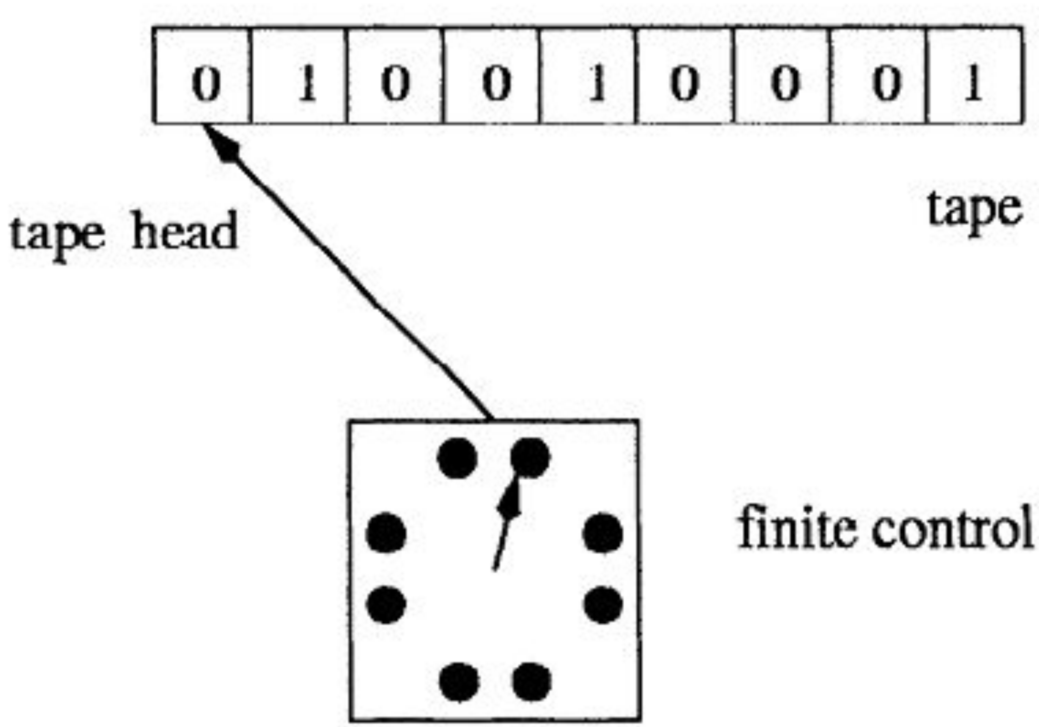
Finite automata

Irina Prosvirnina

- Deterministic finite automata
- Nondeterministic finite automata

Deterministic finite automata

Deterministic finite automaton (DFA) consists of three parts: a tape, a tape head (or, simply, head), and a finite control.

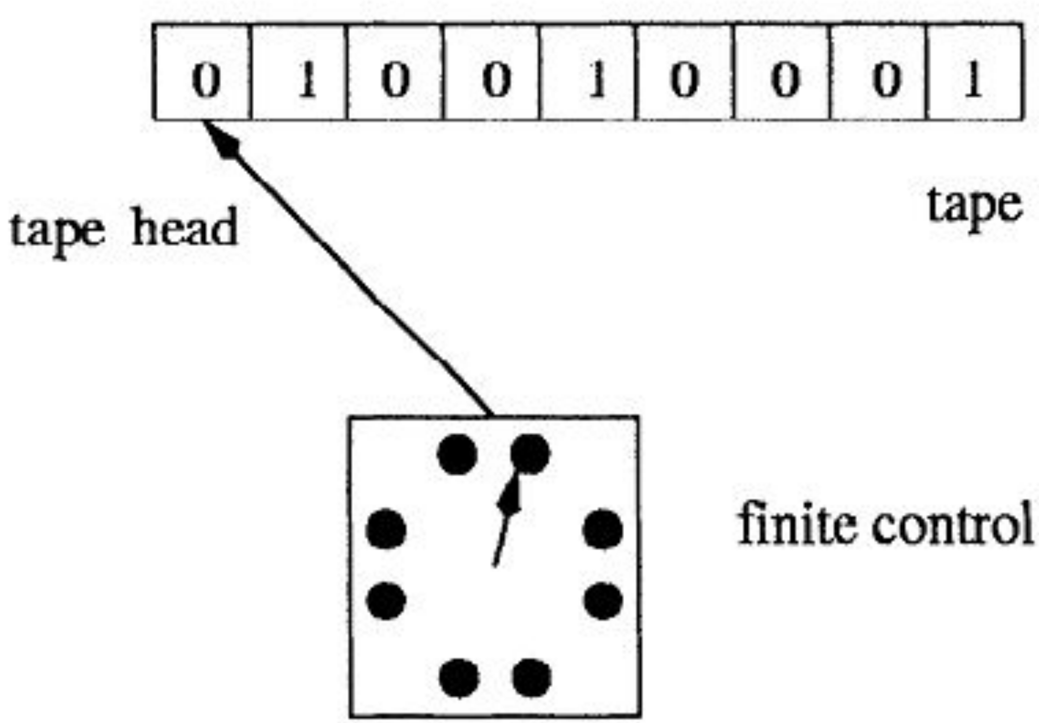


Deterministic finite automata

- In a formal theory, it is necessary to fix the set of symbols used to form strings.

Such a finite set of symbols is called an **alphabet**.

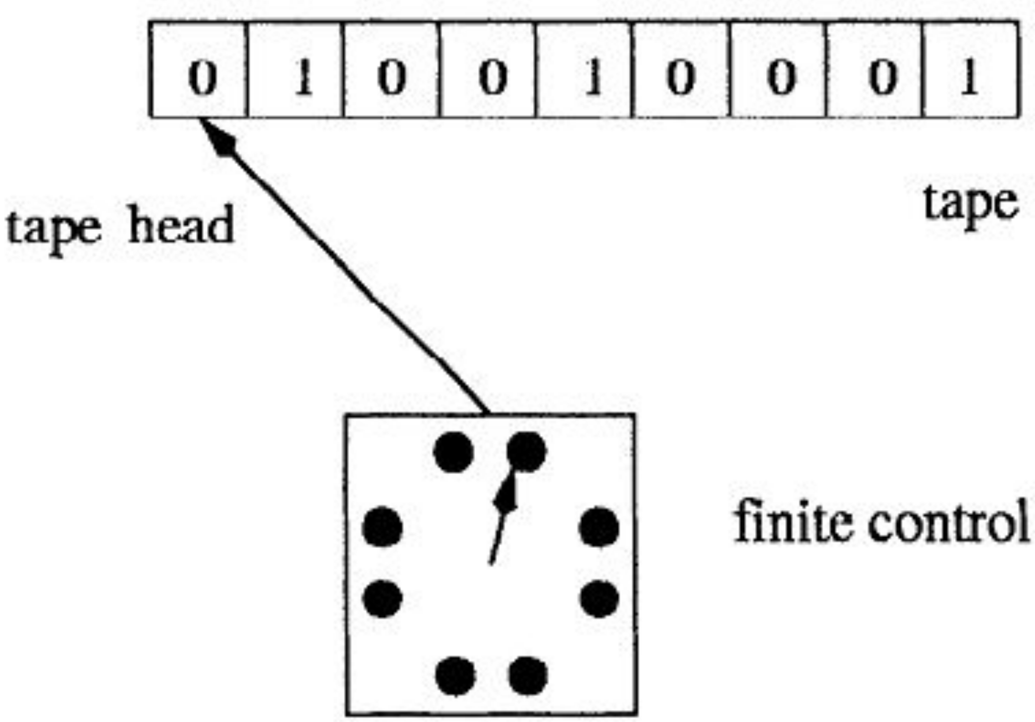
For exam



A string c **y**
string.

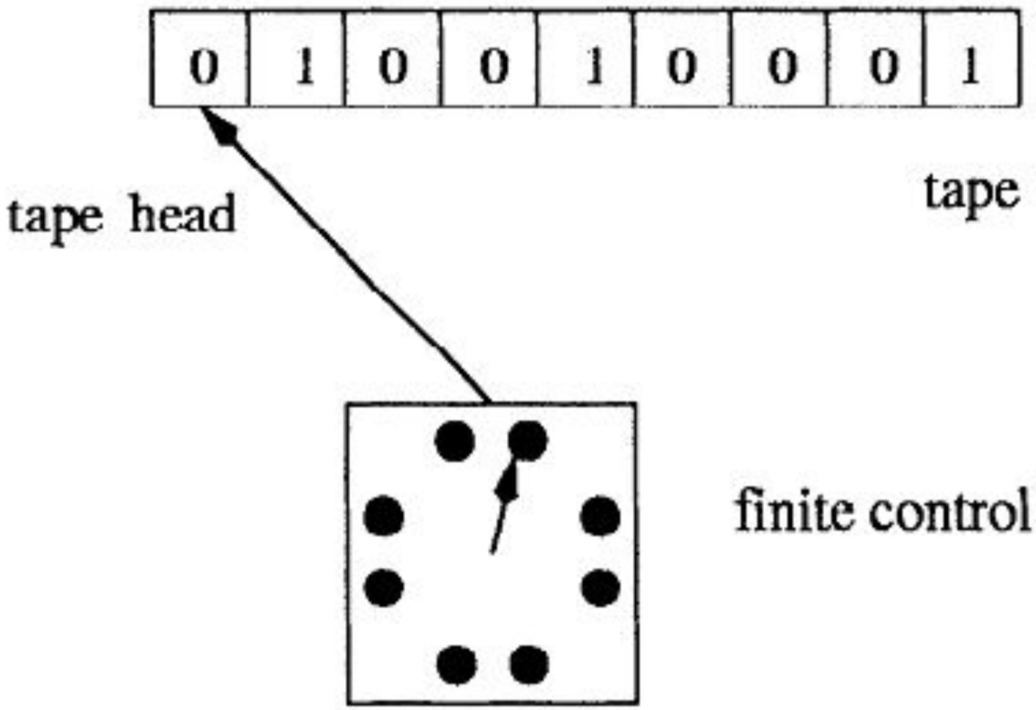
Deterministic finite automata

The tape head scans the tape, reads symbols from the tape, and passes the information to the finite control.



Deterministic finite automata

At each move of the DFA, the head scans one cell of the tape and reads the symbol in the cell, and then moves to the next cell to the right.



Deterministic finite automata

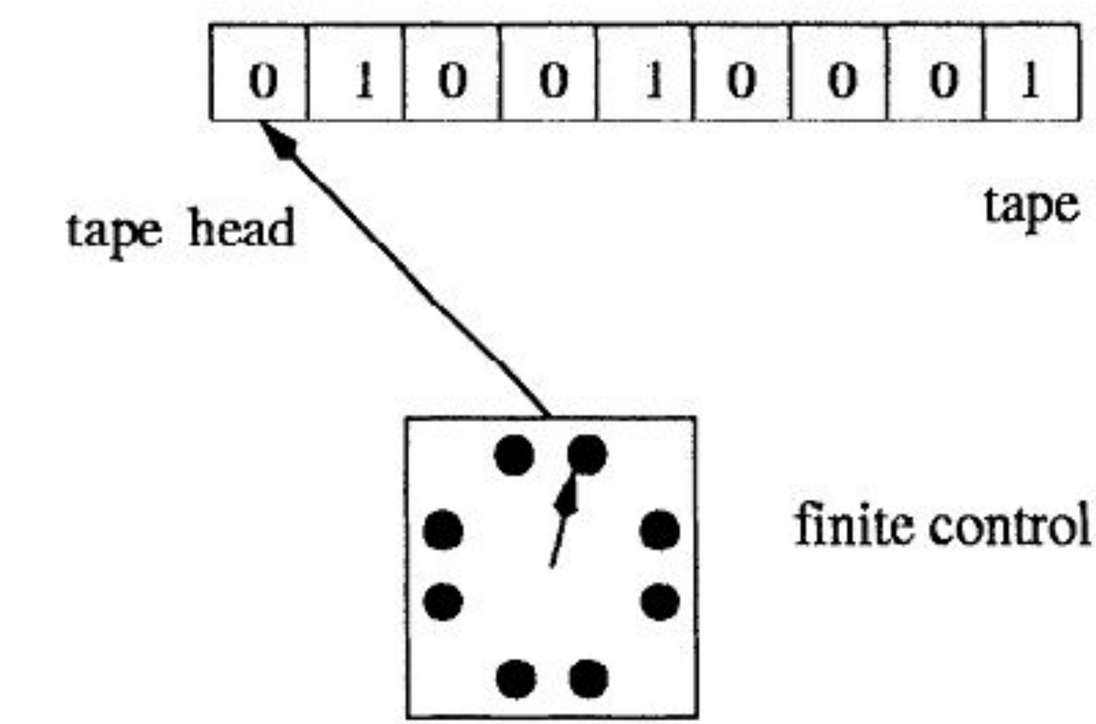
The **length** of a string x, denoted by $|x|$, is the number of symbols contained in the string.

For example,

|stri

|cs5

|10(

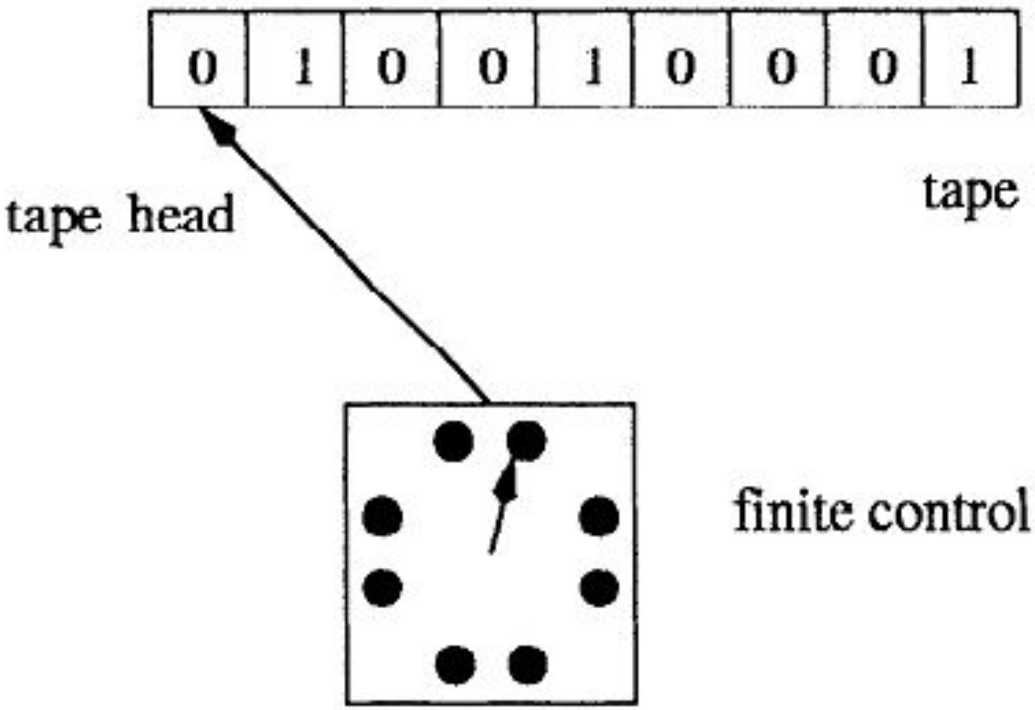


The **emp** no symbol.

Clearly, |

Deterministic finite automata

Then, it determines, from the current state and the symbol read by the tape, how the state is changed to a new state.



Deterministic finite automata

Example 1

How many strings over the alphabet $A = \{a_1, a_2, \ldots, a_k\}$ are there which are of length n, where n is a nonnegative integer?

Solution. There are n positions in such a string, and each position can hold one of k possible symbols. Therefore, there are k^n strings of length exactly n.

Deterministic finite automata

- For a string x over alphabet Σ the **reversal** of x, denoted by x^R, is defined by

$$x^R = \begin{cases} \varepsilon, \text{if } x = \varepsilon, \\ x_n \ldots x_2 x_1, \text{if } x = x_1 x_2 \ldots x_n. \end{cases}$$

Deterministic finite automata

Example 2

For strings x and y

$$(xy)^R = y^R x^R.$$

Deterministic finite automata

When the DFA halts, it accepts the input string if it halts in one of the final states.

Otherwise, the input string is rejected.

Deterministic finite automata

A **language** is a set of strings.

Let Σ be an alphabet. We write Σ^* to denote the set of all strings over Σ.

Thus, a language L over Σ is just a subset of Σ^*.

For any finite language A we write $|A|$ to denote the size of A (i.e. the number of strings in A).

Deterministic finite automata

The transition diagram of a DFA is an alternative way to represent the DFA.

Deterministic finite automata

- **Union**.

If A and B are two languages, then

$$A \cup B = \{x |\ x \in A \text{ or } x \in B\}.$$

Deterministic finite automata

- **Intersection**.

If A and B are two languages, then

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}.$$

Deterministic finite automata

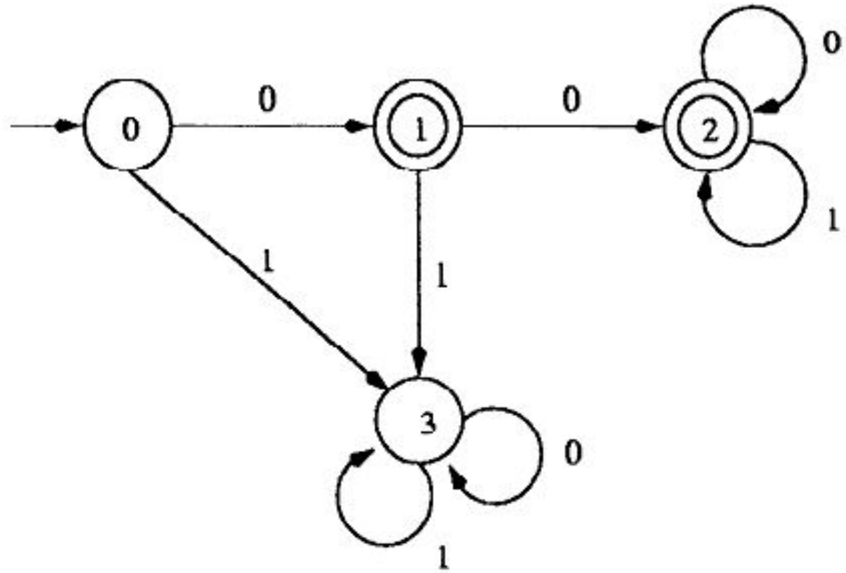
- **Complementation.**

If A is a language over the alphabet Σ, then

$$\bar{A} = \Sigma^* - A.$$

- **Concatenation.**

If A and B are two languages, then their concatenation is



We

Deterministic finite automata

- Since the transition function $\delta: Q \times \Sigma \rightarrow Q$ is well-defined on $Q \times \Sigma$, the transition diagram of the DFA has the property that for every vertex (state) q and every symbol a, there exists exactly one edge with label a leaving q.

This implies that for each string x, there exists exactly one path starting from q_0 whose labels form the string x.

This path is called the computation path of the DFA on x. We note that a string x is accepted by M if and only if its computation path ends at one of the final states.

Deterministic finite automata

Example 4

The language $\{0,10\}^*$ is the set of all binary strings having no substring 11 and ending with 0.

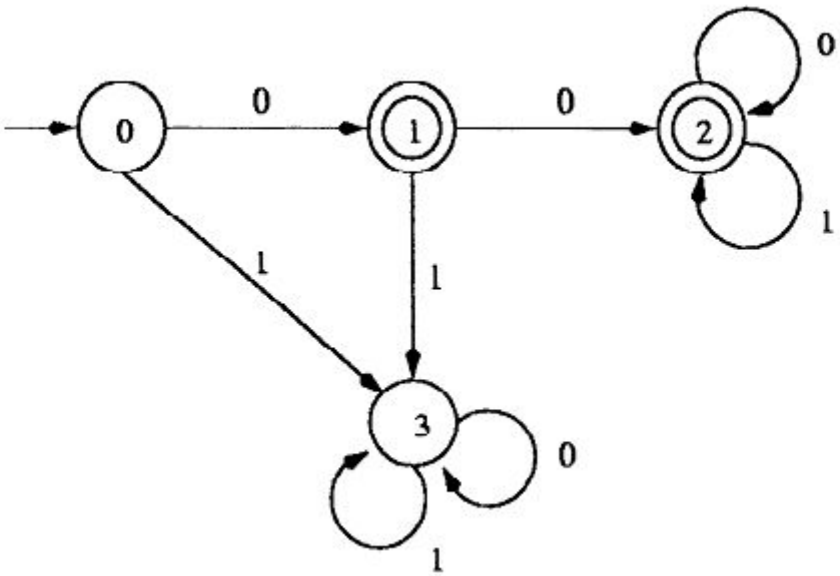
Deterministic finite automata

- The concept of **regular languages** (or, regular sets) over an alphabet Σ is defined recursively as follows:

1) The empty set $\emptyset$ is a regular language.
2) For every symbol $a \in \Sigma$, $\{a\}$ is a regular language.
3) If A and B are regular languages, then $A \cup B$, AB, A^* are all regular languages.
4) Nothing else is a regular language.

- Example 5

The set $\{\varepsilon\}$ is a regular language, because $\{\varepsilon\} = \emptyset^*$.



Deterministic finite automata

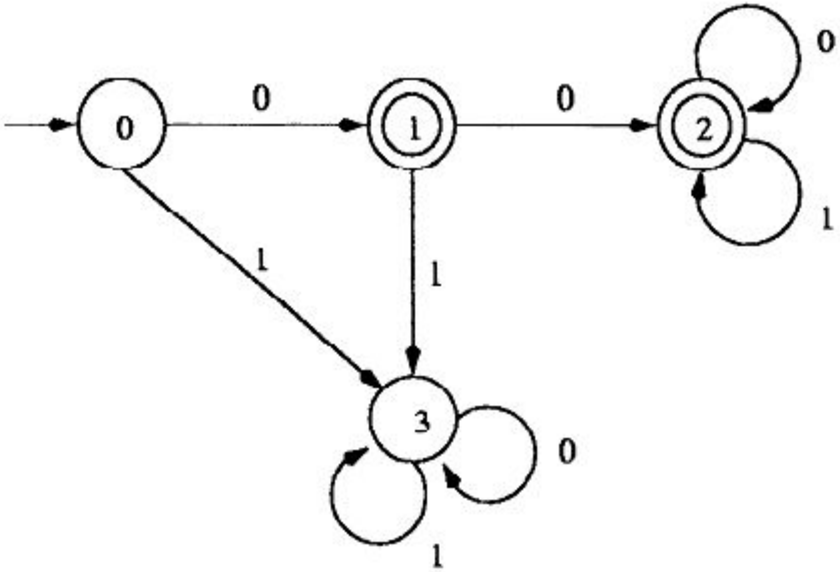
Example 6

The set $\{001, 110\}$ is a regular language over the binary alphabet:

$$\{001, 110\} = (\{0\}\{0\}\{1\}) \cup (\{1\}\{1\}\{0\}).$$

Example 5

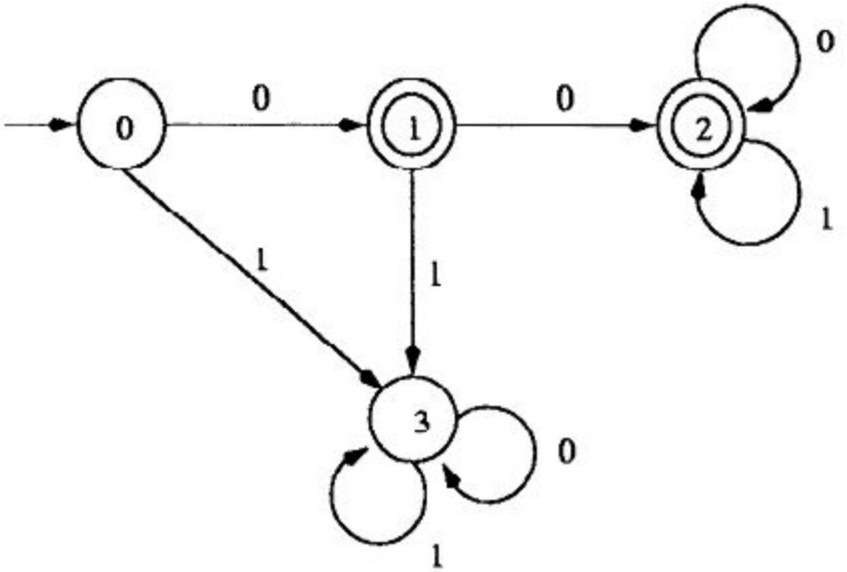
The set $\{\varepsilon\}$ is a regular language, because $\{\varepsilon\} = \emptyset^*$.



Deterministic finite automata

- To simplify the representations for regular languages, we define the notion of regular expressions over alphabet Σ.

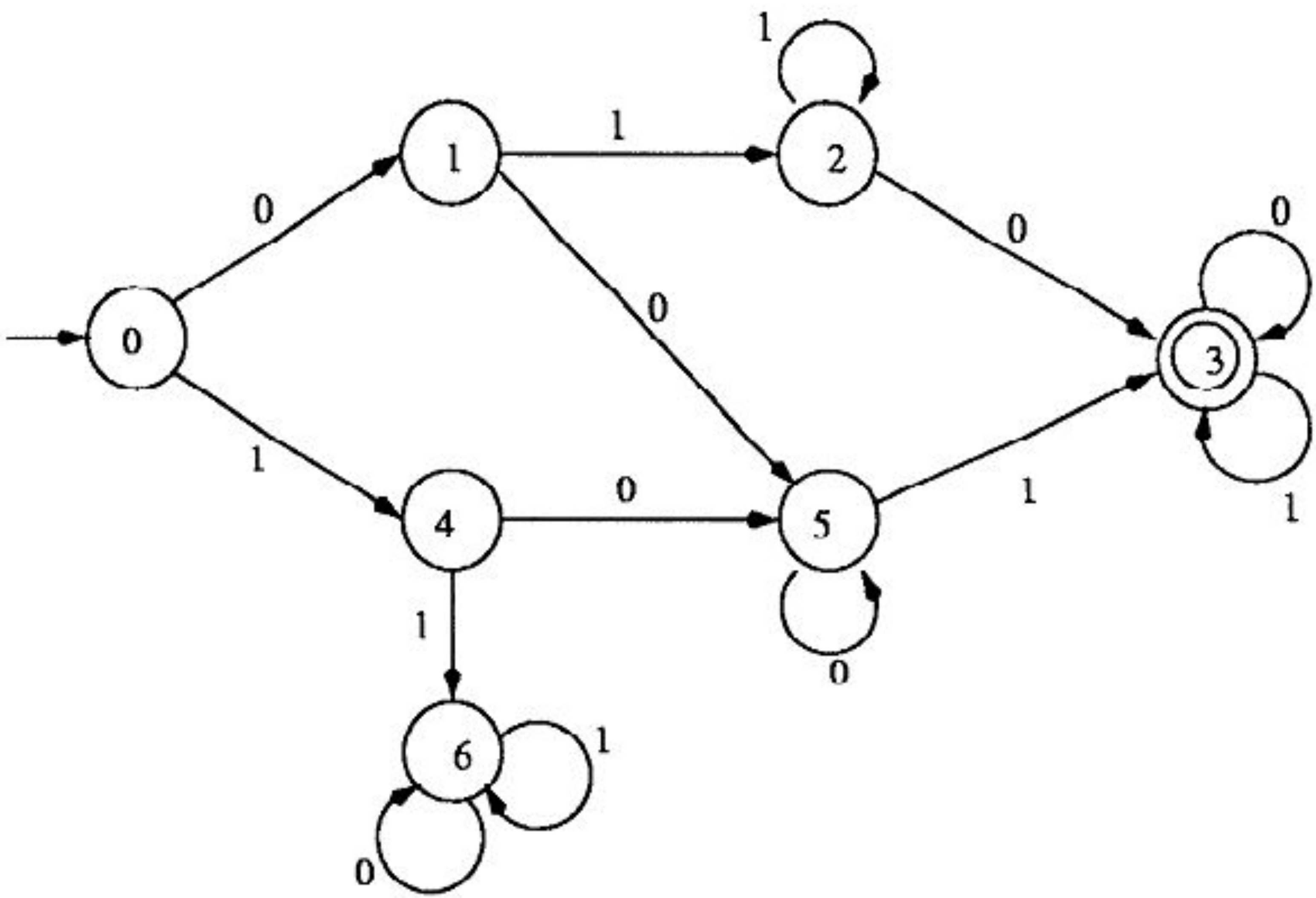
- The concept of regular expressions over an alphabet Σ is defined recursively as follows:
1) Ø is a regular expression which represents the



5) Nothing else is a regular expression over Σ.

Example 8

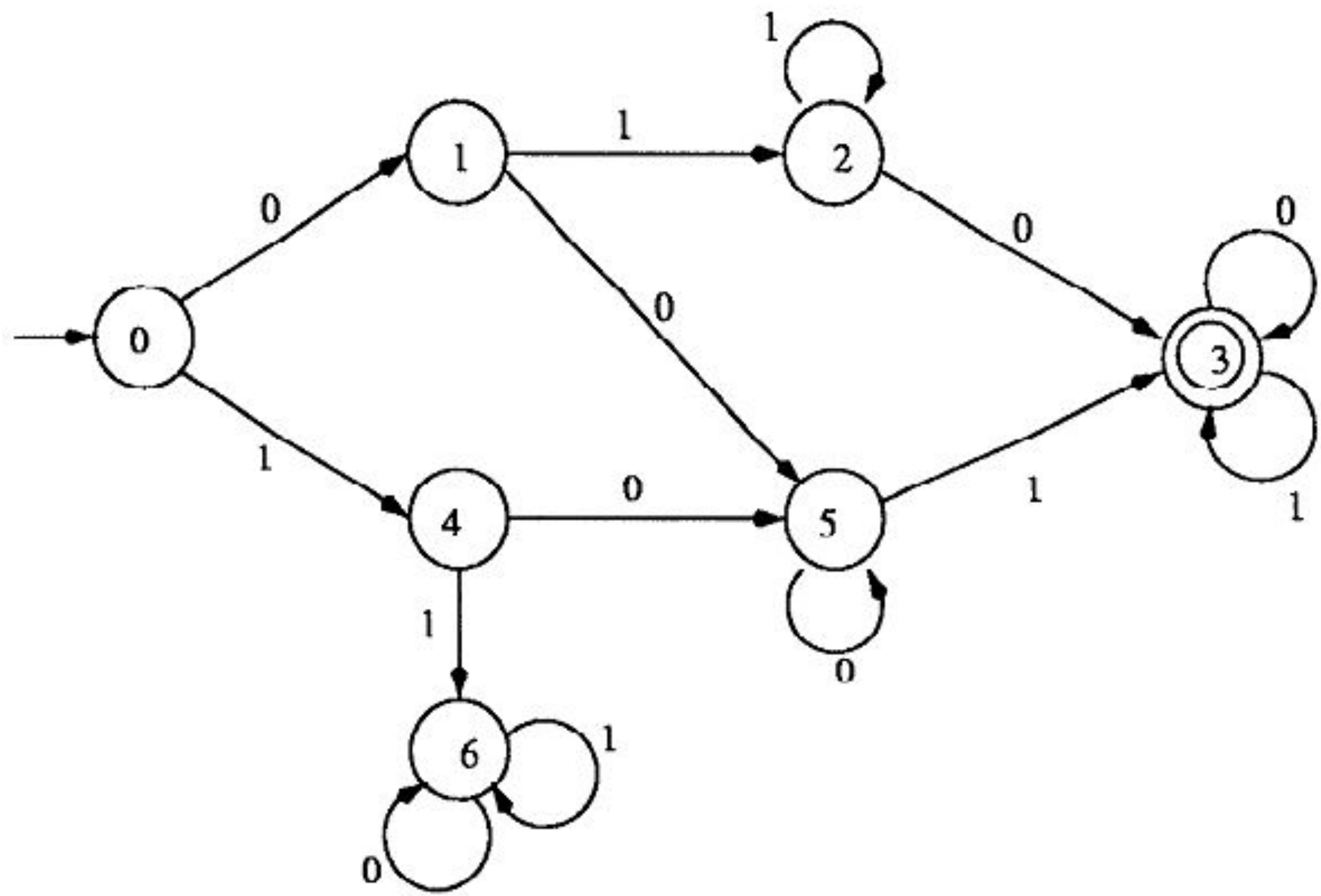
Language $A = \{0\}^*$ has a regular expression $r_A = (\mathbf{0})^*$.
Language $B = \{00\}^* \cup \{0\}$ has a regular expression $r_B = (((\mathbf{0})(\mathbf{0}))^*) + (\mathbf{0})$.



We can simplify the regular expression $r_A = (\mathbf{0})^*$ to $\mathbf{0}^*$.

We can simplify the regular expression
$r_B = (((\mathbf{0})(\mathbf{0}))^*) + (\mathbf{0})$
to $(\mathbf{00})^* + \mathbf{0}$.

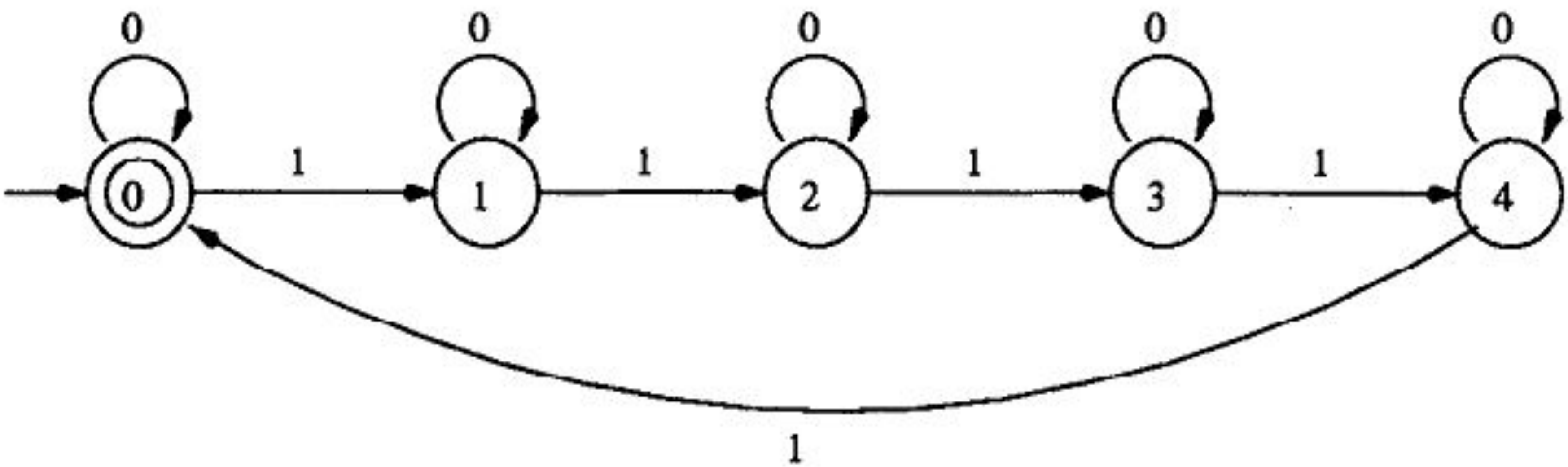
We can simplify the regular expression
$(\{0\}^* \cup (\{1\}\{0\}\{0\}^*))\ \{1\}\{0\}^*((\{0\}\{1\}^*) \cup \{1\}^*)$
to $(\mathbf{0}^* + \mathbf{100}^*)\mathbf{10}^*(\mathbf{01}^* + \mathbf{1}^*)$.



Deterministic finite automata

- The operations $+$ and $\cdot$ in a regular expression satisfy the **distributive law**: for any regular r, s and t,

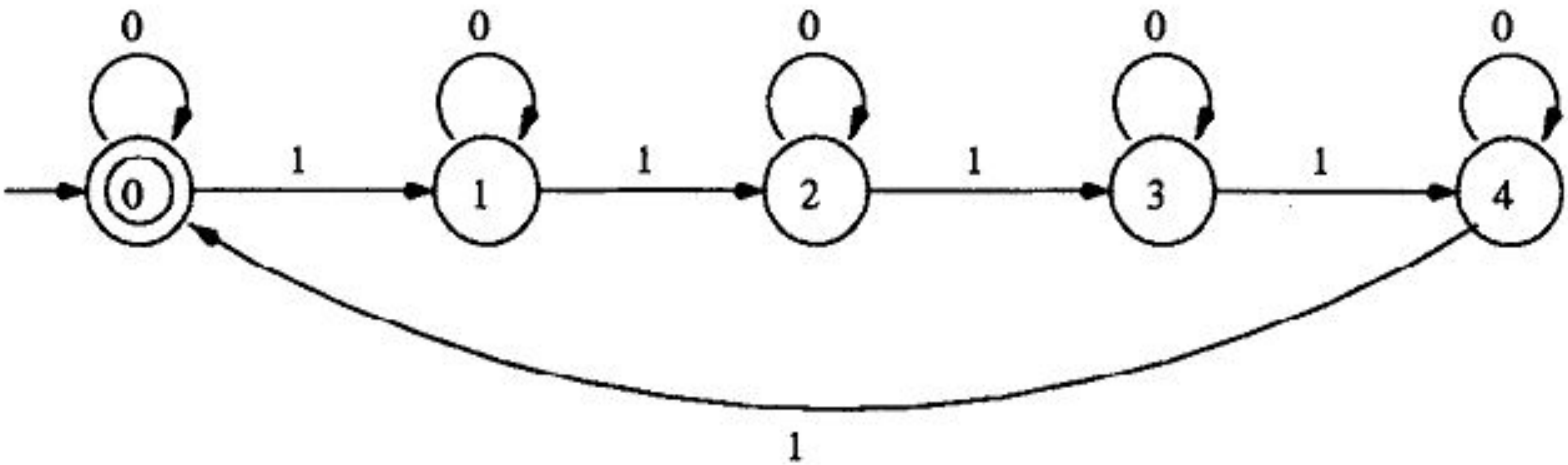
$$r(s+t) = rs + rt,$$



Deterministic finite automata

Example 10

Find a regular expression for the set of binary integers which are the expansions of 4.



Nondeterministic finite automata

•

Nondeterministic finite automata

•

Nondeterministic finite automata

•

Nondeterministic finite automata

-

Nondeterministic finite automata

•

Nondeterministic finite automata

•

Nondeterministic finite automata

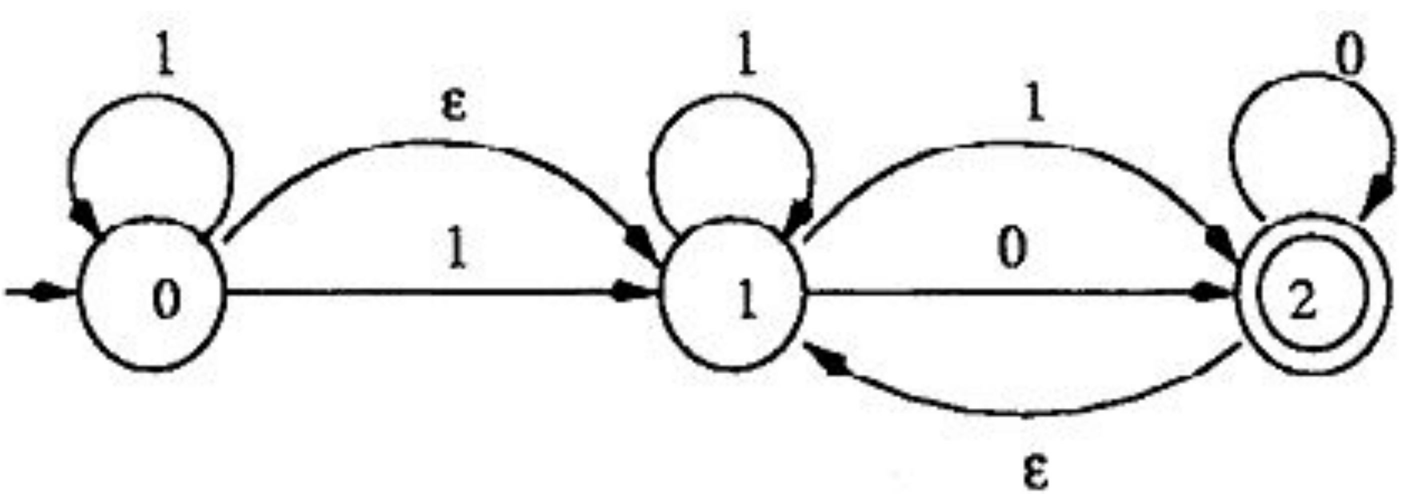
NFA’s, like DFA’s, can also be represented by transition diagrams.

In the transition diagram, we still use a vertex to represent a state and a labeled edge to represent a move, except that we allow multiple edges from one vertex to other vertices with the same label.

Nondeterministic finite automata

-

Nondeterministic finite automata

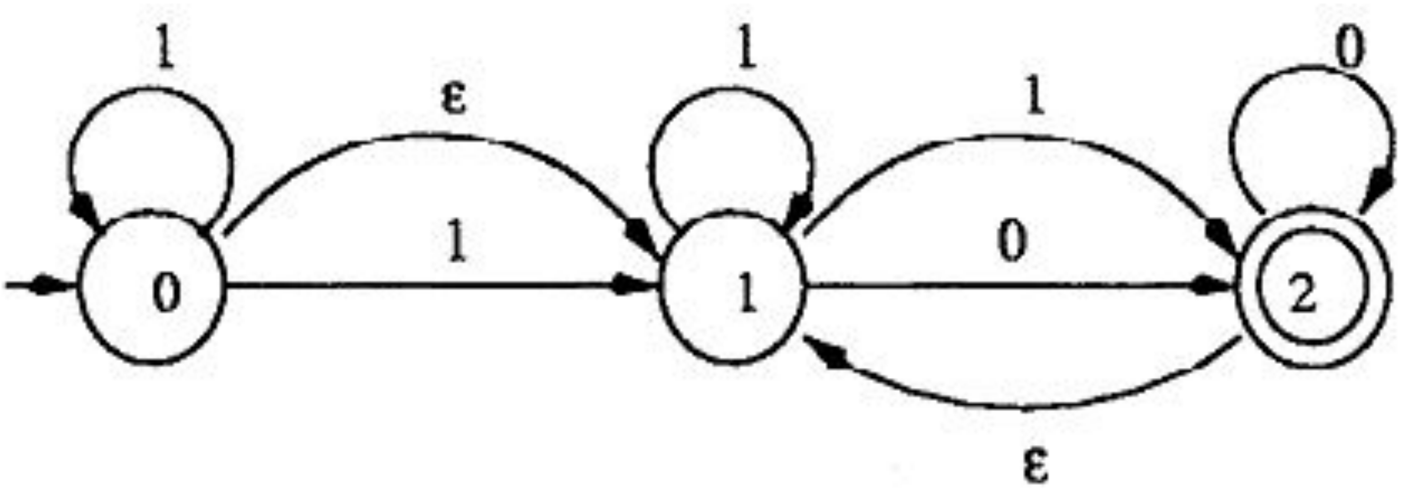


Nondeterministic finite automata

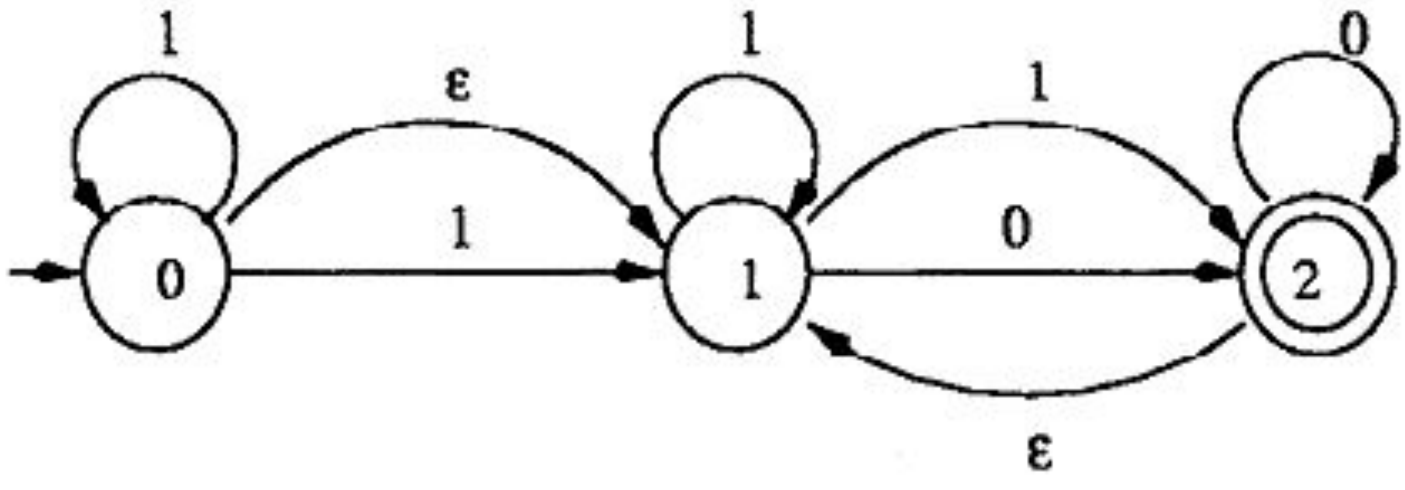
•

Nondeterministic finite automata

•



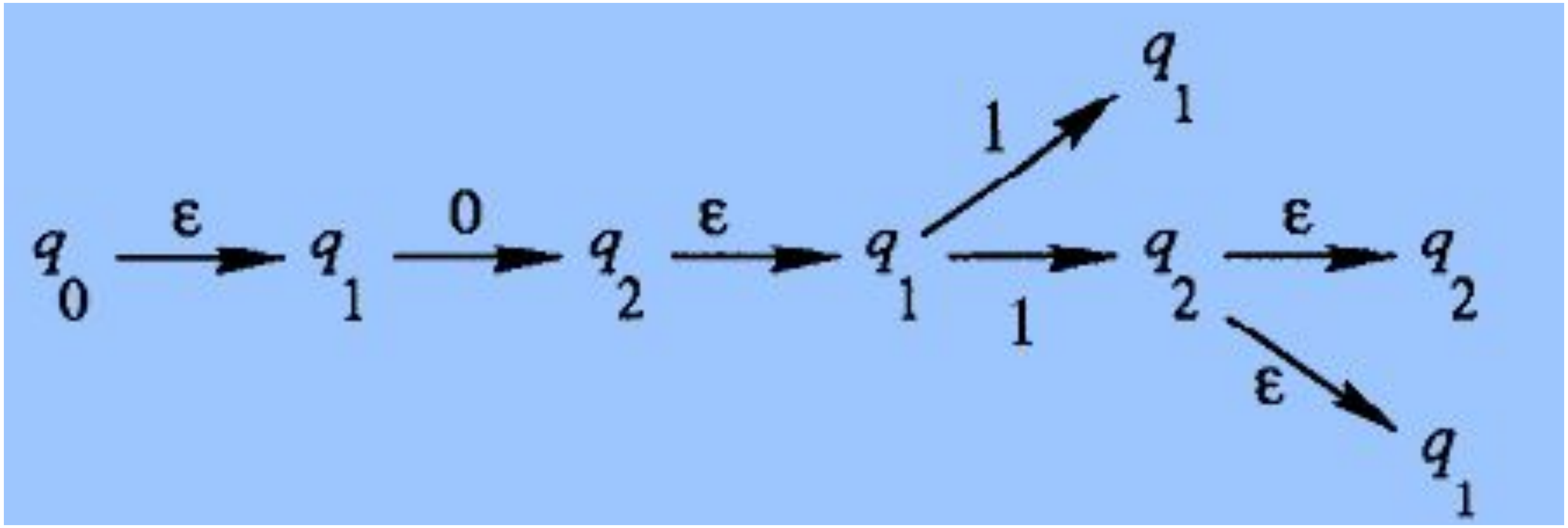
•


1
ε
1
1
0
1
0
0
1
2
0
ε

Nondeterministic finite automata

-

•



q_0
ε
q_1
0
q_2
ε
q_1
1
q_1
1
q_2
ε
q_2
ε
q_1

Nondeterministic finite automata

Some of these computation paths lead to final states and some do not.

Nondeterministic finite automata

-

Finite automata

-