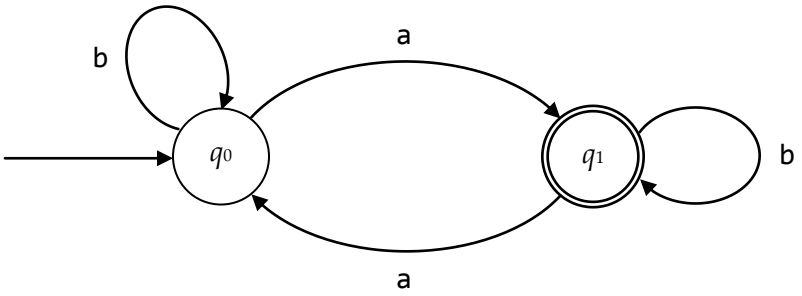
Finite automata (aw-**tom**-uh-tuh)

Dillon Zhi – LA Math Circle

1. This, believe it or not, is a picture of a machine:



The two circles represent the **states** of the machine. Here, the states are labeled q_0 and q_1. You feed the machine a **string** of `a`'s and `b`'s, such as `aaba`. The machine processes this string. It starts in the state which has the unlabeled arrow pointing to it. Then it **transitions** from state to state by following the arrows that correspond to the **letters** in the string.

In this example, the machine:

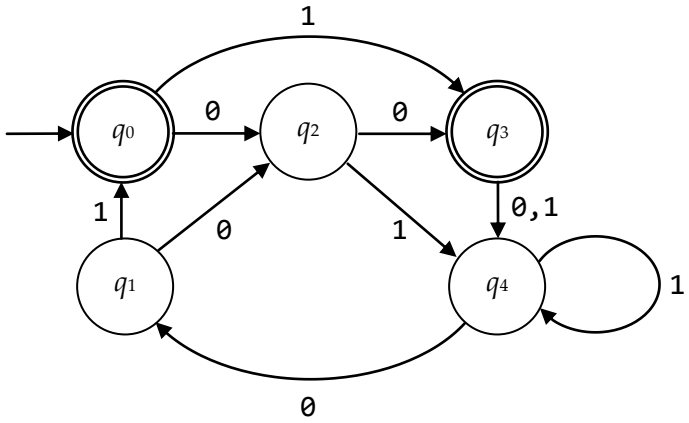
- starts in state q_0
- follows the `a` arrow from state q_0 to state q_1
- follows the `a` arrow from state q_1 to state q_0
- follows the `b` arrow from state q_0 to state q_0
- follows the `a` arrow from state q_0 to state q_1

After processing the string, the machine **accepts** it if the machine ended up in an **accept state** (shown with a double circle) and **rejects** it if not.

a. Does the machine above accept `aaba`?

2. In the machine shown in the state diagram below:

a. What sequence of states does the machine go through if you input `11001`?



b. Does the machine accept `11001`?

c. The Greek letter ε (lowercase epsilon) is the traditional symbol for the **empty string**. Does the machine accept ε?

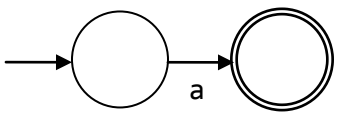
3. If you were paying attention to the title, you may have realized these machines are called **finite automata**. *Finite* means the number of states is finite. *Automata* is what these *self-operating machines* will make you puny humans call them after they take over th... never mind. (**Automaton** is singular and **automata** is plural, just like *mitochondrion* and *mitochondria*.)

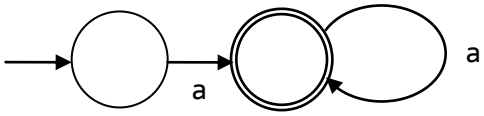
   More specifically, they're called **deterministic** finite automata (**DFAs**). Perhaps someday you just might learn about *nondeterministic* finite automata too.

   Here are some important notes about DFAs:

   - A DFA has an **alphabet**, the set of **letters** it takes. In number 1, the alphabet was the set {`a`, `b`}.
   - **Each state of a DFA has exactly one arrow leading out of it for each letter of the "alphabet".**

   a. (really easy) In number 2, the "alphabet" was _____.
   b. (also really easy) Which of these is not a DFA?





4. Draw a DFA that accepts the strings over the alphabet {`Quarter`, `DollarBill`} whose total monetary value is at least $1.75.

5. Draw a DFA that accepts the strings over the "alphabet" {`+`, `-`} such that the number of `+`'s minus the number of `-`'s in the string is a multiple of 5.

6. The set of strings accepted by an automaton is also called the **language recognized by** the automaton.
   a. Give a *simple* description of the language recognized by the DFA in number 1.

7. Draw a DFA with 6 states recognizing the language of strings over the alphabet {`a`, `b`} that contain `abbaa` as a substring at least once.

8. Draw a DFA recognizing the language of strings over the alphabet {`0`, `1`} that end with either `00` or `11`.

9. Draw a DFA recognizing the language of strings over {`0`, `1`} that *don't* end with either `00` or `11`.

10. A language (i.e. a set of strings over a given alphabet) is called **regular** if there is some DFA that recognizes it. So all the languages we've looked at so far are regular.
    a. Show that if L is a regular language, so is its complement. (The complement is the set of strings over the same alphabet that are not in L.)

       (Psst: Let's say you have an DFA which recognizes L. How do you use it to construct a DFA that recognizes the complement of L?)

So now you know a little bit about deterministic finite automata. But wait, there's more!

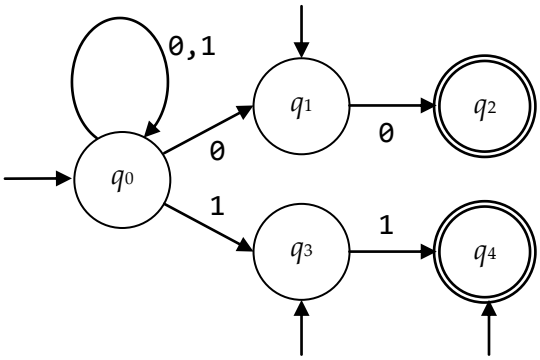
A. Read section 2 to find out about **nondeterministic finite automata**.
B. Can *English* be recognized by a finite automaton? **What languages *aren't* regular**?

To find out, go to section 3!

2. Nondeterministic finite automata

11. So the day has come, young apprentice, for you to learn the ways of ~~the dark side~~ nondeterministic finite automata (NFAs).

Here, the word "nondeterministic" doesn't mean "random", and it doesn't mean anything spooky.



The figure shows an NFA. The key difference is that NFAs don't have to follow this rule about DFAs:

- Each state of a DFA has exactly one arrow leading out of it for each letter of the "alphabet".

In this NFA, q_0 has two arrows marked `1` exiting it, while q_1 has none!

NFAs are also allowed to have **multiple start states**. Here, there are four.

The machine accepts a string if *some sequence of transitions* corresponding to the letters in the string leads from a start state to an accept state.

Example: The machine accepts `00`. How? It:

- starts in state q_0
- follows the `0` arrow from state q_0 to state q_1
- follows the `0` arrow from state q_1 to state q_2

Note that there are other paths it could have followed. For example:

- start in state q_0
- follow the `0` arrow from state q_0 to state q_0
- follow the `0` arrow from state q_0 to state q_1

Or:

- start in state q_3
- … have nowhere to go (ok, so this one isn't really a path)

But as long as at least one of the paths ends in an accept state (in this case, the path from q_0 to q_1 to q_2), it accepts the string.

a. What language does this NFA recognize?

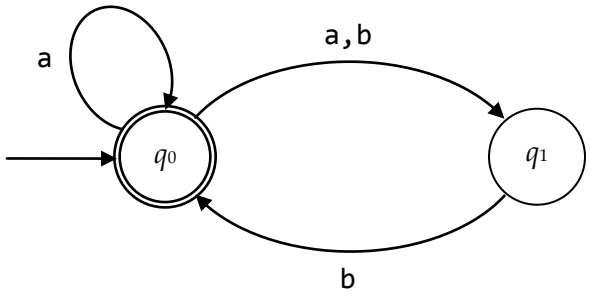
12. Draw an NFA with 6 states that recognizes the language of strings over {`a`, `b`} that contain `abbaa` as a substring at least once.

    Note that **every DFA is also a NFA** (why?), so your answer to number 7 would work for this too. *You must take advantage of the special features of NFAs – if you do, your answer to this question will be simpler and have fewer arrows than your answer to number 7.*

13. Draw an NFA that recognizes the language of strings over {`a`, `b`, `c`, `d`} that don't contain all four letters (for example, `abcabc` is accepted, but `abcdabc` is not). Try to use as few states as possible!

14. Draw a DFA that recognizes the same language. Again, try to use as few states as possible. Don't worry, I left you plenty of space.

15. a. Describe a procedure that allows you to take any NFA with n states and convert it into a DFA with 2^n states that recognizes the same language. Try to be as precise as possible.



    b. Demonstrate it on the NFA shown.

16. Number 15 shows that if a language is recognized by some NFA, then it is recognized by some DFA, i.e. it is a regular language. **Even though NFAs are more general than DFAs, they still recognize the same class of languages** (the regular languages).
    a. Make sure you understand why.

The next few problems explore several nice properties of the class of regular languages.

17. Show that the **union** of any two regular languages is regular. (The union of two languages L_1 and L_2, denoted $L_1 \cup L_2$, is the set of all strings that belong to at least one of the languages.)

    (Psst: Let's say you have an NFA which recognizes L_1 and another NFA which recognizes L_2. How do you put them together to make an NFA which recognize the union?)

18. The **concatenation** of two languages L_1 and L_2 is denoted L_1L_2. It consists of all strings that are formed by taking some string in L_1 and appending some string in L_2 to it.

    (So for example, if L_1 is the language of strings that only contain `a`'s, and L_2 is the language of strings that only contain `b`'s, then L_1L_2 is the language of strings of the form `aaa…abbb…b`, with any number of `a`'s and `b`'s.)

    Show that the concatenation of any two regular languages is regular.

19. The "**star**" of a language L is denoted L^*. It consists of all strings that are formed by putting together any number of strings that are in L. "Any number" means 0 or more, so the empty string ε is automatically in L^*.

    (So for example, if L is the language {`ab`, `a`}, then L^* the language of strings that don't have any consecutive `b`'s.)

    Show that the "star" of any regular language is regular.

20. Draw an NFA that recognizes the language ({`101`}*{`01`}*) ∪ {`1`}.

3. Nonregular languages

Some languages can't be recognized by *any* DFA – that is, they are nonregular.

21. But before we do that, try this first:
    For each $m > 0$, let L_m be the language of strings over the alphabet {+, -} such that the number of +'s minus the number of -'s in the string is a multiple of m.
    Use the "pidgeonhole principle" to show that any DFA with fewer than m states cannot recognize L_m.

22. Let L be the language over of strings over the alphabet {+, -} such that the number of +'s equals the number of -'s.
    Show that L cannot be recognized by any DFA, i.e. L is not a regular language.

23. Let L be the language of strings of the form aa…abb…b, where the number of a's is equal to the number of b's. (In symbols,

$$L = \{\mathtt{a}^k\mathtt{b}^k : k \geq 0\}.$$

The notation $\mathtt{x}^k$ means k consecutive x's.) Show that L is not a regular language.

24. Let L be the language of strings over the one-letter alphabet {a} whose lengths are perfect squares. That is, $L = \{\varepsilon, \mathtt{a}, \mathtt{aaaa}, \mathtt{aaaaaaaaa}, \ldots\}$. Show that L is not a regular language.

25. Is English a regular language? Why? (Hmmm…)