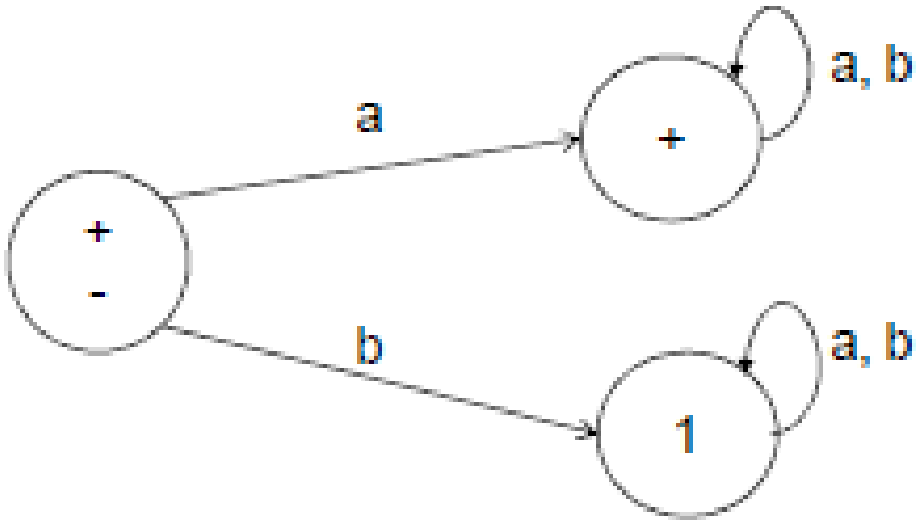
- FA corresponding to L1



  - The language L1 may be expressed by the Regular Expression a(a+b)*

Finite Automaton

- FA corresponding to L2

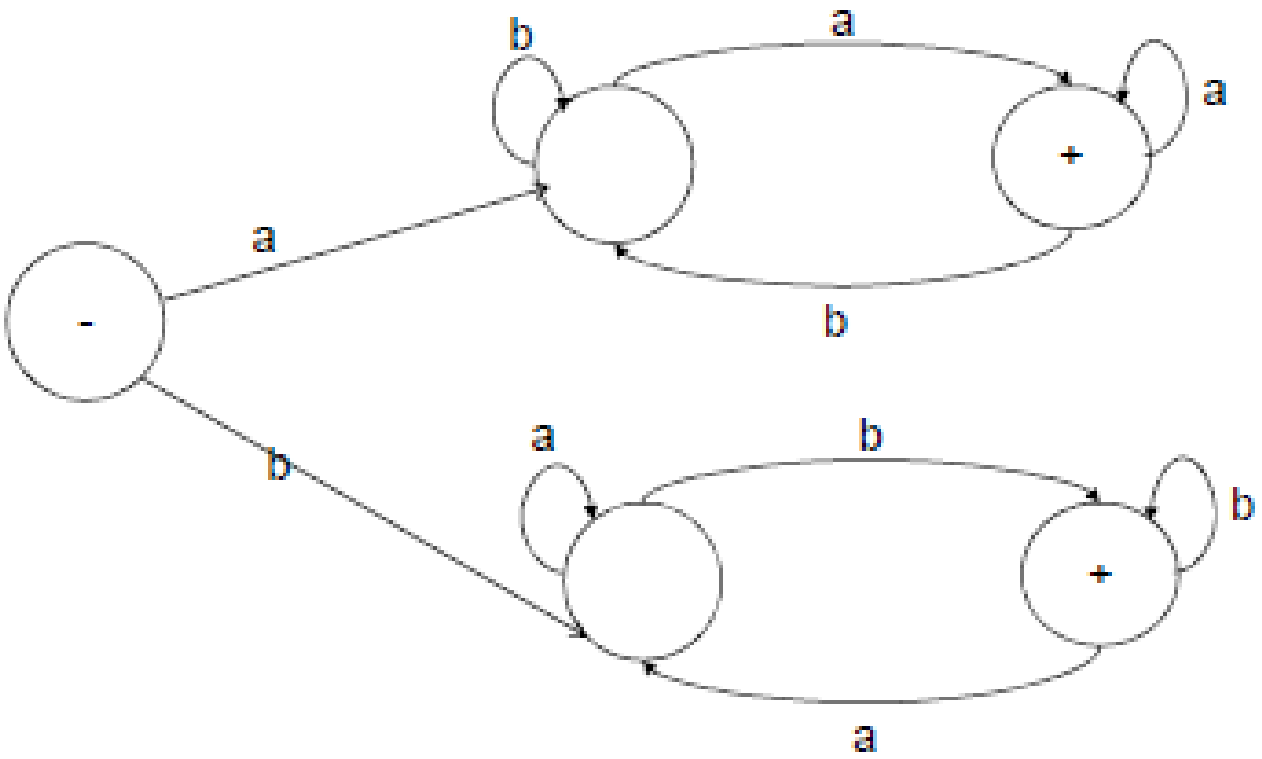


  - The language L2 may be expressed by the Regular Expression a(a+b)* + Λ

Finite Automaton

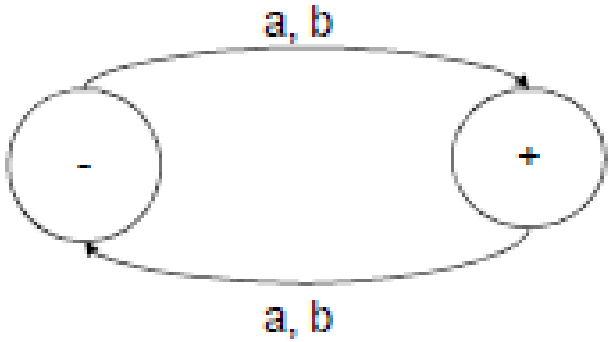
- Example:
  - Consider the language L of strings of ***length two or more,*** *defined over Σ = {a, b},* ***beginning with and ending in same letters.***
  - The language L may be expressed by Regular Expression a(a+b)*a + b(a+b)*b
  - It is to be noted that if the condition on the length of string is not imposed on 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

Finite Automaton



Finite Automaton

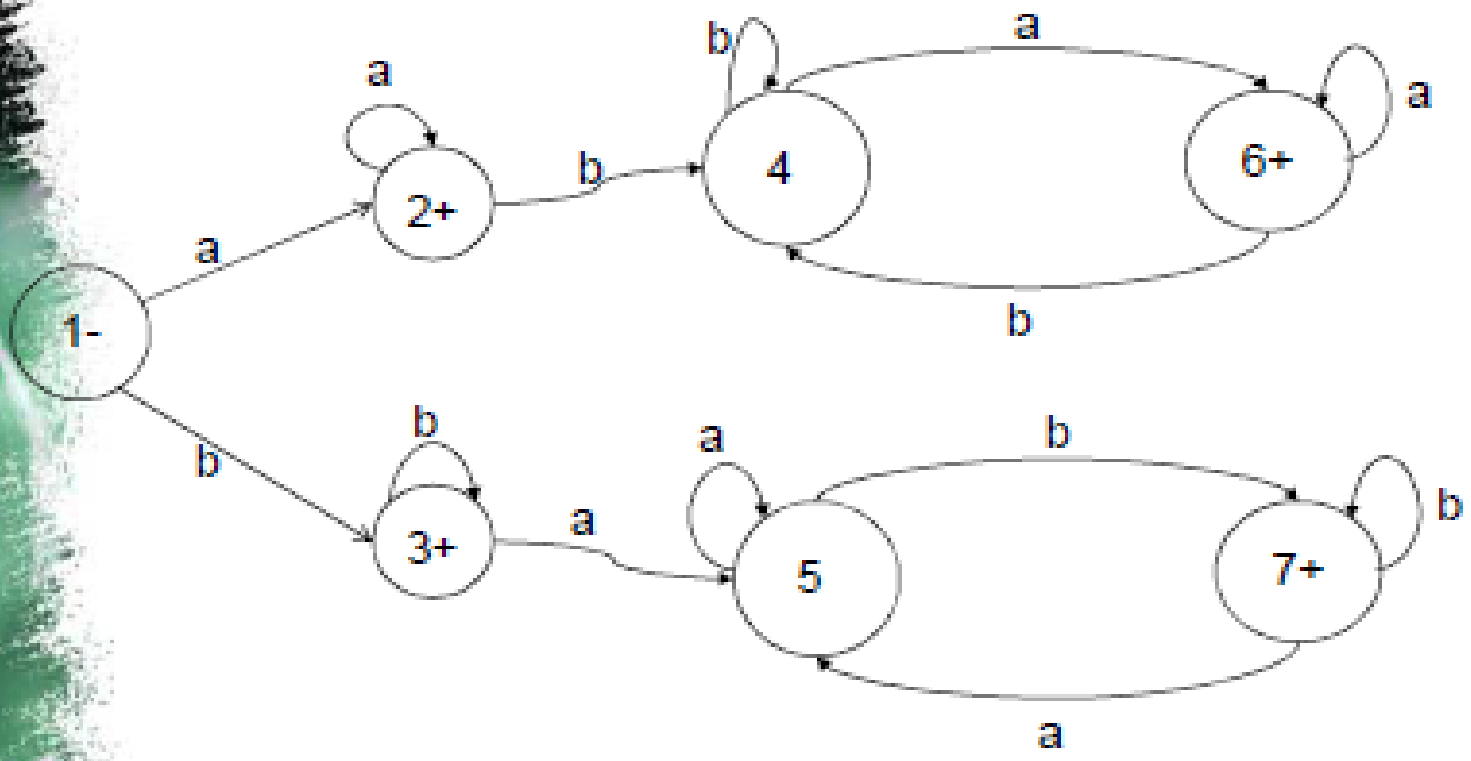
- **Task –** Build an FA for the language L of strings, defined over Σ = {a, b}, odd length.
- The language L may be expressed by RE (a+b)((a+b)(a+b))*
- This language may be accepted by the following FA



Finite Automaton

- **Task:** Build an FA accepting the language L of strings, defined over Σ = {a, b}, ***beginning with and ending in same letters.***
  - Hint: letter a and letter b will be included in the string
- The language L may be expressed by the following RE (a+b)+a(a+b)*a+b(a+b)*b
- This L may be accepted by the following FA

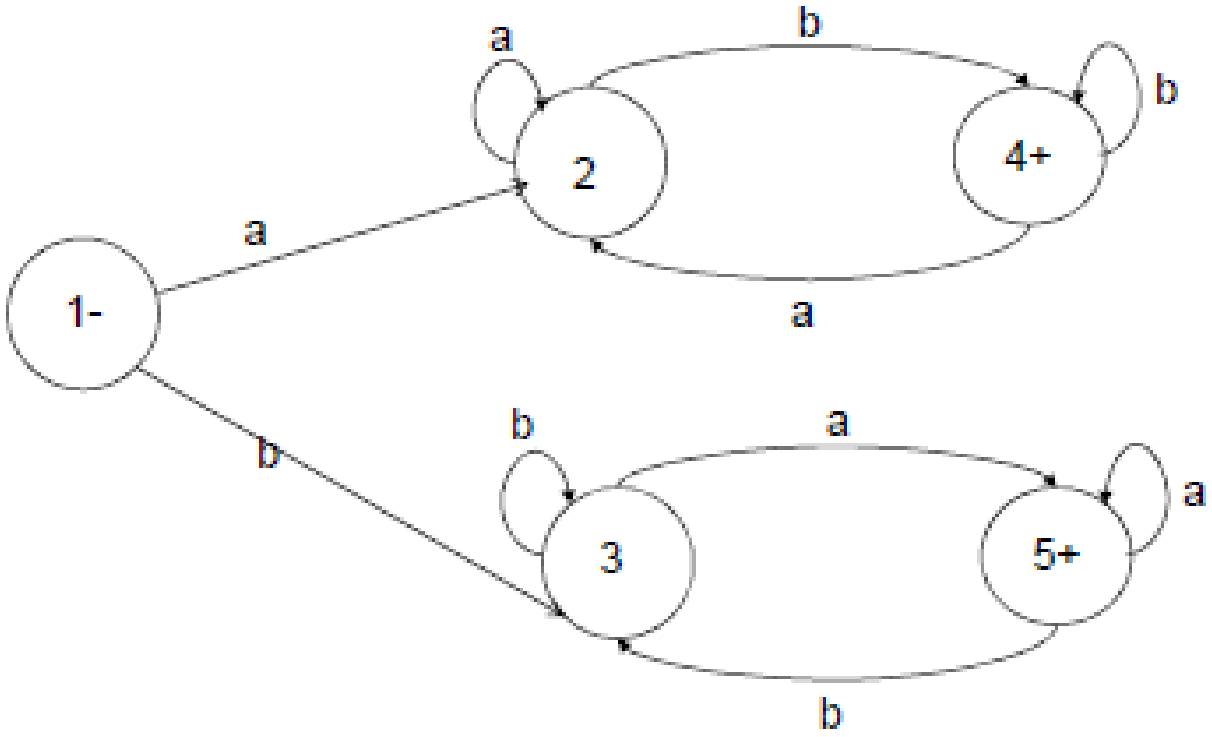
Finite Automaton



Finite Automaton

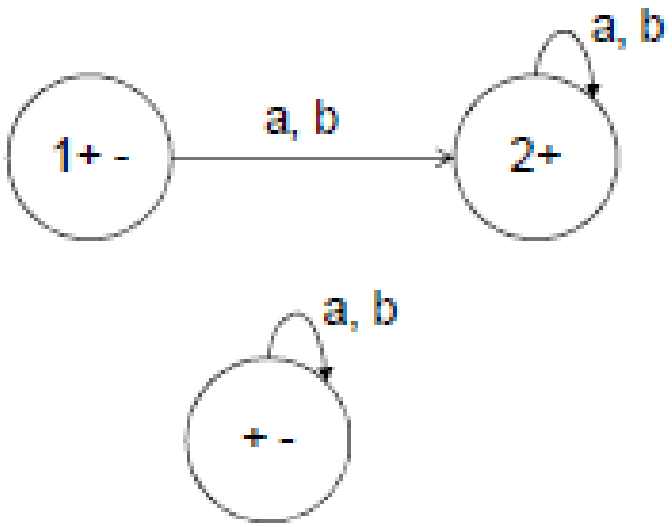
- Example:
  - Consider the language L of strings, defined over Σ = {a, b}, ***beginning with and ending in different letters.***
  - The language L may be expressed by Regular Expression a(a+b)*b + b(a+b)*a
  - This language L may be accepted by the following FA

Finite Automaton



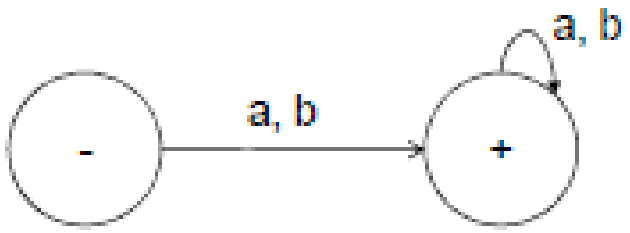
Finite Automaton

- Example:
  - Consider the language L of strings of, defined over Σ = {a, b}, of all strings including null.
  - The language L may be accepted by the following FA



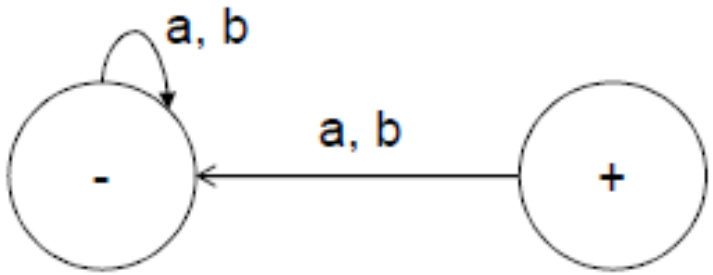
Finite Automaton

- Example:
  - Consider the language L of strings of, defined over Σ = {a, b}, of all non empty strings.
  - The language may be expressed by the following regular expression $(a+b)^+$
  - The language L may be accepted by the following FA



Finite Automaton

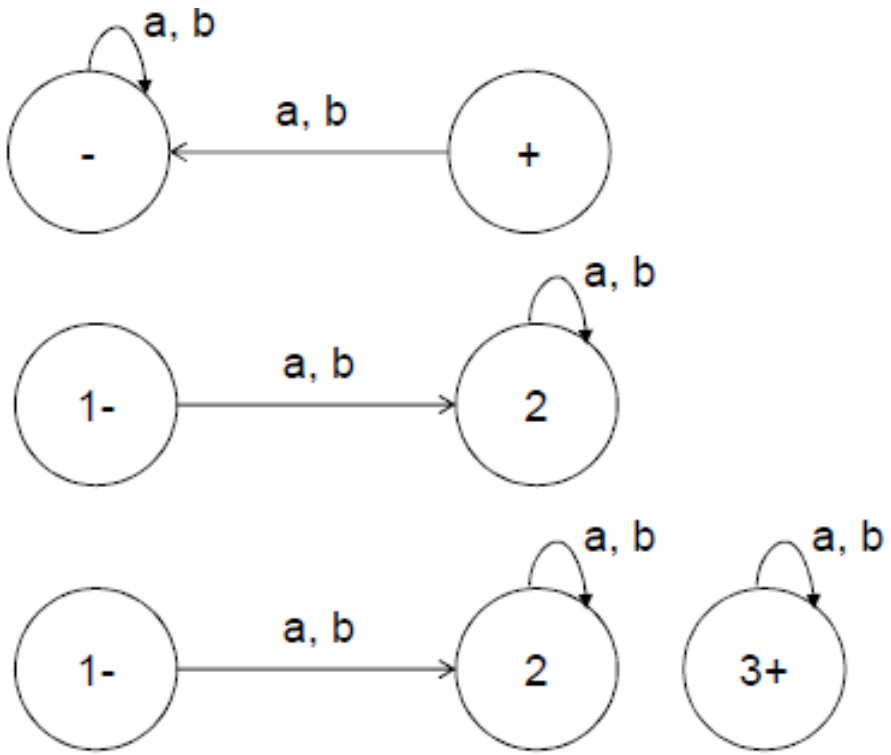
- Example:
  - Consider the following FA, defined over Σ = {a, b}.



  - 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.

Finite Automaton

- **Equivalent FAs:**
  - It is to be noted that two FAs are said to be equivalent, if they accept the same language
  - FA1
  - FA2
  - FA3

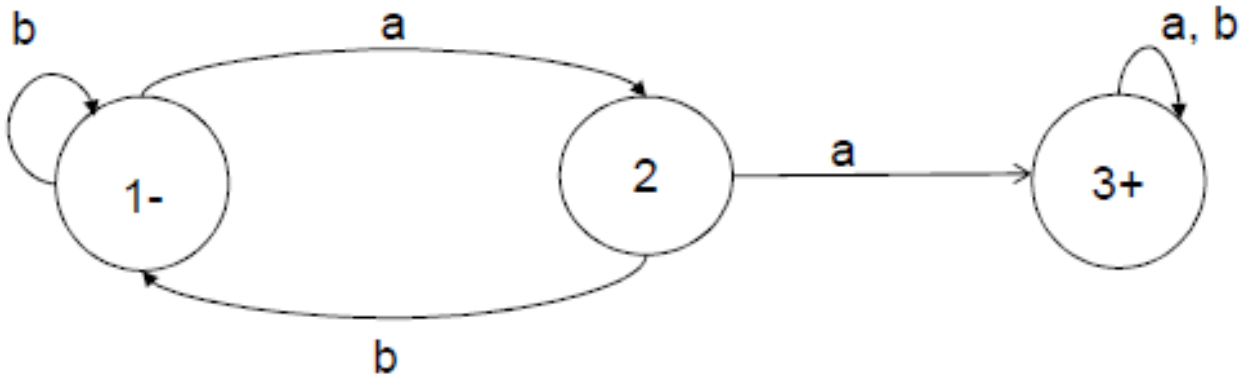


Finite Automaton

- Note:
  - FA1 has already been discussed, while in FA2, there is no final state and in FA3, there is a final state but the state is disconnected.
  - It may be noted that the language of strings accepted by FA1, FA2 and FA3 is denoted by the empty set i.e. { } or ø

Finite Automaton

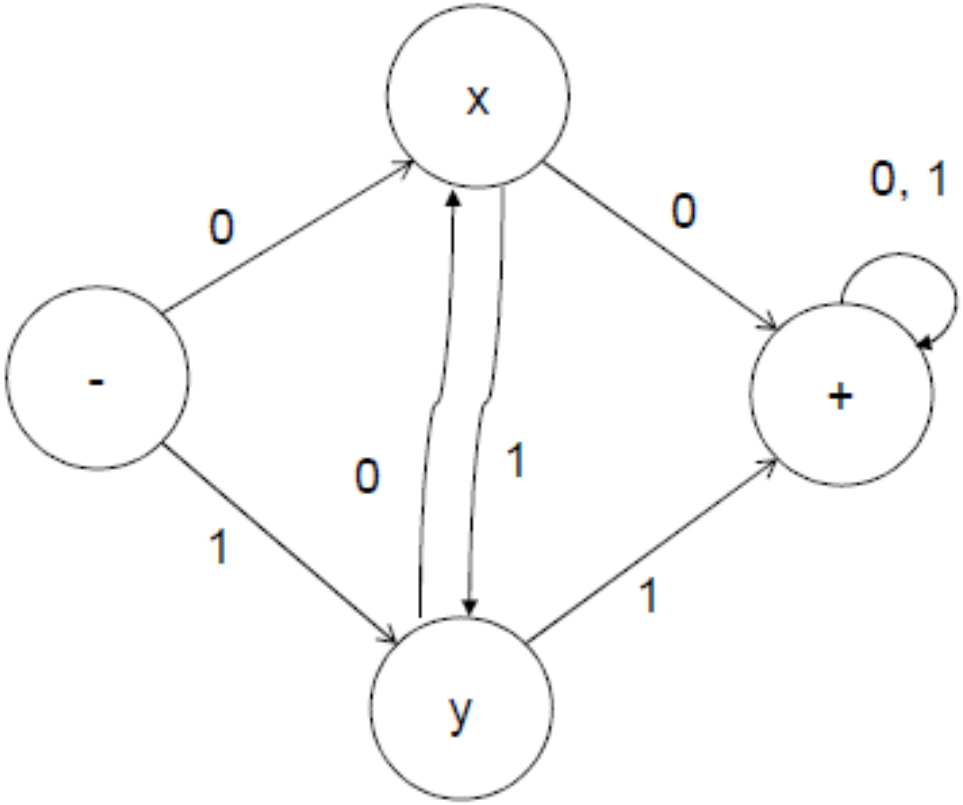
- Example:
  - Consider the language L of strings, defined over Σ = {a, b}, containing double a.
  - The language may be expressed by the following regular expression (a+b) * aa (a+b) *
  - The language L may be accepted by the following FA



Finite Automaton

- Example:
  - Consider the language L of strings, defined over Σ = {0, 1}, having double 0's or double 1's.
  - The language may be expressed by the following regular expression (0+1) * (00+11) (0+1) *
  - The language L may be accepted by the following FA

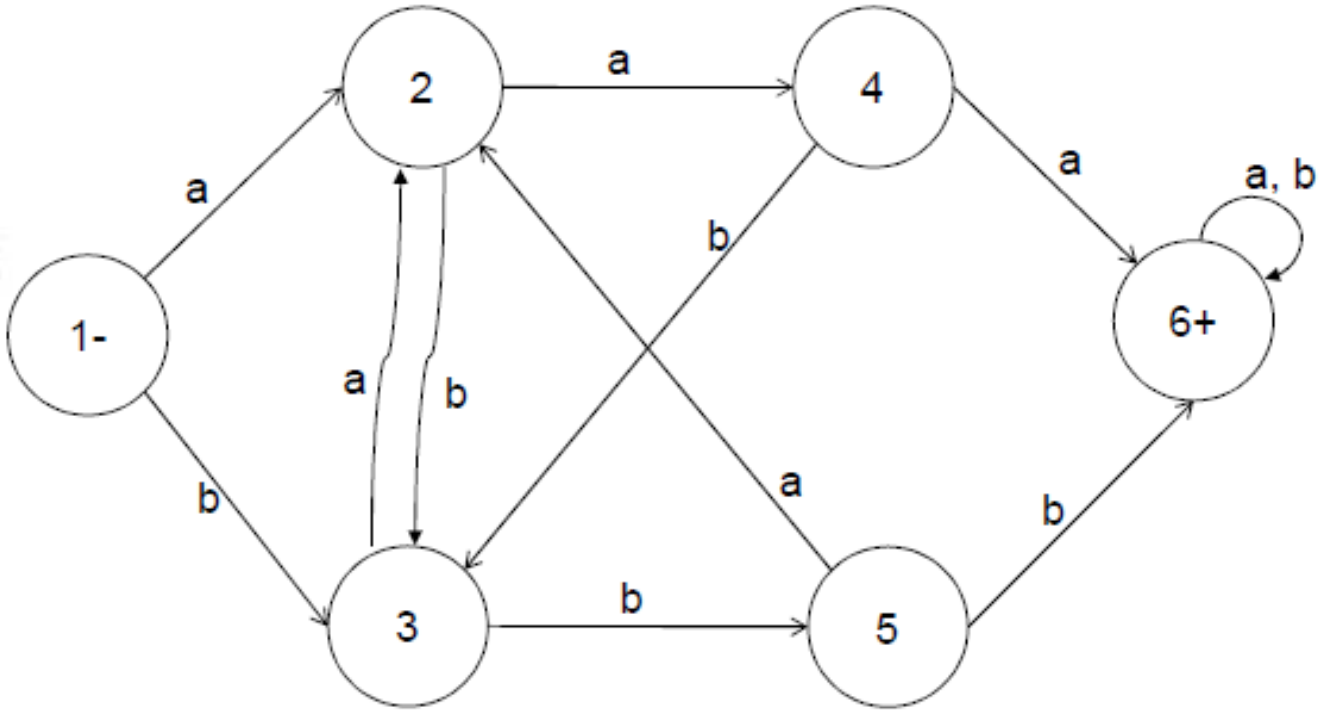
Finite Automaton



Finite Automaton

- Example:
  - Consider the language L of strings, defined over Σ = {a, b}, having triple a's or triple b's.
  - The language may be expressed by the following regular expression (a+b) * (aaa +bbb) (a+b) *
  - The language L may be accepted by the following FA

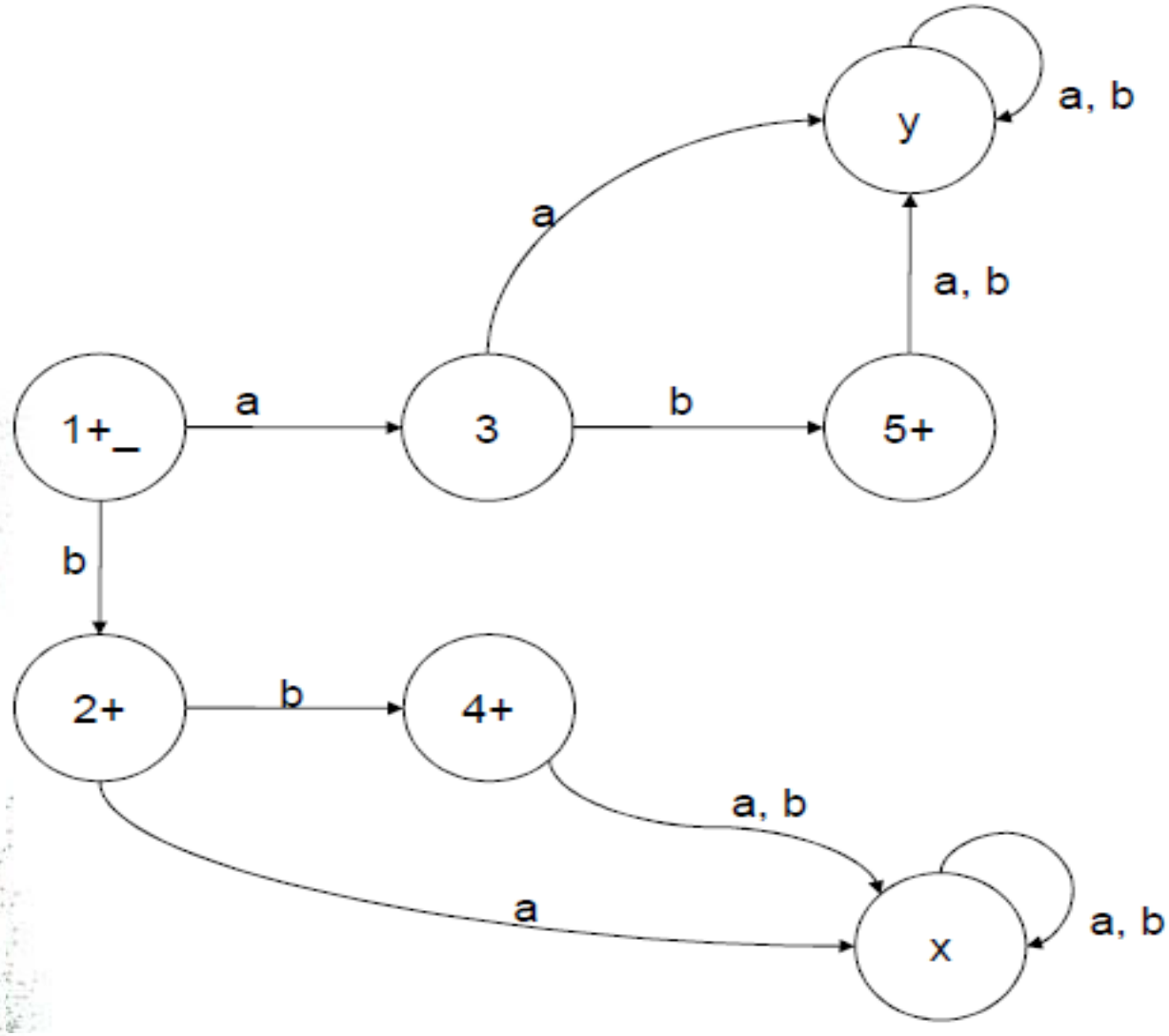
Finite Automaton



Finite Automaton

- We have discussed the infinite string languages.
- Lets move on to the FAs corresponding to the finite languages.
- Example:
  - Consider the language L = {∧, b, ab, bb} defined over Σ = {a, b} may be expressed by
  - ∧+b+ab+bb OR ∧ + b(∧+a+b)
  - The language L may be accepted by the following FA
  - The FA for finite language is difficult for beginners because, there are chances of unwanted string acceptance.

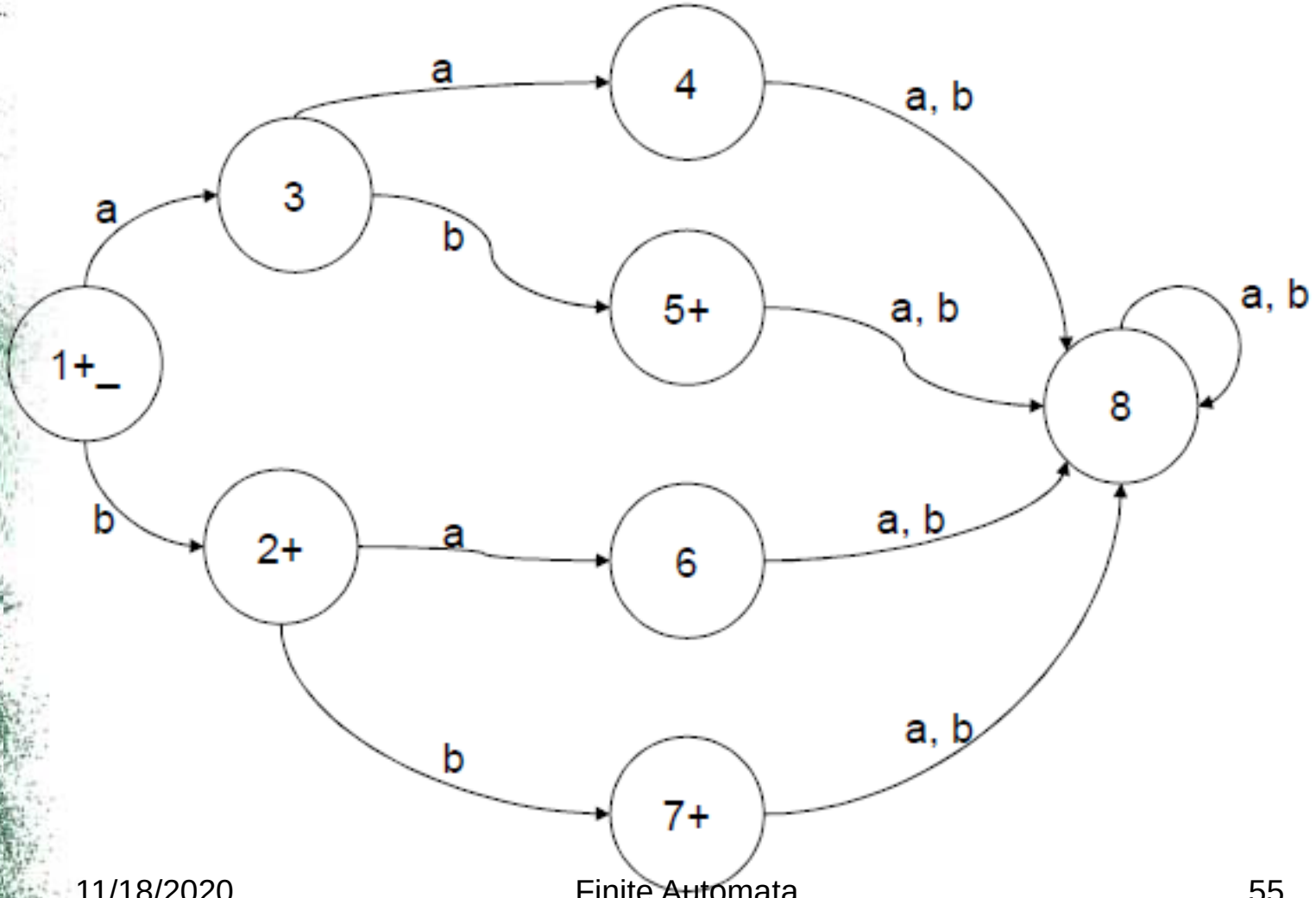
Finite Automaton



Finite Automaton

- It is to be noted that state x and y are called ***Dead States, Waste Baskets, or Davey John Lockers,*** *as the moment one enters these* states there is no way to leave it.
- It is to be noted that to build an FA accepting the language having less number of strings, the tree structure may also help in this regard, which can be observed in the following transition diagram for the language L discussed in previous example.

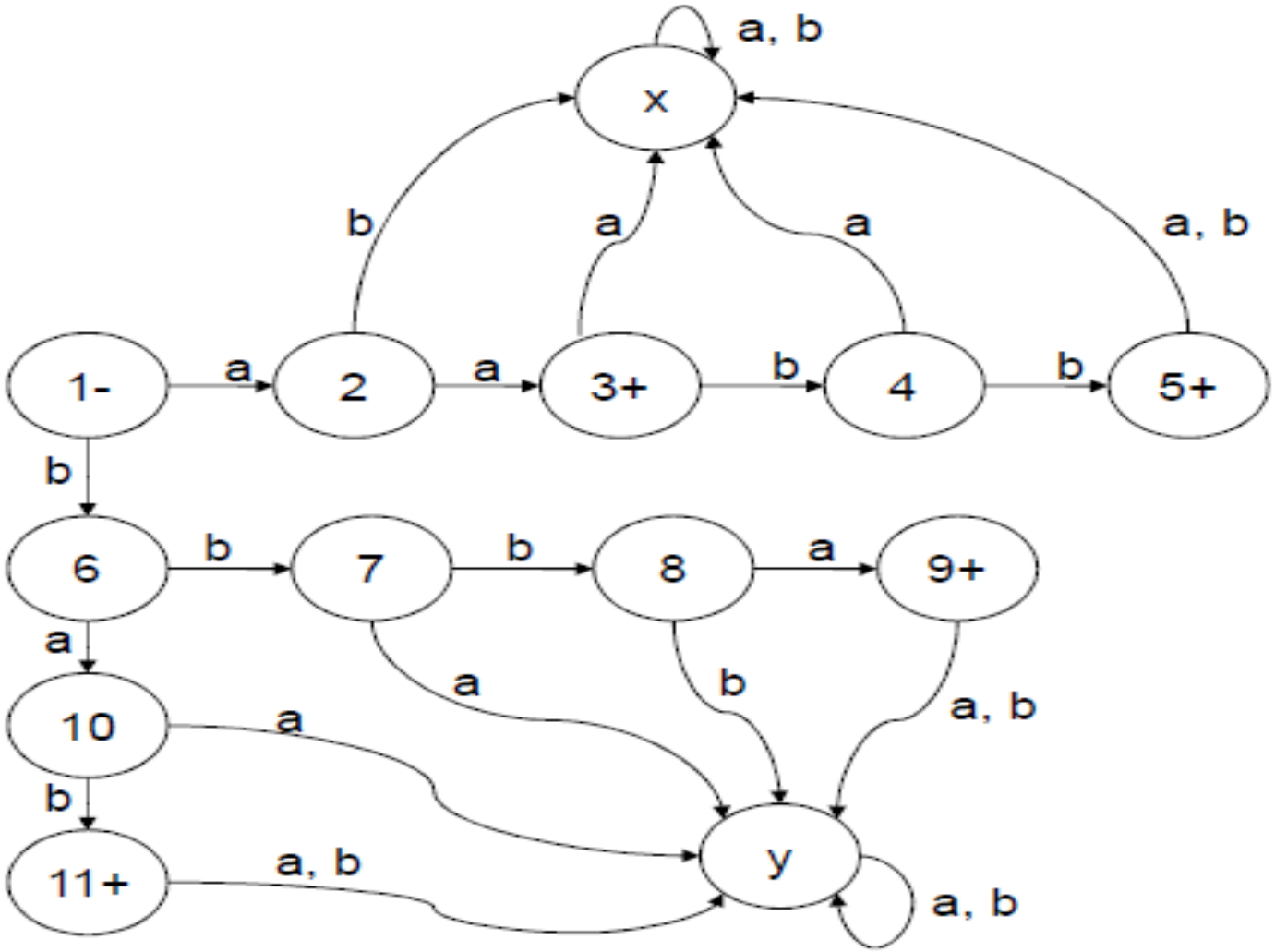
Finite Automaton



Finite Automaton

- Note:
  - In tree structure there is a separate transaction for every state.
  - There is always a loop for waste basket, If we will not make, it means we are not following the FA definition.
- Example:
  - Consider the language L = {aa, bab, aabb, bbba} defined over Σ = {a, b} may be expressed by
  - aa+bab+aabb+bbba OR aa(∧ + bb)+b(ab+bba)
  - The language L may be accepted by the following FA

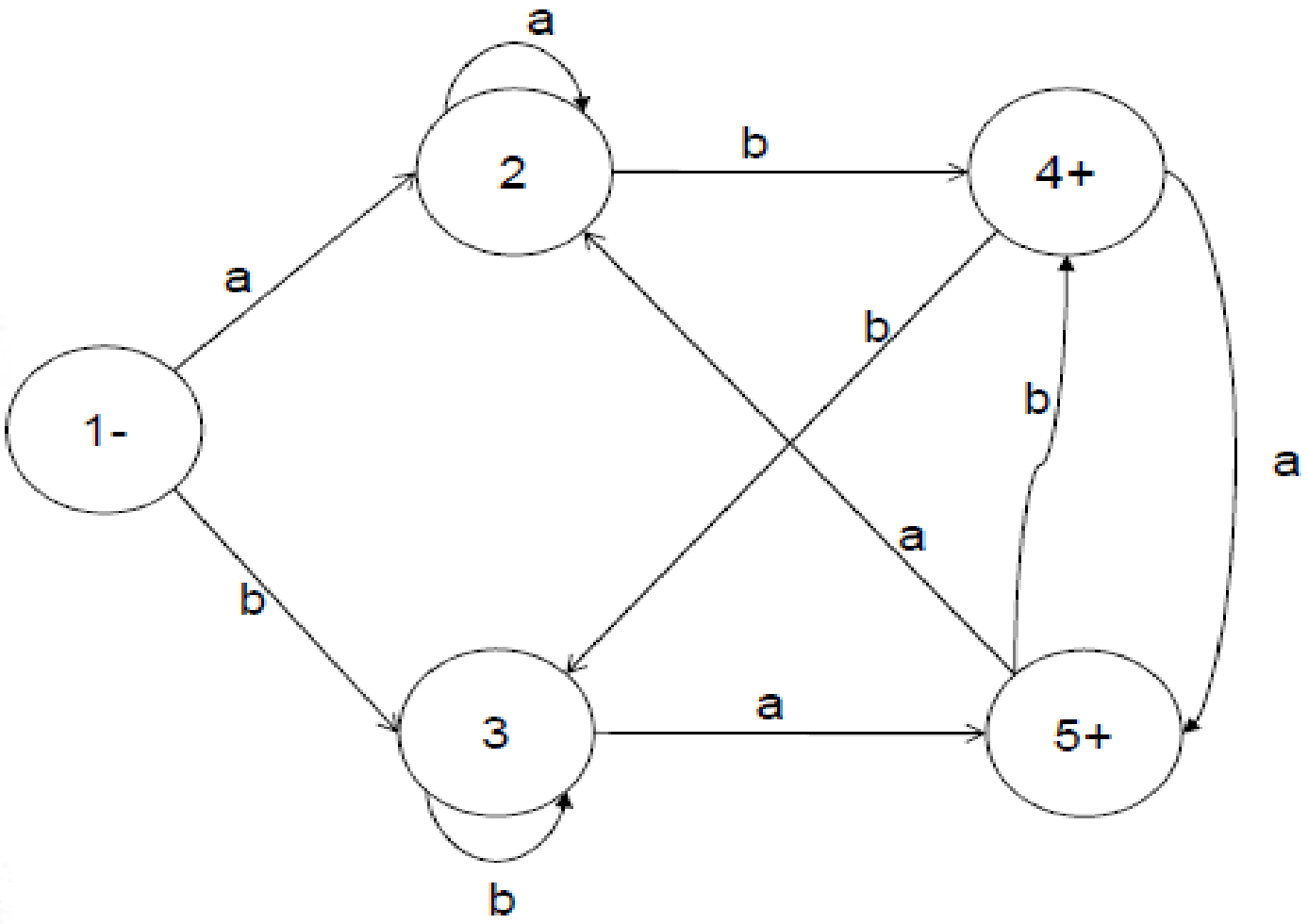
Finite Automaton



Finite Automaton

- Example:
  - Consider the language L = {w belongs to {a, b}*: length(w) >=2 and w neither ends in aa nor bb}
  - The language L may be expressed by the regular expression (a+b)*(ab+ba)
  - The language L may be accepted by the following FA

Finite Automaton



More Examples

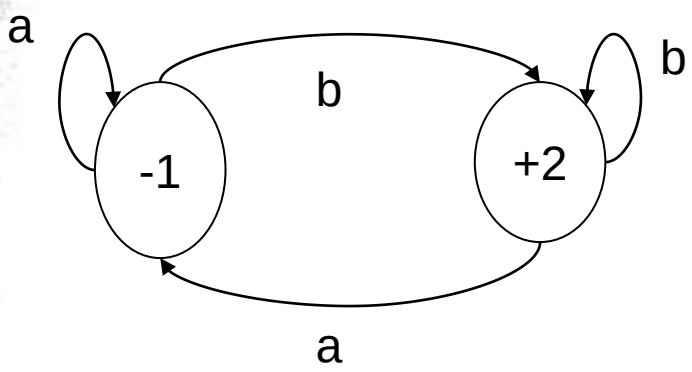
- All words that contain even number of letters
- All words ending with *ab*
- All words that start with *a*
- All words of length >=3
- All words that don’t end at *ba*
- All words that contain a triple letter either *aaa* or *bbb*

Mathematical Representations of FA

- FA = (Q, $\sum$,q_0,F,δ)
  - Q = {q_0,q_1,q_2,..q_n where n is finite}
  - $\sum$ = set of input alphabets
  - q_0 is the start states
  - F $\subseteq$ Q is the set of final states F may be φ
  - δ is the transition function
    - δ (q_i,x_j) = q_k
- Mathematical representation of FA for all words ending at *b*

Transition Tables

- Tabular representation of an FA
  - Table representing states and transitions



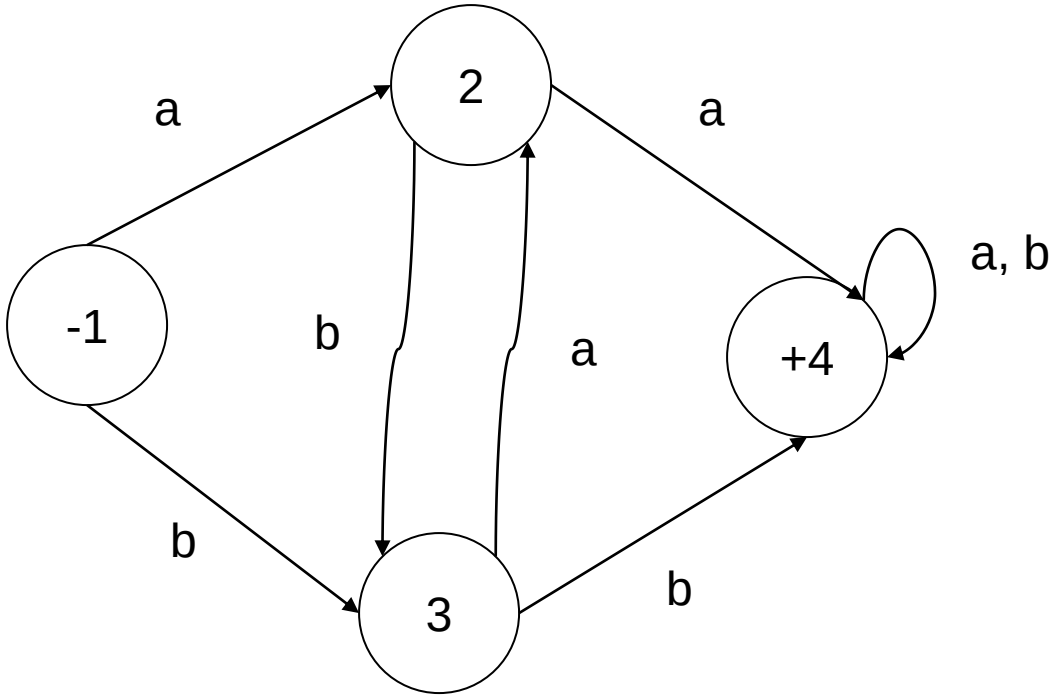
	a	b
-1	1	2
+2	1	2

Languages of FA

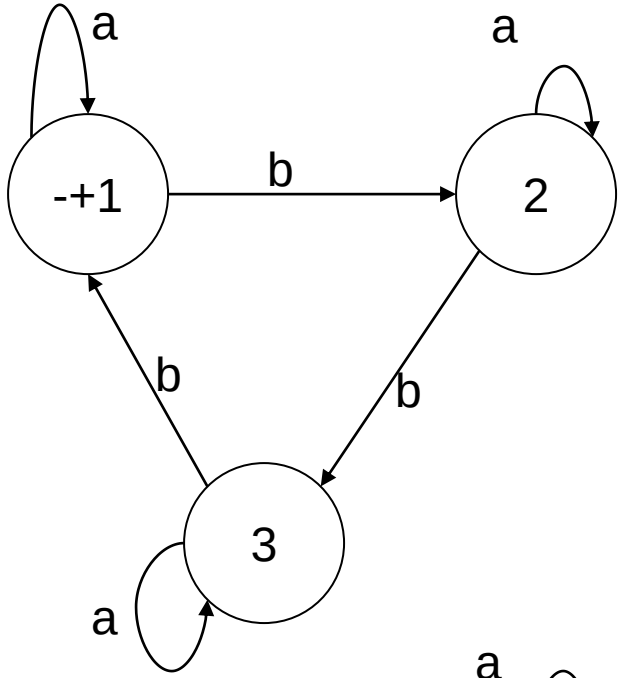
- FAs define Regular Language
- Any language that can be define by a regular expression can be recognized by an FA

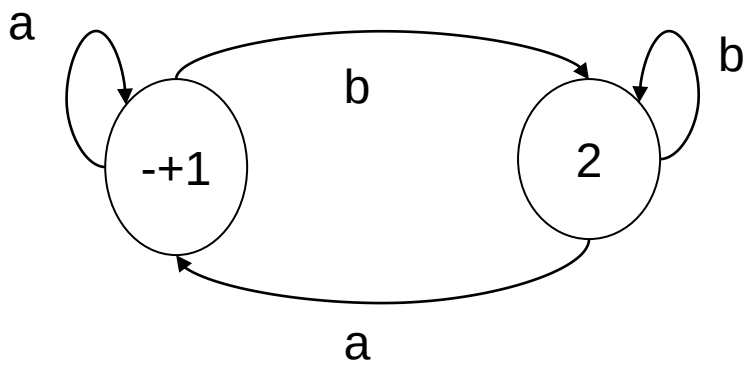
Language of FA

- What language does the following FA define



Language of FA



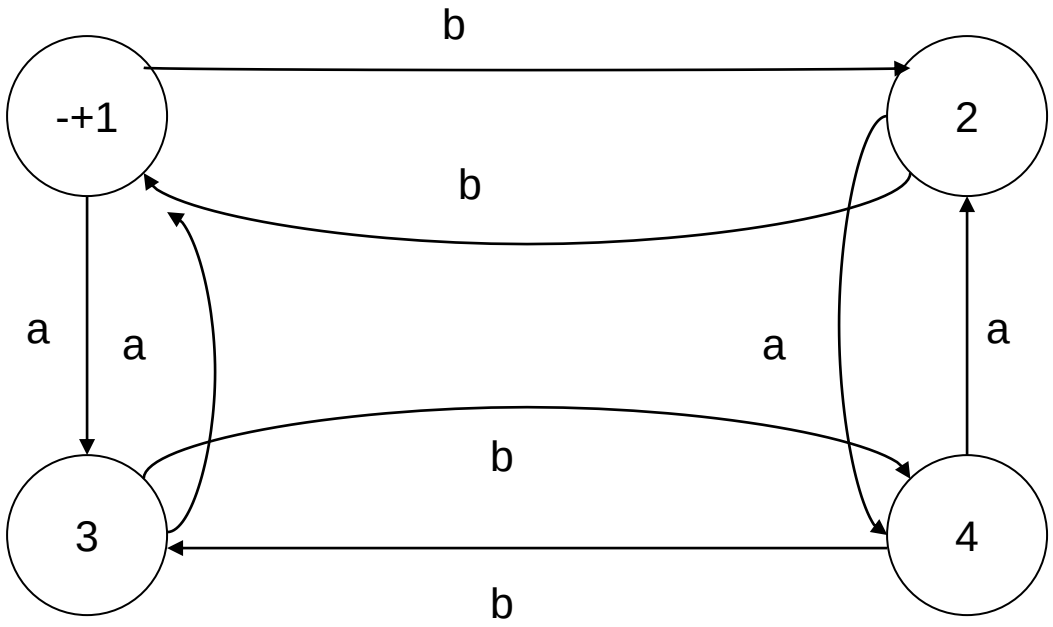


Examples

- All words that start with a double letter
- All words that start and end with a double letter
- All words that do not start with a double letter
- All words in which is the second letter is ***b***

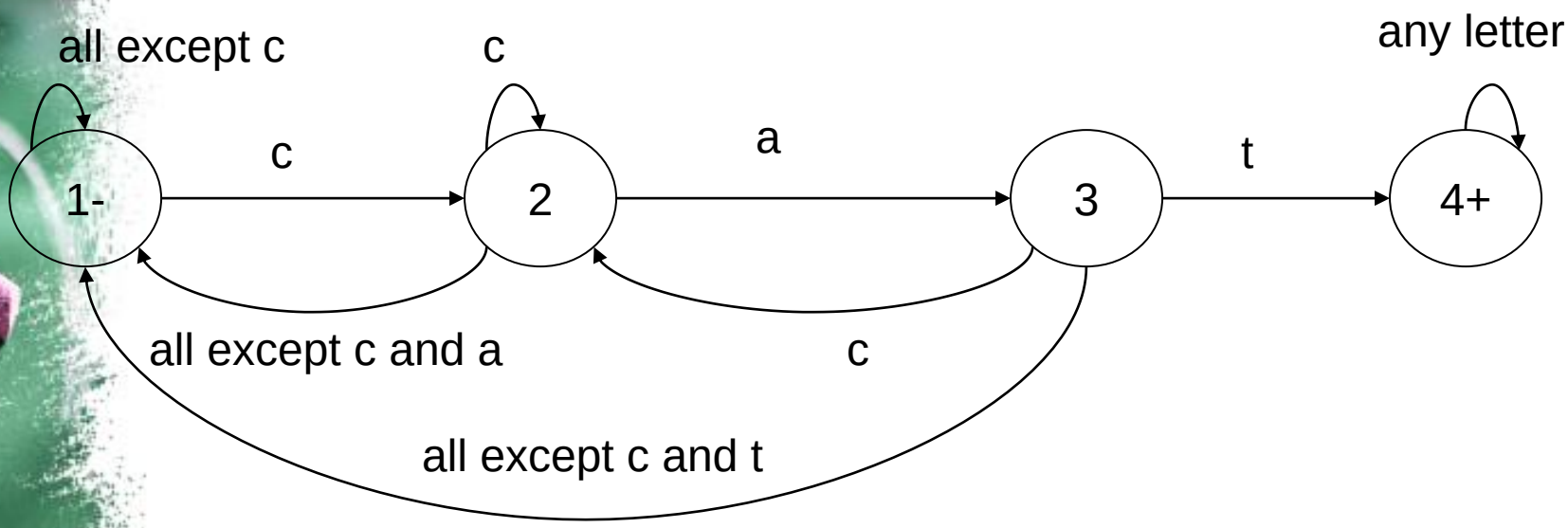
FA

- EVEN-EVEN
  - Language of all words having even number of ***a*s** and even number of ***b*s**



Applications of FA

- Lexical Analyzer of Compiler
- Search mechanism in word processor



Finite Automaton (Summary)

- Automated language recognition
- Machines embedded with grammatical rules that recognize a language
- REs define a language and FAs accept (or reject) them
- FAs serve two purposes
  - Implicit language definition
  - Recognition
- Sort of graphs consisting of nodes called states and edges called transitions

Note:

- Finite Automata FA are Deterministic Finite Automata DFA