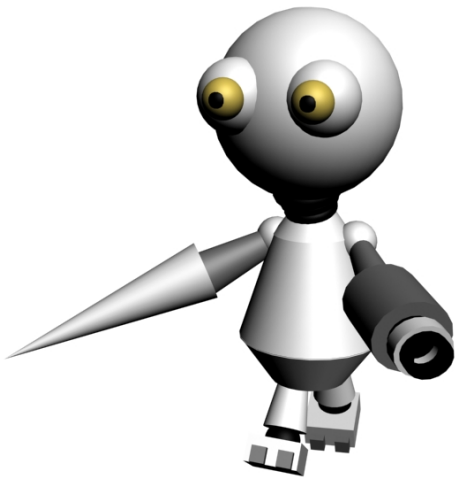
Math Circle - Automata

Today we're going to build some very simple computers, called automata[1]. Each automaton will be specifically designed to solve one task, and the tasks themselves will be pretty simple. Your automaton will take as input a string of characters. It will read in the **input string** one-character-at-a-time from left to right, and once it's finished will decide to either ACCEPT or REJECT that input.

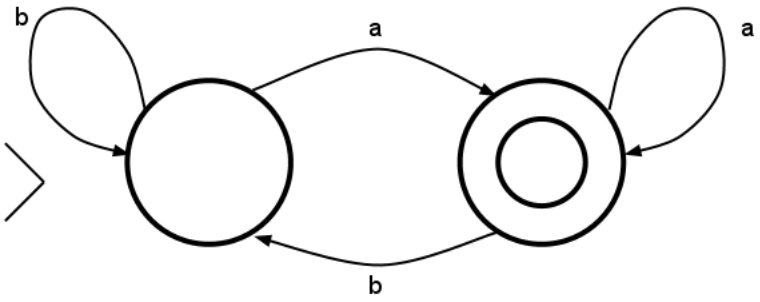
At any given point in time, after reading some or none or all of the input string, an automaton is always in one of its finitely-many **states**. Upon reading another character the automaton will *beep* and *bleep* and *bloop* and move to another state, depending only on what was just read. When it is done reading the entire input string, it will ACCEPT the string if the automaton has ended in one of its **final states**. Otherwise, the automaton with REJECT the string.

Thankfully, we can draw automata. A state is drawn as just a circle, and so an automaton at its core is just a bunch of circles. Each circle has some arrows coming out of it. These arrows tell the automaton, based on certain character inputs, which state to go to next. An arrow can point to another state, or back to the same one. A final state is depicted as a state with a smaller circle inside of it. The start state (the state the automaton is in before it gets any input) is pointed to by an arrowhead (>). Let's take a look at a few examples on the next page.



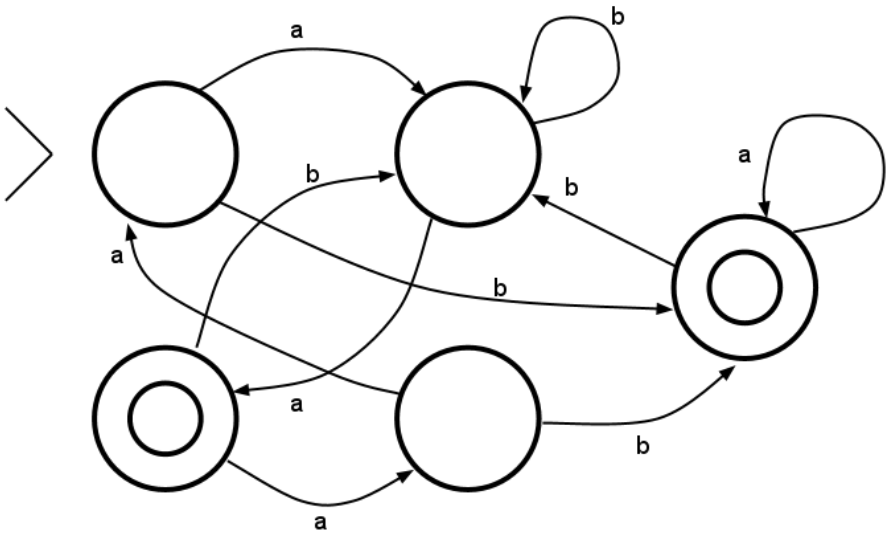
[1] *singular:* automaton

Meet Henry. Henry is one of the smaller automata, as he only has two states.



Henry takes as input a string of as and bs. No matter which of the two states Henry is in, if he sees a as the next character, he moves to the rightmost state (even if he's already in this state!). If he sees b, Henry moves to the leftmost state. The rightmost state is Henry's only final state. The leftmost state is Henry's starting state – the state he starts in before reading any input. Convince yourself that Henry ACCEPTs input strings which end in a and REJECTs those which do not.

In general automata can have many states, including many possible final states, but always only one start state.



Though, it is not always easy to give a simple description of which input strings will be ACCEPTed and which will be REJECTed.

Tasks for Automata

For each of the following, create an automaton whose input strings are strings of a and b. It should ACCEPT the specified strings, and REJECT all others.

1. Strings which have an odd number of a.
2. Strings which have an odd number of a and an even number of b.
3. Strings for which the number of a is odd and also the number of b is not a multiple of 3.
4. Strings in which any a must be immediately followed by b.
5. Strings of the form $a^n b^m a^k$ for some nonnegative integers n, m, and k.
6. Strings of the form $a^n b^m$ OR $b^n a^m$ for some nonnegative integers n and m.
7. Strings which start with a and end with b.
8. Strings which never include the smaller string aba inside of them.

For each of the following, create an automaton whose input strings are strings of a, b, and c. It should ACCEPT the specified strings, and REJECT all others.

9. Strings which have an odd number of a and an even number of both b and c.
10. Strings which are of the form $a^n b^m c^k$ for some nonnegative integers n, m, and k.
11. Strings which start with a, end with c, and contain at least one b.
12. Strings for which a comes before b, which comes before c.
13. Strings in which any a must be immediately followed by b, and any b must be immediately followed by c.

For each of the following, create an automaton whose input strings are strings of digits $0-9$, representing a nonnegative integer. It should ACCEPT the specified numbers, and REJECT all others.

14. Integers which are divisible by 5.
15. Integers which are divisible by 25.
16. Integers which are divisible by 3.
17. Integers which are divisible by 6.