MATH 433

Applied Algebra

Lecture 16:
Finite state machines (continued).
Permutations.

Finite state machine

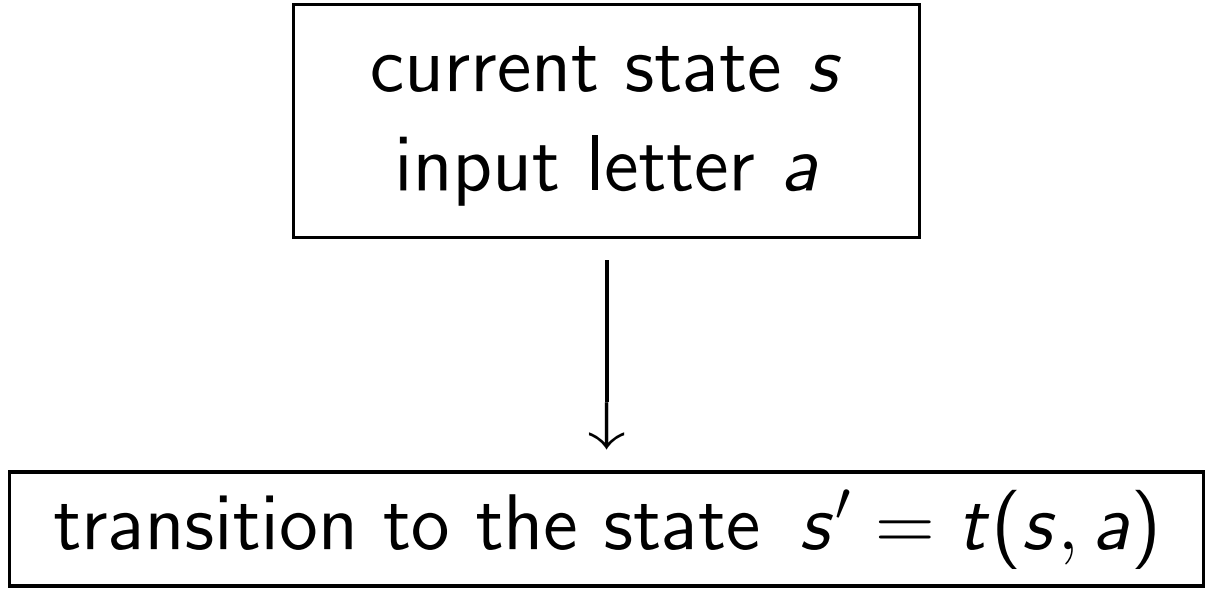
A **finite state machine** is a triple $M = (S, A, t)$, where S and A are nonempty finite sets and $t : S \times A \to S$ is a function.

- Elements of S are called **states**.
- There is one distinguished element of S called the **initial state**.
- The set A is called the **input alphabet**, its elements are called **letters**.
- The function t is called the **state transition function**.

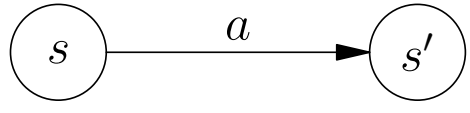
Notation in the textbook: the states are denoted by natural numbers; the initial state is denoted 0; the alphabet is a subset of the Roman alphabet.

We think of the finite state machine $M = (S, A, t)$ as a formal description of a certain device that operates in a discrete manner. At any moment it is supposed to be in some state $s \in S$. The machine reads an input letter $a \in A$. Then it makes transition to the state $t(s, a)$. After that the machine is ready to accept another input letter.

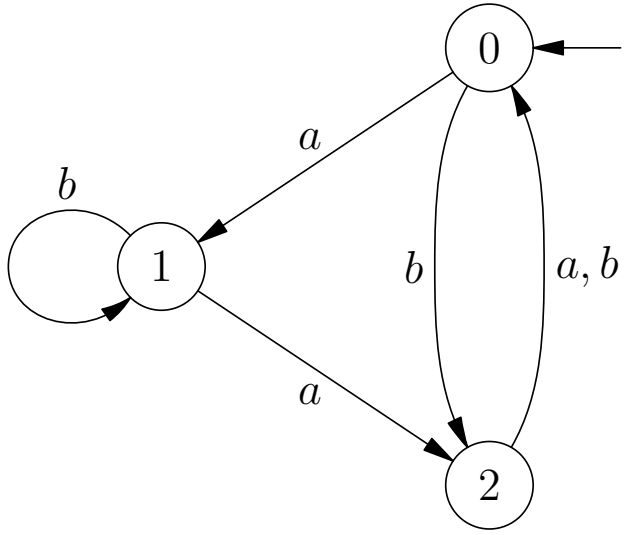
A machine's job looks as follows. We prepare an **input word** w, i.e., a sequence $a_1 a_2 \ldots a_n$ of letters from A. Then we set the machine to the initial state s_0 and start inputting the word w into it, letter by letter. The machine's job results in a sequence of states $s_0, s_1, \ldots, s_n$, which describes the internal work of the device.



One way to define (or picture) a finite state machine is the **Moore diagram**. This is a directed graph with labeled vertices and edges. Vertices are labeled by states, edges show possible transition routes (labeled by letters). The initial state is marked by an arrow pointing at it from nowhere.



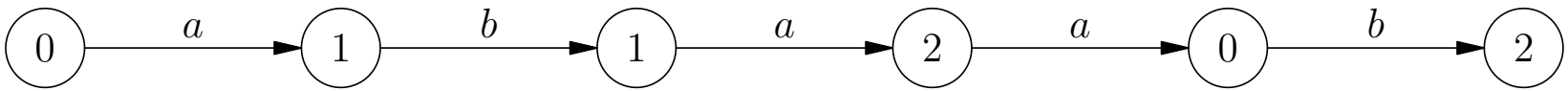
Example



	a	b
0	1	2
1	2	1
2	0	0

(The table is another way to define the transition function.)

Suppose that the input word is $w = abaab$. The internal work of the machine on this input is determined by a path in the graph such that **(i)** the path starts at the initial state 0 and **(ii)** the word w is read off the labels along the path.



Actions

The finite state machine as defined above is simply a **transition machine**. To make use of it, we need to add **actions**, which means activities triggered by the machine's work.

There are four types of actions:

- **entry action** is performed upon entering a state,
- **exit action** is performed upon exiting a state,
- **transition action** is performed upon specific transition,
- **input action** is performed depending on present state and input.

The input action is the most general type of action. Actions of the other types can be simulated by input actions.

A **Moore machine** is a finite state machine that uses only entry actions.

A **Mealy machine** is a finite state machine that uses only input actions.

Both kinds of machines are equivalent in terms of functionality. The Moore machines have simpler behaviour while the use of the Mealy machines allows to reduce the number of states.

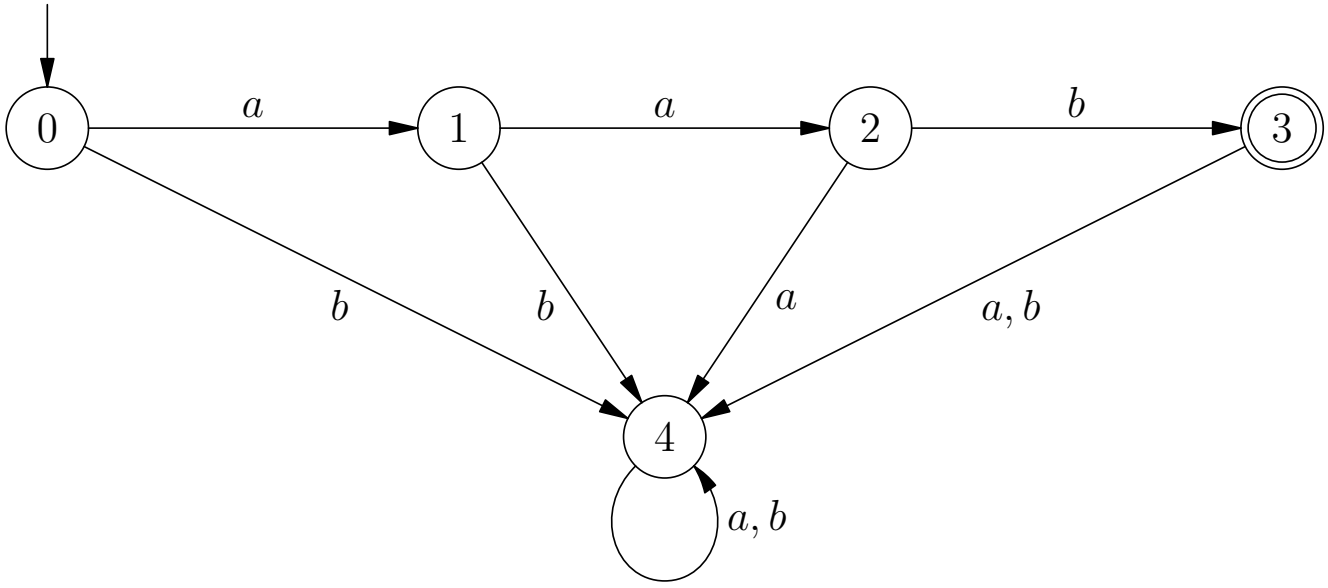
Automata

An **automaton** is a mathematical model of a finite state machine equipped with actions. There are two very different kinds of automata: **acceptors** and **transducers**.

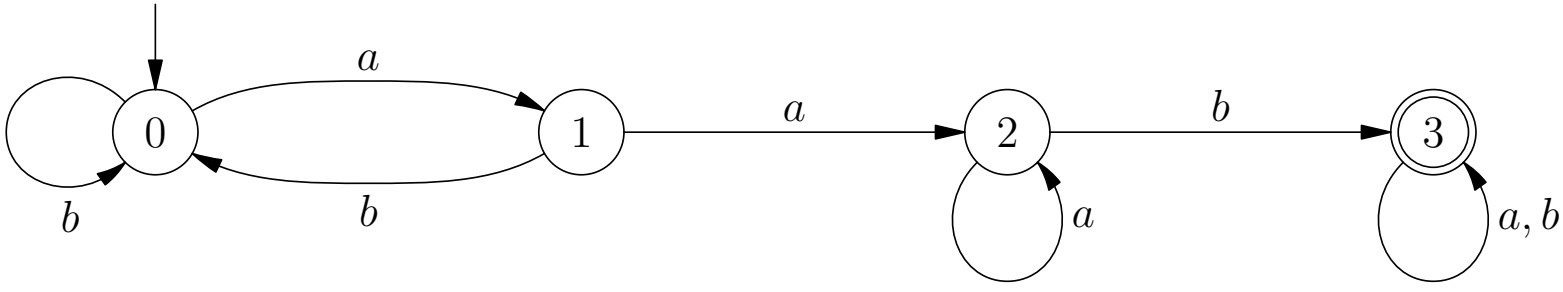
An **acceptor** is a finite state machine with a chosen subset F of states called **acceptance states**. After reading an input word, the acceptor produces a binary output (yes/no) depending on the terminal state ("yes" if the terminal state is in F and "no" otherwise).

On the Moore diagram, the acceptance states are marked with double circles.

Examples

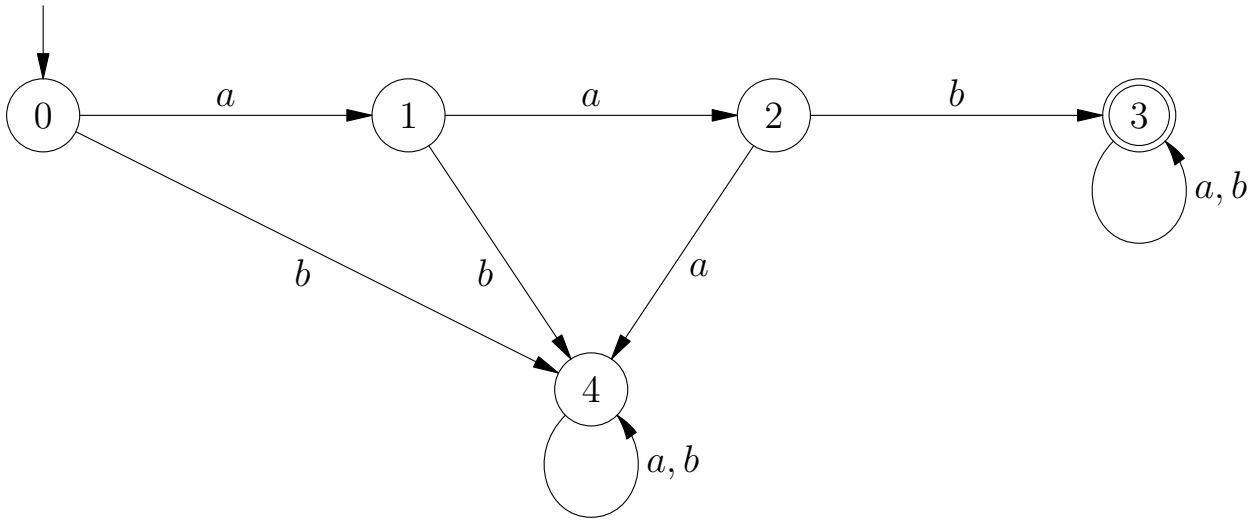


The automaton accepts only the word *aab*.

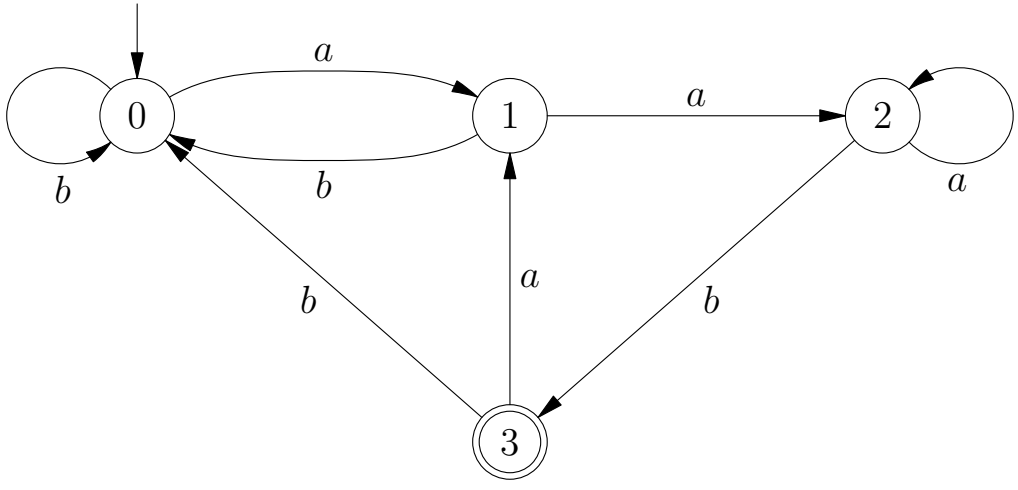


This automaton accepts words that contain subword *aab*.

Examples

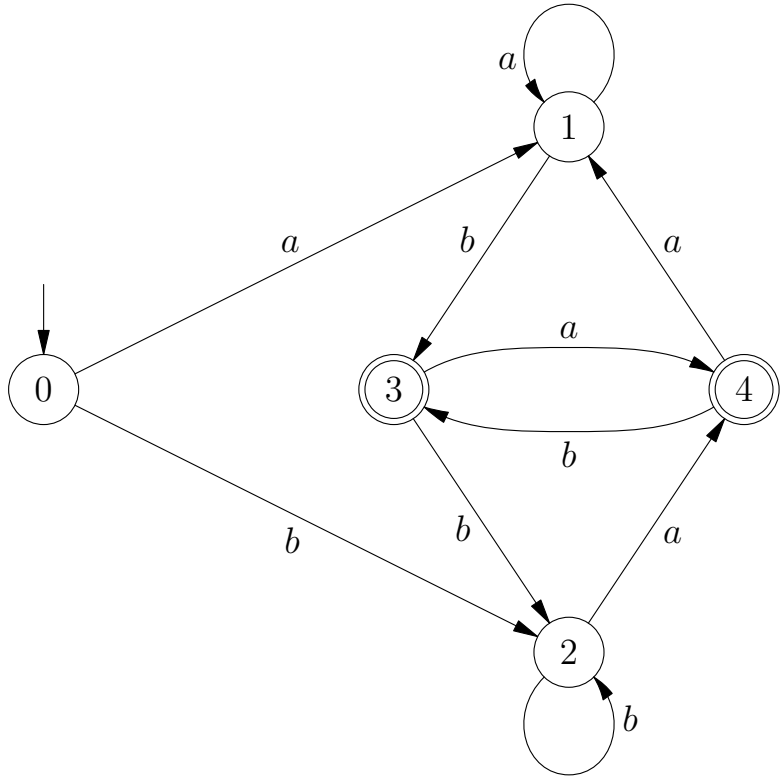


The automaton accepts all words that begin with *aab*.



This automaton accepts all words that end with *aab*.

Examples



The automaton accepts those input words in which the last two letters are distinct.

The automaton memorizes the latest two letters of input by changing its state. Namely, it is in the state 1 if the last two letters of input are *aa*, in the state 2 if the last two letters of input are *bb*, in the state 3 if the last two letters of input are *ab*, and in the state 4 if the last two letters of input are *ba*.

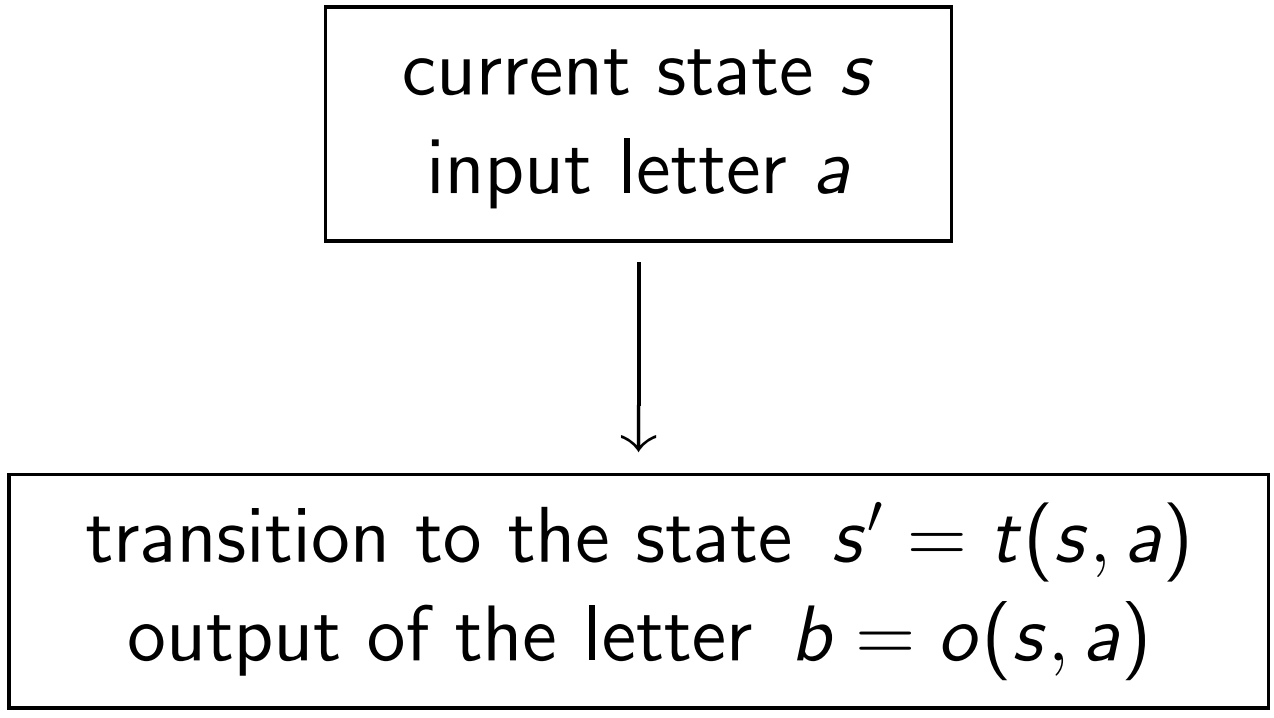
Automata: transducers

A **transducer** is a finite state machine capable of producing output upon reading a particular letter of an input word.

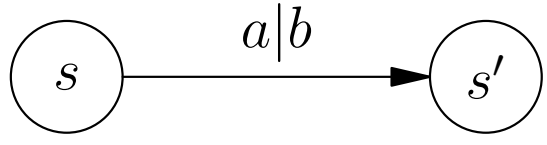
In addition to the set of states S, the initial state s_0, the input alphabet A and the state transition function t, the definition of a transducer includes an **output alphabet** B and an **output function** $o : S \times A \to B$. That is, the transducer produces one letter of output per each letter of input.

Very often, the output alphabet B is the same as the input alphabet A. In that case, the automaton gives rise to a transformation of the set A^* of finite words in the alphabet A.

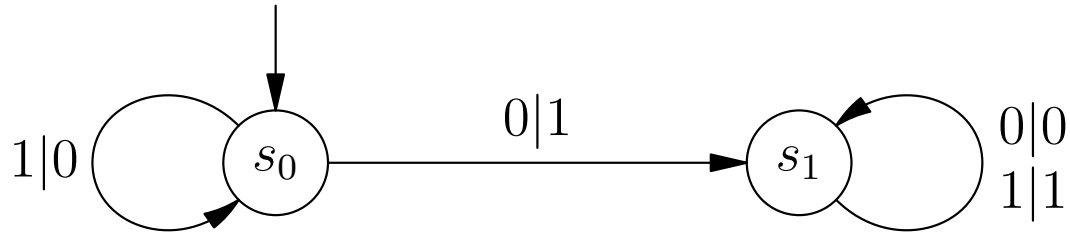
The transducers as defined above are called **synchronous**. For an **asynchronous** transducer, the output function takes values in the set B^* of finite words in the alphabet B.



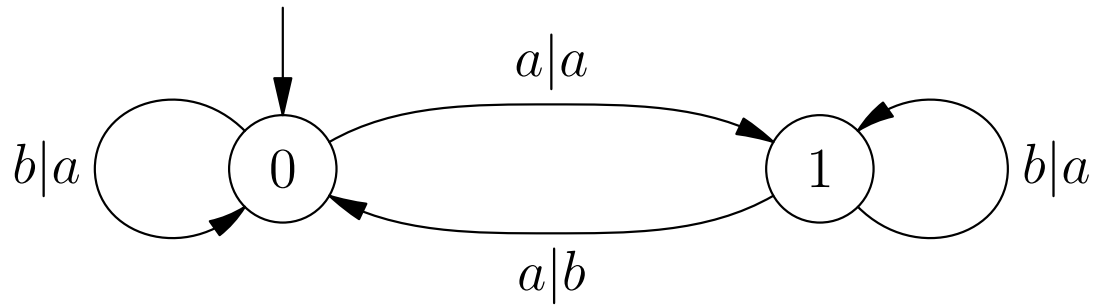
In the Moore diagram of a transducer, edges carry labels with two fields. The left (or top) field of a label is the input letter. The right (or bottom) field is the output letter.



Examples. Adding machine.



The automaton switches all letters (actually, digits) in an input word up to the first 0. When we write the word backwards, this means adding 1 to a binary number.



This automaton switches all letters *b* to *a*. Besides, every other letter *a* is switched to *b* (2nd, 4th, 6th,...).

Permutations

Let X be a finite set. A **permutation** of X is a bijection from X to itself.

Let $f : X \to X$ be a function. Given $x \in X$, the element $y = f(x)$ is called the **image** of x under the function f. Also, x is called **preimage** of y under f.

The function $f : X \to X$ is **injective** (or **one-to-one**) if any $y \in X$ has at most one preimage. The function f is **surjective** (or **onto**) if any $y \in X$ has at least one preimage. The function f is **bijective** if any $y \in X$ has exactly one preimage.

The inverse function f^{-1} is defined by the rule

$$x = f^{-1}(y) \iff y = f(x).$$

The inverse f^{-1} exists if and only if f is a bijection. If f^{-1} exists then it is also a bijection.

Theorem If X is a finite set, then the following conditions on a function $f : X \to X$ are equivalent:

- f is injective,
- f is surjective,
- f is bijective.

Examples. • The identity function $\mathrm{id}_X : X \to X$, $\mathrm{id}_X(x) = x$ for every $x \in X$.

• Let G_n be the set of invertible congruence classes modulo n, $[a] \in G_n$, and define a function $f : G_n \to G_n$ by $f([x]) = [a][x]$. Then f is a permutation on G_n (which is the key fact in the proof of Euler's theorem).