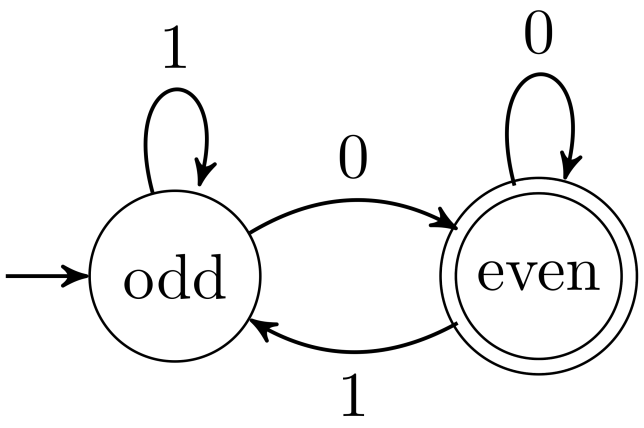
Finite Automata: The smolest machines

Clarissa Littler

Computer scientists like to study *kinds* of computers. That's not surprising, it's in the name. But what might be a little surprising is that "computer" doesn't only mean an object like a desktop or smartphone! No, the kinds of "computers" that computer scientists study include a bunch of different *theoretical* machines that are simulated with just a pen and paper, each with different abilities and restrictions on what it can do. These range from the mighty Turing machine, a nearly century old pen & paper "computer" that has infinite memory, to the tiniest, smolest, machine called a finite automata. The finite automata happen to be the subject of our article.

Finite automata are so simple, actually, that we can build and simulate one with just a few circles and lines drawn on paper. All a finite automata is capable of is processing a series of inputs one after the other and then responding "yes" or "no". Here's our first example, which is a finite automata that answers the question "is this binary number even?"



I've already implied that you can "run" this machine with a pen & paper, so let's describe how! We start at the circle on the left with the arrow pointing into it and then feed the machine one input at a time. In this case, those inputs are digits of the binary number. For each input, follow the arrows between the different circles, or *states*. If, after we've eaten all the digits of the number, we're in a double circle then the answer is "yes" but if we're in only a single circle the answer is "no".

The logic of this automata is that if a *binary* number ends in a 0 then the number is even, but if it ends in a 1 the number is odd.

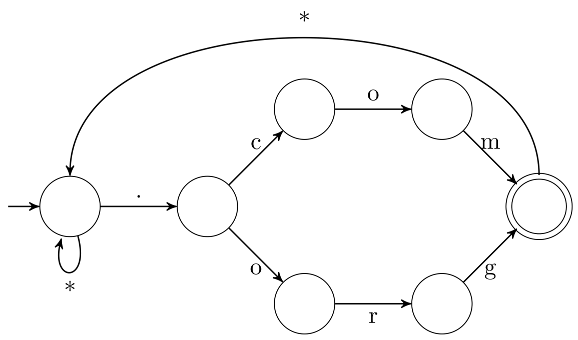
Let's try an example! Run the automata on the input "1010", which is *10* in decimal. If you need a refresher on binary, check the further reading links!

1. You start in the "odd" state and the first character is a 1, so you loop back into the "odd" state

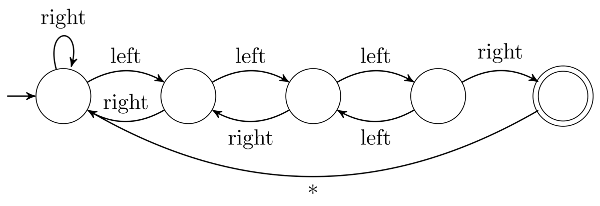
2. The next character is a 0, so you move into the "even" state
3. The next character is a 1, so you move back into the "odd" state
4. The last character is a 0, so you move into the "even" state and stop. This means the number is even!

To test yourself, try a few other binary numbers like "111", which is *7* in decimal, or "11010" which is *26* in decimal.

Okay, but what else are finite automata good for? They're used under the hood of a lot of software as very efficient ways to calculate the answers to all sorts of questions like "Does this email address end with either '.com' or '.org'?" which is solved by the following automata. Note the use of the “*” symbol as a shorthand for “any character” rather than drawing a separate loop for every letter, number, and symbol.



Or this one that tests a solution to a Legend of Zelda style maze where you have to move through a specific set of exits or get sent backwards



These automata are all just pen & paper simulations of machines but you *can* implement them inside any programming language. They’re used all the time for efficiently executing *regular expressions*, which are a popular tool for searching & matching text.

Computer scientists have another interest in finite automata, though, because we like to know how much power different models of computation have. This means we don't just want to know what it *can* do but what it *can't* do.

To that end, see what goes wrong if you try to write a finite automata that answers the question "is this word a palindrome?". Maybe think about what’s the simplest feature you could add to a finite automata that would let it answer the question.

So that's our introduction to finite automata, the simplest kind of computing machine we can define. They're shockingly powerful for how simple they are and even more shockingly close to being real computers.

Try writing some and seeing what you can do!

Further reading:

- Reference on binary: https://learn.sparkfun.com/tutorials/binary/all
- Spoilers for the kind of automata that can detect palindromes: https://en.wikipedia.org/wiki/Pushdown_automaton