What Are Finite Automata?

What problem we are solving. Before the computer compiles a program, it needs to recognize:

- that a certain string of symbols is an identifier,
- that another string of symbols represents an integer, etc.

For example, in a code

```
int value1a = -19;
double number = +1.3;
```

we need to recognize that `value1a` is an identifier, that `-19` is an integer, and that `+1.3` is a real number.

For example, if you type in

```
int value1a = -19x;
```

the computer will immediately recognize that the sequence `-19x` is not an integer, so it will give you an error message.

How can we distinguish between sequences which are correct and sequences which are not correct?

Toy example. In principle, to describe an integer, we can use digits 0 through 9, we can use signs $+$ and $-$. In general, if we type something, we can use all the symbols on the keyboard. The computer has to deal with all possible symbols.

Before we describe how to recognize integers, let us consider a simplified version of this program, when our keyboard has only three symbols: 0, 1, and a letter a. In this case:

- if we have a sequence consisting of only 0s and 1s, this is a legitimate integer – namely, an unsigned binary integer;
- on the other hand, if a sequence of symbols that we type is empty or contains a letter a, it is not an integer.

The computer reads the symbols one by one.

- In the beginning, we have not read any symbol. We can say that we are in a start state s (s for *s*tart).

- If in this state s, we read the first symbol, then our reaction depends on what symbol we see.
  - If this symbol is 0 or 1, then what we have read so far is a binary integer: namely, 0 or 1. We can say that we are in the state i (i for *i*nteger).
  - If this symbol is a, then what we have read is not a binary integer, it is an error. We can say that we are in the state e (e for *e*rror).
- If we are in the state i – meaning that what we have read so far is an integer, and we read the next symbol, then our reaction also depends on what symbol we see:
  - If this symbol is 0 or 1, then what we have read so far is still a binary integer, so we are still in the state i.
  - In this symbol is a, then what we have read is not a binary integer, so we are in the state e.
- If we are in the state e – meaning that what we have read so far is an error – then, no matter what symbol we read, we still have an error, i.e., we are still in the state e.

In the end, after we have read all the symbols:

- if we end up in the state i, this means that we have an unsigned binary integer,
- otherwise – if we end up in a state s (after reading an empty string) or in the state e – we do not have an unsigned binary integer.

States that confirm that we have a correct string are called *final*; they are also sometimes called *accept states*. In our case, there is only one final state: the state i.

Comments.

- Please note that a *final state*:
  - is *not* a state in which we end up after we read the whole string,
  - it *is* a state indicate that the word that we reading is what we want it to be – in this case, an integer.

  After we finish reading all the symbols:
  - we may end up in a final state,
  - or we may end up in a state which is not final.
- Can a computer read two or more symbols at a time?
  - Theoretically, yes: we will study such machine later.

– However, to the best of my knowledge, these are purely theoretical concepts, real computers only read one symbol at a time.

Graphical description. For us humans, pictures help understand the material, so let us come up with a picture that captures the above description.

We want to describe:

- what are the states,
- what are the transitions between the states,
- what is the starting state, and
- what are the final states.

States are usually described by small boxes or circles with the state name inside.

- to indicate the starting state, we place an arrow coming into this state;
- to indicate a final state, we make a border thicker – e.g., make a box or a circle with double lines.

Transitions are usually described by arrows, with symbols on top.

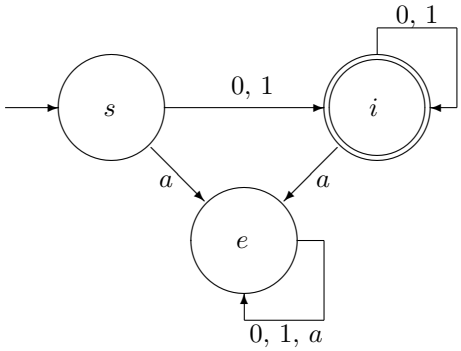
The whole picture is known as a *finite automaton.*

Linguistic comment. The word *automaton* is a Greek word, so its plural is *automata* (and *not* automatons).

This may be somewhat confusing, since:

- the Spanish term for an automaton is *automata*, and
- the plural in Spanish is *automatas*.

Toy example: graphical description. Let us see how an automaton looks for our toy example.



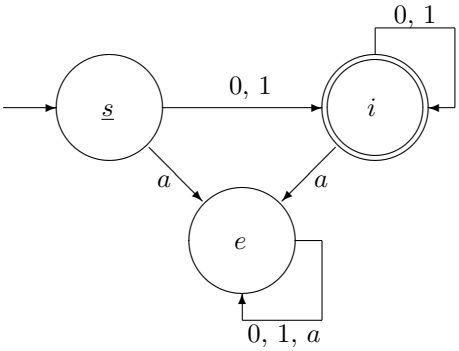
Notion of a sink state. In this automaton, once we reached the state e, no matter what symbol we see, we will stay in this state.

- Nowadays, we would call such a state – from which it is impossible to get out – a black hole.
- However, automata theory was invented before this term appeared, so it is called a *sink state*.

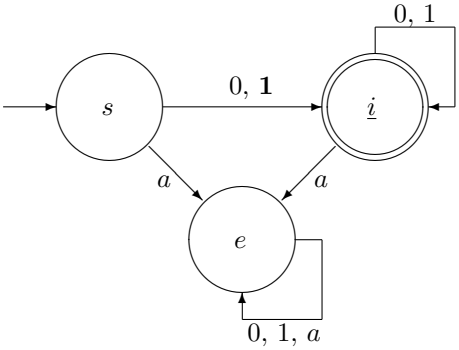
Examples. Let us trace how this automaton allows us to check whether a word is an unsigned binary integer or not. In this tracing, at each moment of time:

- the current state will be underlined,
- the symbol that is read at this moment of time, will be made bold, both in the word and in the automaton.

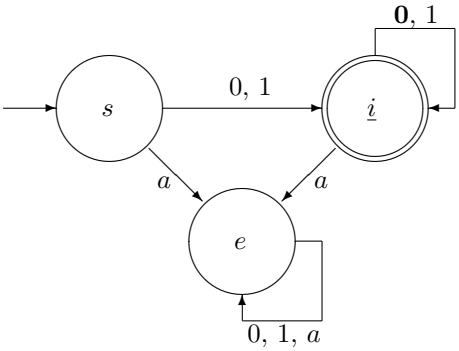
First example: the word 101. Let us show how the above automaton will recognize that 101 is an unsigned binary integer. At first, before reading any symbol, we are in the start state.



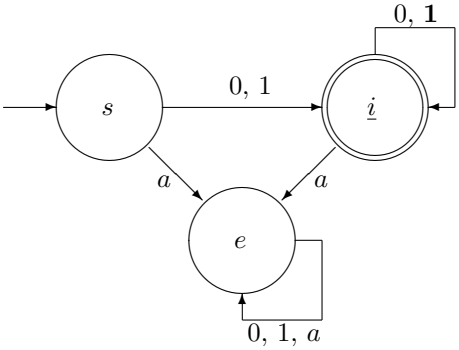
Then, we read the first symbol 1 of the word **1**01. If we are in the state s and see 1, then, according to the automaton, we should got to the state i;



After that, we read the next symbol 0: 1**0**1. According to the automaton, if we are in the state i and we see a symbol 0, we should go back to the same state i:



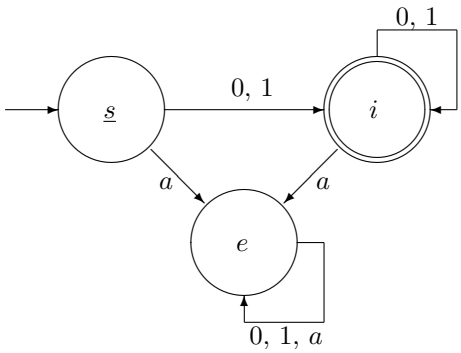
Next, we see the symbol 1: 10**1**. According to the automaton, if we are in the state i and we see the symbol 1, we should stay in the state i:



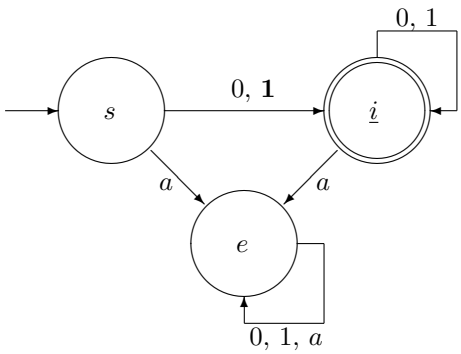
We finished reading all the symbols, and we are in the final state. This means:

- that this automaton *accepts* the word 101,
- i.e., that 101 is a unsigned binary integer.

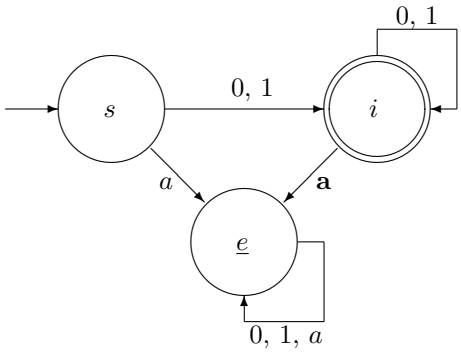
Second example: the word 1a1. In this case, we also start with the start state:



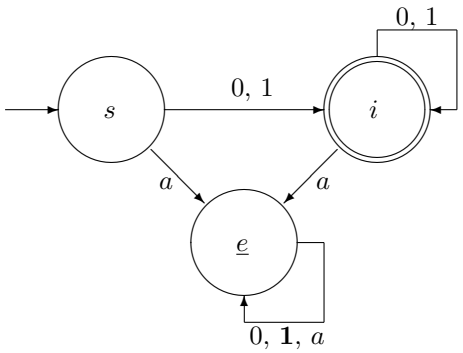
Then, we read the first symbol 1: **1**a1. Just like in the first example, this brings us to the state i:



After that, we read the symbol a, which brings us to the error state e:



The last symbol we read is the symbol 1: 1a**1**, this keeps us in the same error state e:



Now, we have finished reading, and we are in the error state e which is not final. This means:

- that the word $1a1$ is *not* accepted by this automaton, i.e.,
- that this word is *not* an unsigned integer.

Practice examples. Please practice tracing the automaton on different words: e.g., the word 01 and the word $aa0$.