Biçimsel Diller ve Otomata Teorisi

Sunu IV
Sonlu Otomata

İZZET FATİH ŞENTÜRK

Yet Another Method for Defining Languages

- Several games fit the following description
  - Pieces are set up on a playing board
  - Dice are thrown, and a number is generated at random
  - Depending on the number, the pieces on the board is rearranged in a fashion completely specified by the rules
- Player has no options about chaning the board
  - Everything is determined by the dice
  - No matter who throws the dice, no skill or choice is involved
  - The winner depends entirely on what sequence of numbers is generated by the dice, not on who moves them

States

- All possible positions of the pieces on the board
  - Let us call them **states**
- The game changes from one state to another by the input of a certain number
  - For each number, there is one and only one resulting state
  - The game can be in the **same state** after a number is entered
  - There is a state that means victory: **final state**
  - The game begins with the initial state (unique)

Finite Automaton

- Finite: The number of possible states and the number of letters in the alphabet (possible dice rolls) are finite
- Automaton: The change of states is totally governed by the input
- The determination of what state is next is automatic
- The plural of automaton is **automata**

Definition: Finite Automaton

- A finite automaton is a collection of three things
  - A finite set of states. One of them is the initial state and called the **start state**. Some (maybe none) are **final states**
  - An **alphabet** Σ of possible input letters
  - A finite set of **transitions** that tell for each state and for each letter of the alphabet which state to go to next
- The definition doesn’t describe how a FA works
  - Reads the input string letter by letter starting at the leftmost letter
  - Beginning at the start state, the letters determine a sequence of states
  - The dequence end when the last input letter has been read

Example

- The input alphabet has two letters a and b
- There are three states, x, y, and z
- The rules of transition
  - Rule 1, From state x and input a, go to state y
  - Rule 2, From state x and input b, go to state z
  - Rule 3, From state y and input a, go to state x
  - Rule 4, From state y and input b, go to state z
  - Rule 5, From state z and any input, stay at state z
- Starting state is x and the only final state is z
- This is a perfecly defined FA because it fulfills al three requirements: states, alphabet, transitions
- What happens when the input string is aaa or abba (accepted or rejected?)

Example

- **The set of strings** <u>accepted</u> by a FA is the **language** associated with this FA
- As soon as b is encountered in the input string, the FA jumps to state z and it is impossible to leave once in state z
- The FA will accept all strings that have the letter b in them
    - (**a** + **b**)***b**(**a** + **b**)*

Transition Table

- Much simpler to summarize rules in a table format
  - Each row is one of the states in the FA
  - Each column is a letter of the input alphabet
  - The entries are the new states that the FA moves into
- The transition table for the FA is

	a	b
Start x	y	z
y	x	z
Final z	z	z

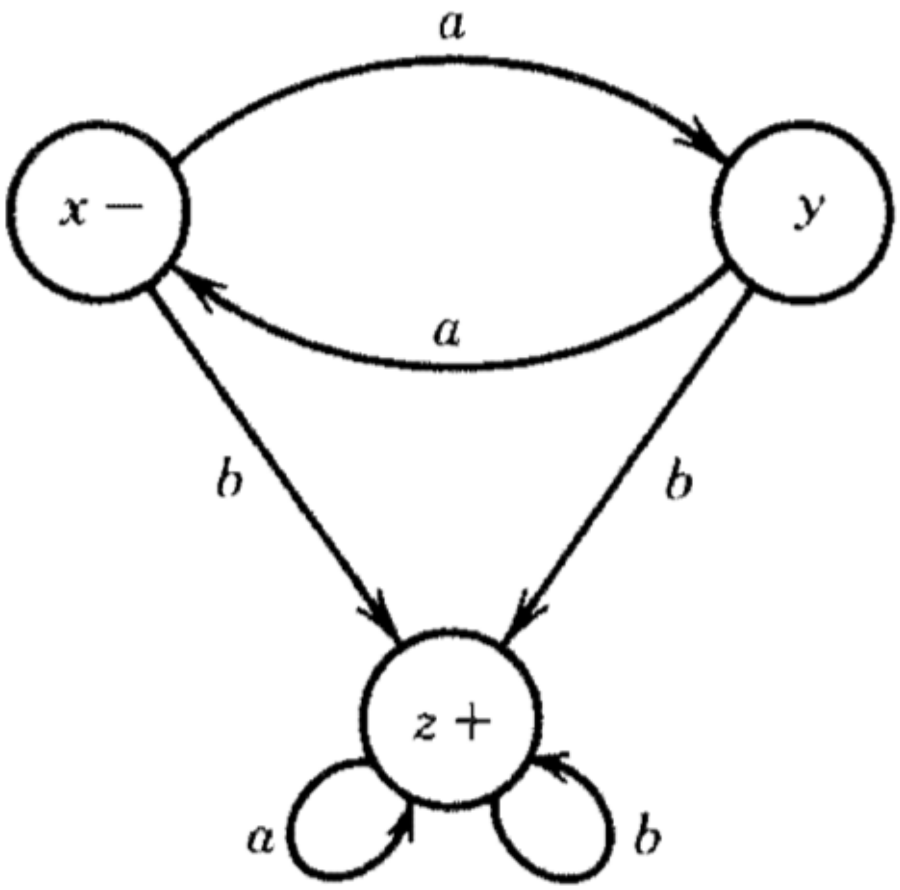
Abstract Definition of an FA

1. A finite set of states $Q = \{q_0, q_1\ q_2 \ldots\}$ of which q_0 is the start state
2. A subset of Q called the final states
3. An alphabet $\Sigma = \{x_1\ x_2\ x_3 \ldots\}$
4. A transition function δ associating each pair of state and letter with a state

   $\delta(q_i, x_j) = x_k$

The Transition Diagram

- Represent each state by a small circle
- Draw arrows from each state to other states
- Label arrows with the corresponding alphabet letters
- If a certain letter makes the state go back to itself: loop
- The start state is indicated with the word "start" or by a minus sign
- The final states are labeled with the word "final" or plus signs

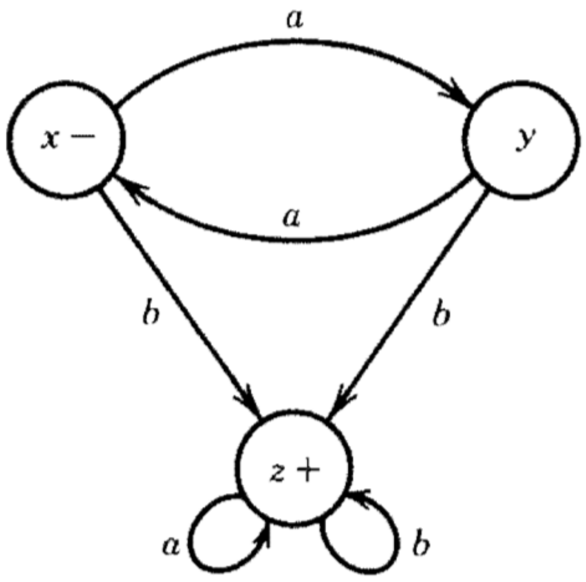


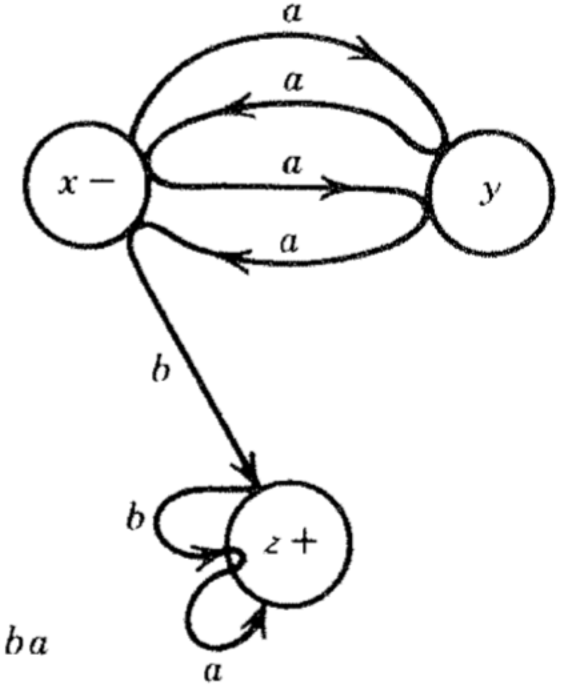
The Letters and the Traversing Path

- Every input string can be interpreted as traversing a path
  - Begin at the start state
  - Move among the states (perhaps visit some states many times)
  - Settle in some particular rest state
    - If it is a final state, the path has ended in success
- The letters of the input string dictate the directions of travel
- When we are out of letters, we must stop

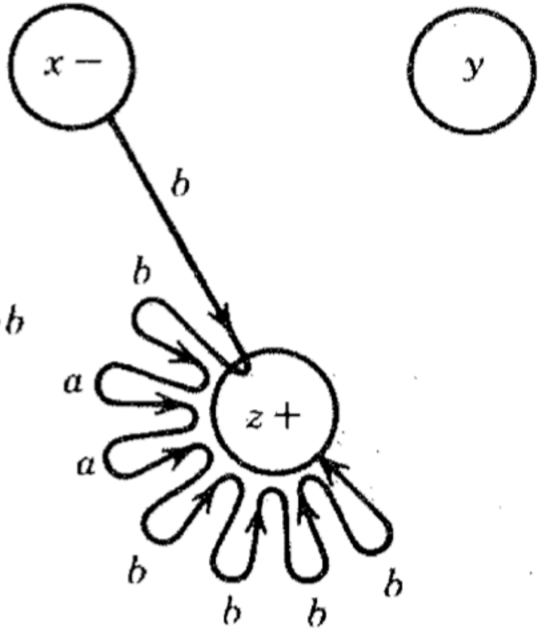
Example

- The paths generated by the input strings aaaabba and bbaabbbb





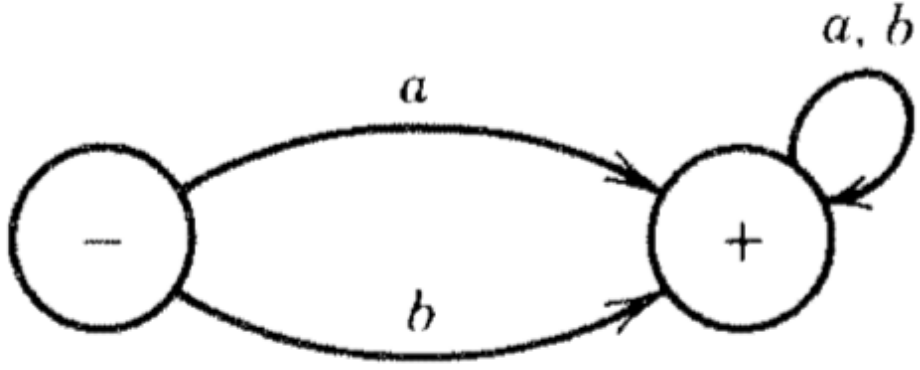
aaaabba



bbaabbbb

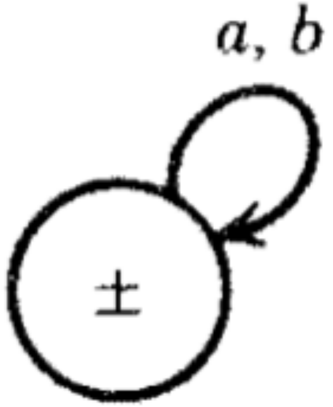
Example

- The language accepted by this machine is the set of all strings except Λ
- (**a** + **b**)(**a** + **b**)* = (**a** + **b**)$^+$



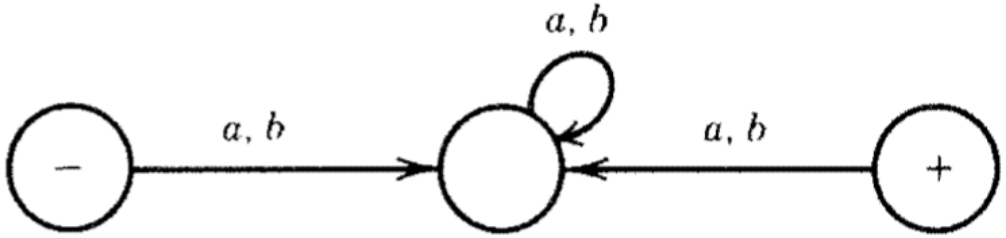
Example

- One of the many FAs that accepts all words is..
- The same state is both a start state and a final state
- (**a** + **b**)*



Example

- There are FAs that accept no language
- There are two types: FAs that have no final states and FAs that the final states cannot be reached from the start state



FAs and Their Languages

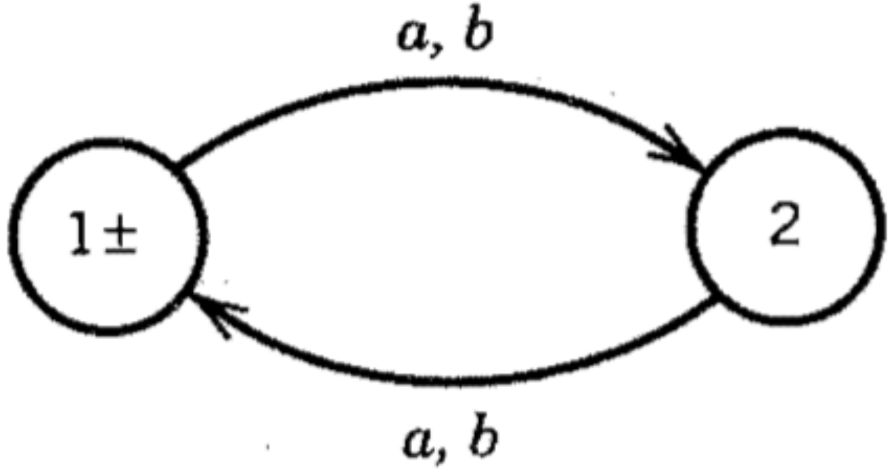
- When a language is defined by a RE, it is easy to produce some arbitrary words that are in the language
  - But it is harder to recognize whether a given string of letters is or is not in the language defined by the expression
- The situation with an FA is just the <u>opposite</u>!
  - Given a language defined by an FA, it is not easy to write down a bunch of words that we know in advance the machine will accept
- We must practice studying FA from two different angles:
  - Given a language, can we build a machine for it?
  - Given a machine, can we deduce its language?

Example

- Build a machine that accepts the language of all words over the alphabet {a b} with an even number of letters
- Mathematician approach: Count the number of letters as we go from left to right
- Computer scientist: ?
  - We are not interested in the exact length of the string
  - This number represents extraneous information gathered at the cost of needlessly many calculations
  - Employ a Boolean flag, employ only one storage location that can contain only one of two different values

Example

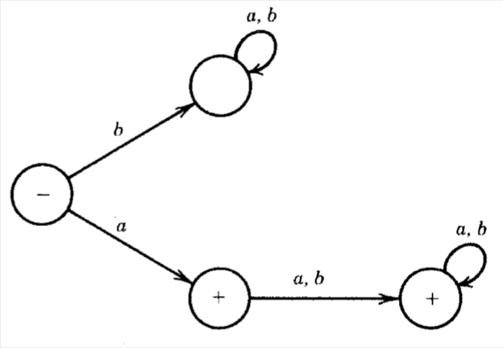
- Build a machine that accepts the language of all words over the alphabet {a b} with an even number of letters

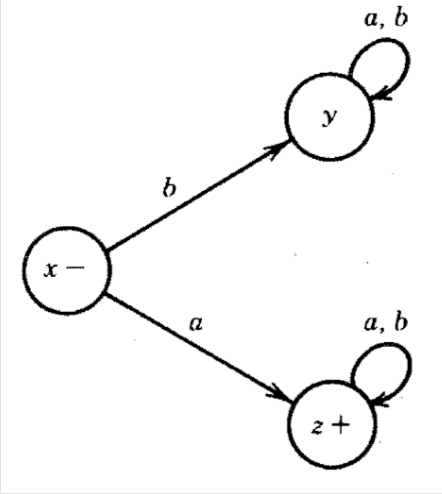


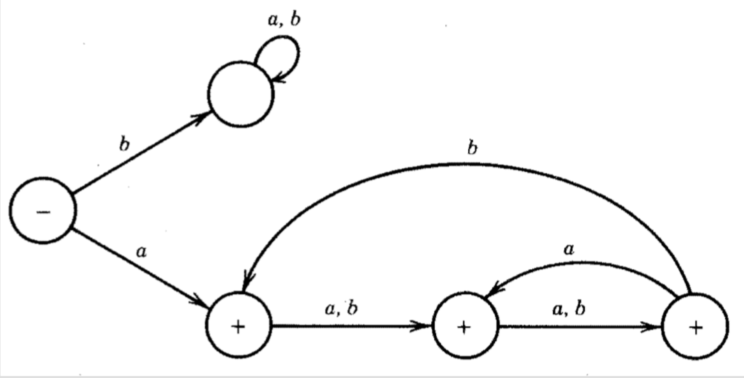
Example

- Build a FA that accepts all the words in the language that is all the strings that begin with the letter a
- **a**(**a** + **b**)*

- There is not a unique machine for a given language
- Is there always at least one FA that accepts each possible language?

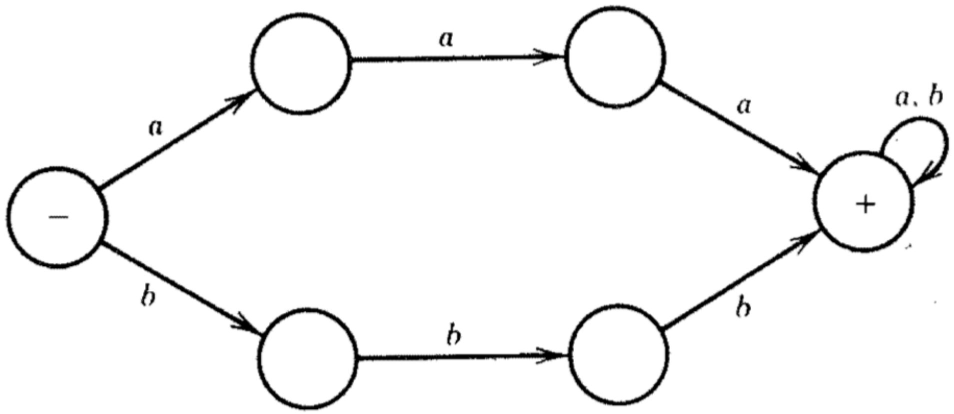


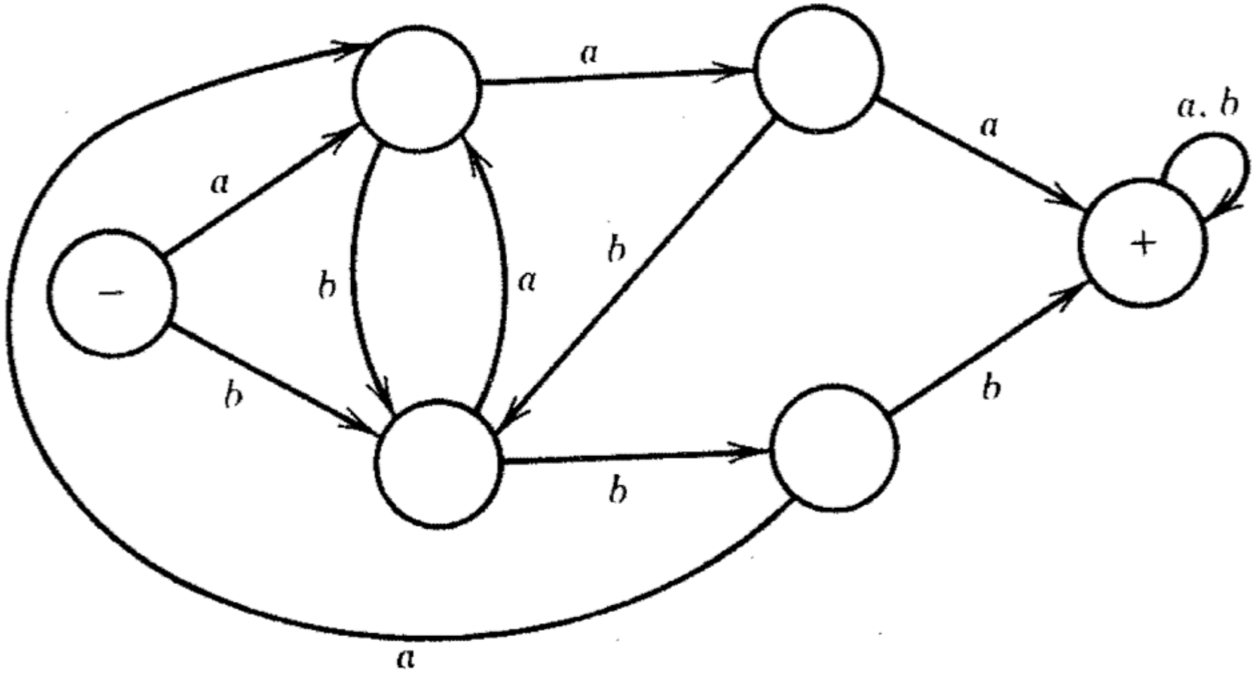




Example

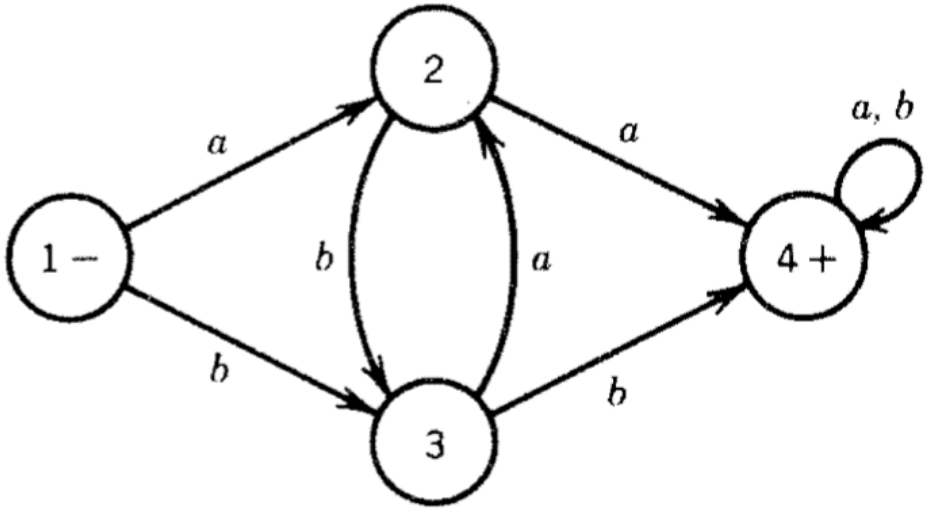
- Build an FA that accepts all words containing a triple letter, either aaa or bbb, and only those words
- We can understand the language and functioning of this FA because we have seen how it was built
  - If we had started with the final pictureand tried to interpret its meaning..





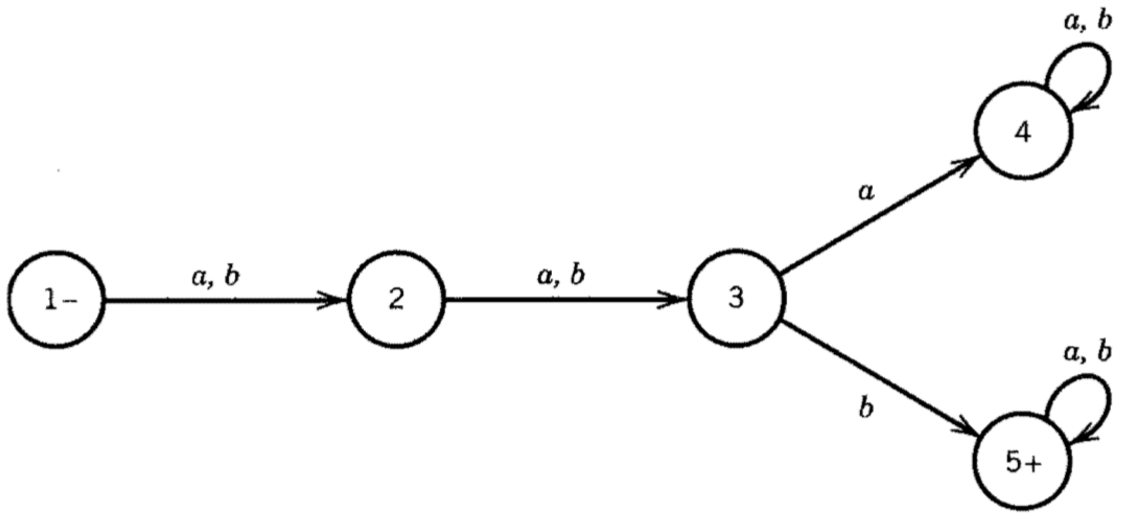
Example

- Examine what language this machine accepts
  - ababa is not accepted
  - babbb is accepted
- There are two ways to get to state 4
  - From state 2 (just read a)
  - From state 3 (just read b)
- Strings that have a double letter:
  - (**a** + **b**)*(**aa** + **bb**)(**a** + **b**)*



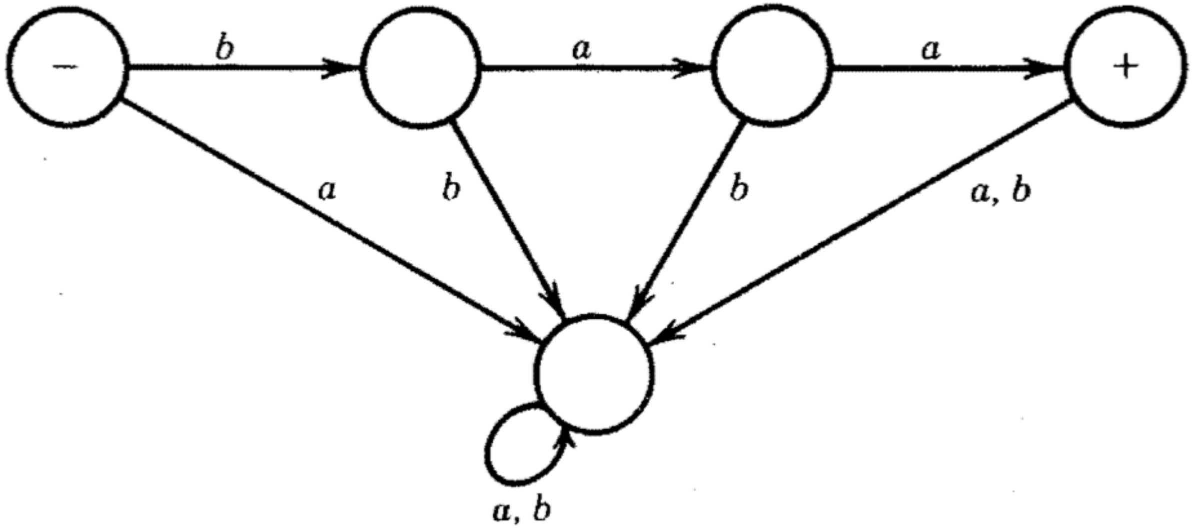
Example

- This machine will accept all words with b as the third letter and reject all other words
- Some Res that define this language:
  - (**aab** + **abb** + **bab** + **bbb**)(**a** + **b**)*
  - (**a** + **b**)(**a** + **b**)(**b**)(**a** + **b**)*



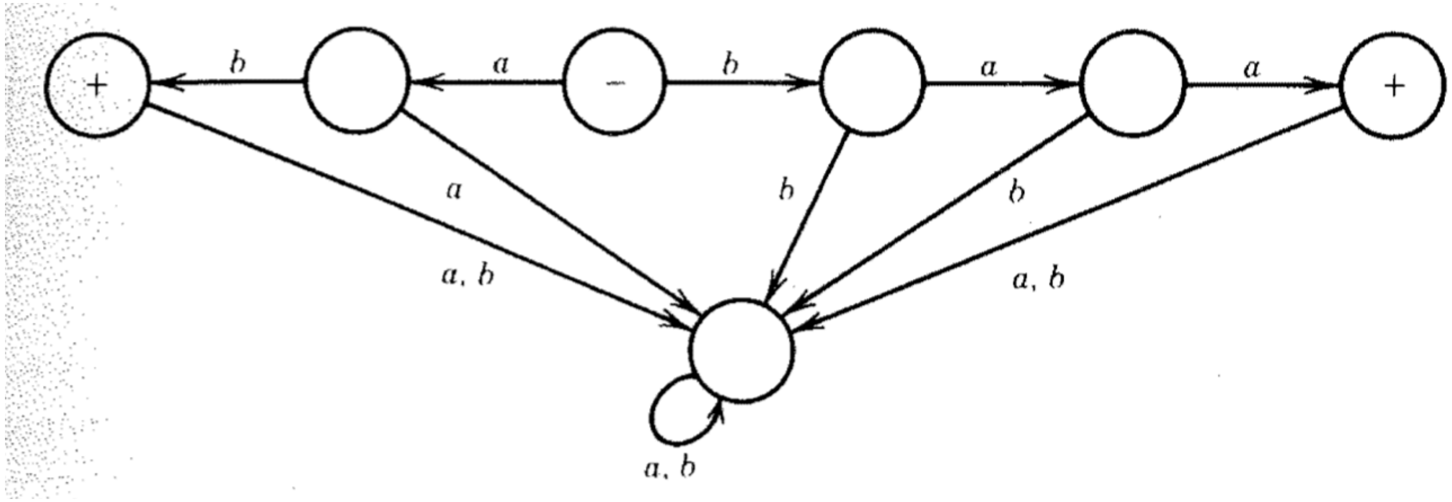
Example

- An FA that accepts only the word baa
- L = {baa}



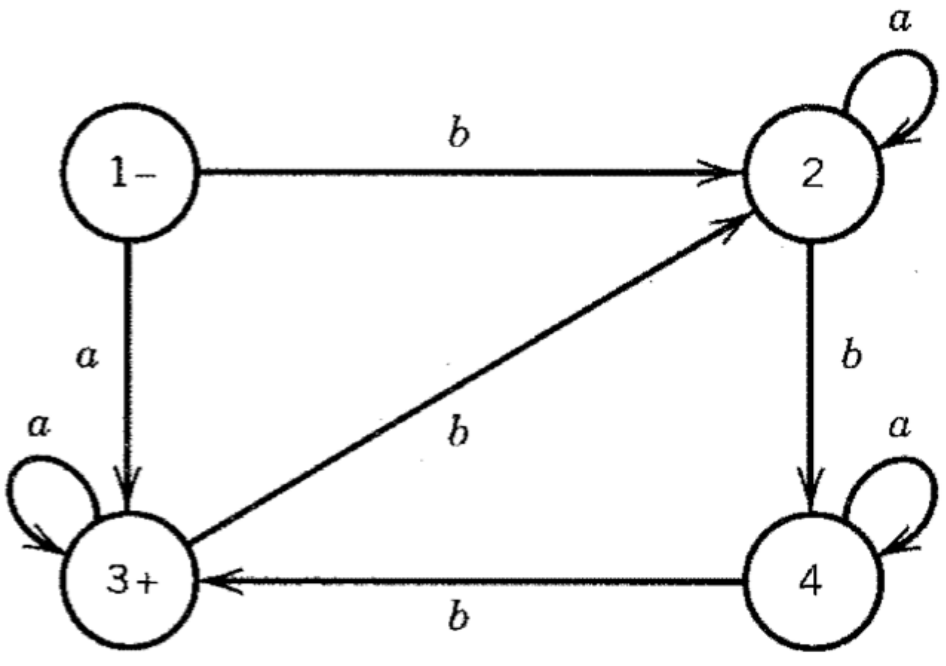
Example

- The FA accepts only two strings: baa and ab
- Big machine, small language



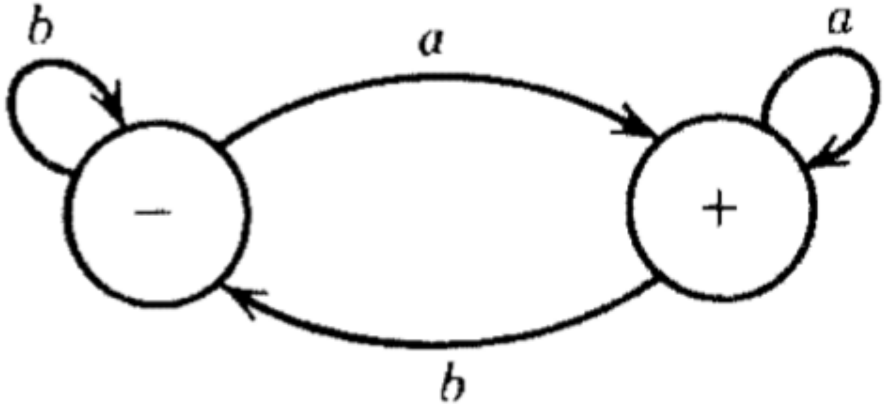
Example

- All words of the form **aa***
- **a***(**a*****ba*****ba*****ba***)*(**a** + **a*****ba*****ba*****ba***)
- The only purpose of the last factor is to guarantee that Λ is not a possibility



Example

- (**a** + **b**)*__a__



Example

- (**a** + **b**)*__a__