Theory of Computational

Finite Automata

Yet Another Method for Defining Languages
FAs and Their Languages
EVEN-EVEN **Revisited**

Yet Another Method for Defining Languages:

Definition:

A **finite automaton** is a collection of three things:

1. A finite set of states, **one** of which is designated as the initial state, called the **start state**, and **some** (maybe **none**) of which are designated as **final states**.
2. An **alphabet Σ** of possible input letters.
3. A finite set of **transitions** that tell for each state and for each letter of the input alphabet which state to go next.

How Does a Finite Automaton work?

- It works by being presented with an input string of letters that it reads letter by letter starting from the leftmost letter.
- Beginning at the start state, the letters determine a sequence of states.
- This sequence ends when the last input letter has been read
- We will use the term FA for the phrase "finite automaton".

Example:

Consider the following FA:

- The input alphabet has only the two letters a and b. (We usually use this alphabet throughout the chapter.)
- There are only three states, *x*, *y* and *z*, where *x* is the start state and *z* is the final state.
- The transition list for this FA is as follows:
  - 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*.
- Let us examine what happens when the input string *aaa* is presented to this FA.
- First input a: state $x \rightarrow y$ by Rule 1.
- Second input *a*: state $y \rightarrow x$ by Rule 3.
- Third input *a*: state $x \rightarrow y$ by Rule 1.

- We did not finish up in the final state z, and therefore have an unsuccessful termination.
- The set of all strings that lead to a final state is called the **language defined by the finite automaton**.
- Thus, the string *aaa* is **not in the language defined by this FA**.
- We may also say that the string aaa is **not accepted** by this FA, or the string *aaa* is **rejected** by this FA.
- The set of all strings accepted is also called the **language associated with the FA**.
- We also say, "This FA **accepts** the language L", or "L is the
- **language accepted** by this FA", or "L is the **language of** the FA", by which we mean that all the words in L are accepted, and all the inputs accepted are words in L
- It is not difficult to find the language accepted by this FA.
- If an input string is made up of only letter a's then the action of the FA will be to jump back and forth between state x and state y.
- To get to state z, it is necessary for the string to have the letter b in it. As soon as a b is encountered, the FA jumps to state z. Once in state z, it is impossible to leave. When the input string runs out, the FA will be in the final state z.
- This FA will accept all strings that have the letter b in them. Hence, the language accepted by this FA is defined by the regular expression: (a + b)*b(a + b)*

Transition Table:

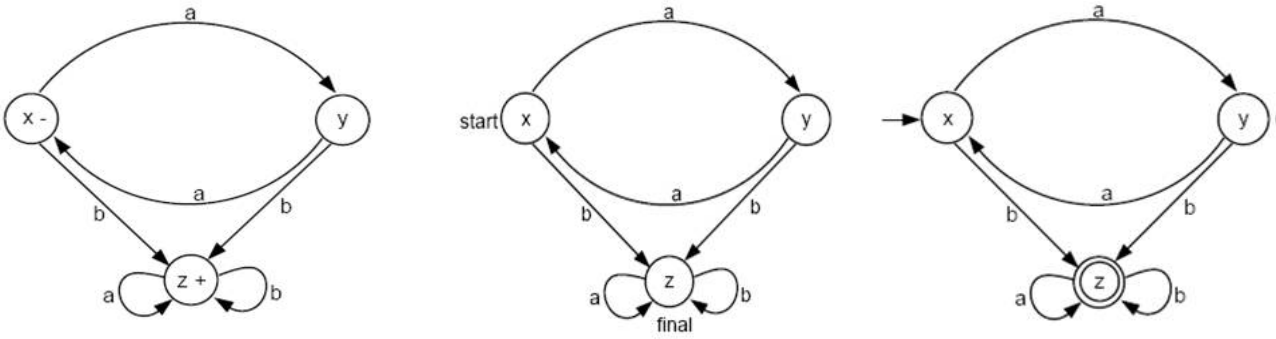
- The transition list can be summarized in a table format in which each row is the name of one of the states, and each column is a letter of the input alphabet.
- For example, the **transition table** for the FA above is

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

Transition Diagrams:

- Pictorial representation of an FA gives us more of a feeling for the motion.
- We represent each state by a small **circle**.
- We draw **arrows** showing to which other states the different **input letters** will lead us. We label these arrows with the corresponding input letters.
- If a certain letter makes a state go back to itself, we indicate this by a **loop**.
- We indicate the start state by a **minus sign**, or by labeling it with the word **start**.
- We indicate the final states by **plus signs**, or by labeling them with the word **final**.

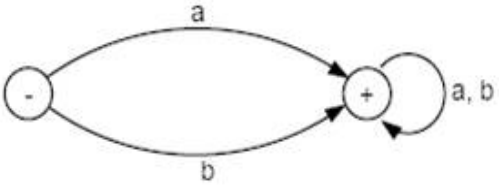
- Sometimes, a start state is indicated by an arrow, and a final state is indicated by drawing another circle around its circle.



- When we depict an FA as circles and arrows, we say that we have drawn a **directed graph**.
- We borrow from Graph Theory the name **directed edge**, or simply **edge**, for the arrow between states.
- **Every state has as many outgoing edges as there are letters in the alphabet.**
- **It is possible for a state to have no incoming edges or to have many.**

Example:

- By convention, we say that the null string starts in the start state and ends also in the start state for all FAs.
- Consider this FA:

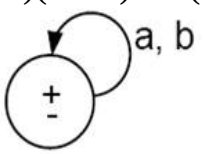


- The language accepted by this FA is the set of all strings except **Λ**.

The regular expression of this language is $(a + b)(a + b)^* = (a + b)^+$

Example:

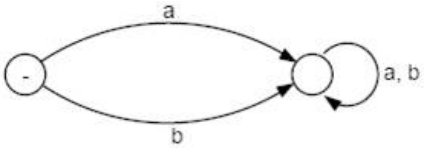
- One of many FAs that accept all words is



- Here, the ± means that the same state is both a start and a final state.
- The language for this machine is $(a + b)^*$

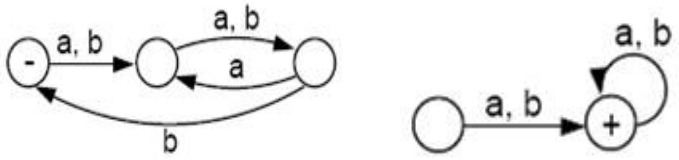
Example:

- There are FAs that accept no language. These are of two types:
- The first type includes FAs that have no final states, such as



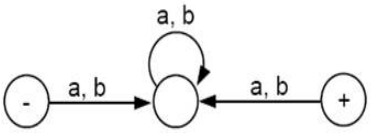
Example:

- The second type includes FAs of which the final states can not be reached from the start state.
- This may be either because the diagram is in two separate components. In this case, we say that the graph is **disconnected**, as in the example below:



Example:

- Or it is because the final state has no incoming edges, as shown below:

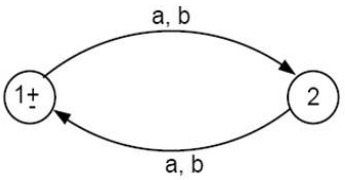


FA and their Languages:

- We will study FA from two different angles:
  1. Given a language, can we build a machine for it?
  2. Given a machine, can we deduce its language?

Example:

- Let us build a machine that accepts the language of all words over the alphabet Σ = {a, b} with an **even number of letters**.
- A mathematician could approach this problem by counting the total number of letters from left to right. A computer scientist would solve the problem differently since it is not necessary to do all the counting:
- Use a Boolean flag, named E, initialized with the value TRUE. Every time we read a letter, we reverse the value of E until we have exhausted the input string. We then check the value of E. If it is TRUE, then the input string is in the language; if FALSE, it is not.
- The FA for this language should require only 2 states:
  – State 1: E is TRUE. This is the start and also final state.
  – State 2: E is FALSE.
- So the FA is pictured as follows:



Example:

- Let us build a FA that accepts all the words in the language: a(a + b)*
- This is the language of all strings that begin with the letter a.
- Starting at state x, if we read a letter b, we go to a **dead-end** state y. A dead-end state is one that no string can leave once it has entered.
- If the first letter we read is an a, then we go to the dead-end state z, which is also a final state.

Example:

- The machine looks like this: