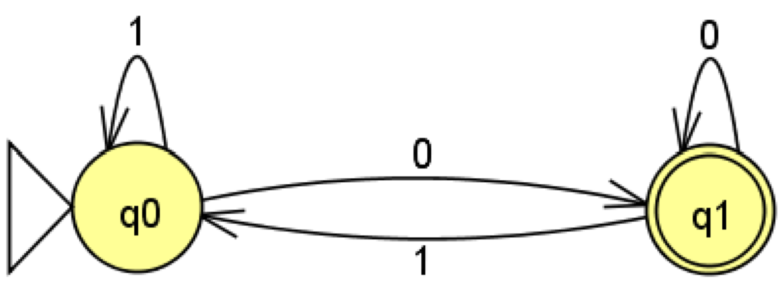
Deterministic Finite Automata

Automata are conceptual models of computers. There are many kinds of such models, depending on several aspects such as whether or not memory is allowed/is finite, etc.

The simplest form among all automata is **Deterministic Finite Automata (DFA)**. A DFA can be formally described as a 5-tuple $(Q, \Sigma, q_0, F, \delta)$:

- Q is a **finite set of states**;
- Σ is the **alphabet** (so that all inputs are strings of this alphabet);
- $q_0 \in Q$ is the **initial state**;
- $F \subseteq Q$ is the **set of final states**;
- $\delta : (Q \times \Sigma) \to Q$ is the **state transition function**.
  - Its domain is the cartesian product of the set of states and the alphabet.
  - Its codomain is the set of states.
  - In combination, this function specifies ***which state the DFA goes to next*** for *every possible state* and *every possible input symbol*.

Let's look at a concrete example. Here is an illustration of a DFA that "accepts" even binary numbers:



- $Q = \{q0, q1\}$ (there are only two states);
- $\Sigma = \{0, 1\}$ (the context is about binary numbers);
- $q0$ is the **initial state**;
- $\{q1\} \subseteq Q$ is the **set of final states** (just one);
- $\delta : (Q \times \Sigma) \to Q$ is represented by the following table:

	0	1
$q0$	$q1$	$q0$
$q1$	$q1$	$q0$

Basically, regardless of which state the DFA is in, it goes to (or stays at) $q0$ when the next input symbol is 1, and goes to (or stays at) $q1$ whenever the next input symbol is 0.

This DFA *"accepts"* even binary numbers in the following sense: if we feed all the digits in a binary number into this DFA, one digit at a time, the DFA ends at the final state $q1$ **if and only if** the last digit is a 0, which is equivalent to saying the binary number is even.

You may want to "simulate" a few inputs (e.g., $101, 1011, 100, 110110$) to get a sense of how this DFA really works. *Remember, before any input, the DFA is at the start state.*

From the above example, we can see that:

- DFAs are deterministic because the state transition **function** δ fully specifies, for each state and each input symbol, **exactly one** next state that the DFA should go to.
  - If this is instead just a **relation**, that means given a particular current state and an input symbol, the FA can go to one of multiple possible next states. That makes it a *Non-deterministic Finite Automata (NFA)* which is beyond what we can discuss in 230.
- DFAs are finite because every of the five elements of it is finite.
- **The set of strings that make the DFA end at a final state is a language.** For a DFA $\mathbf{M}$, we denote $L(\mathbf{M})$ the set of all strings on Σ "accepted" by $\mathbf{M}$.
  - For the example above, $L(\mathbf{M}) = \{0, 1\}^* \circ 0$ (convince yourself that this language characterizes even binary numbers.)
  - Not all languages can be recognized by a DFA.
  - We say a language is *regular* if and only if it is accepted by some DFA. This is the same "regular" as in "regular expressions".

Kleene's Theorem completely characterizes what languages are regular:

> (Kleene's Theorem.) Fix the alphabet Σ.
>
> - The empty language $\varnothing$ is regular.
> - For each symbol $a \in \Sigma$, $\{a\}$ is regular.
> - If L is regular, then L^* is also regular.
> - If L_1 and L_2 are regular, then $L_1 \cup L_2$ and $L_1 \circ L_2$ are both regular.

We can see that Kleene's Theorem is in fact a recursive definition of the set of regular languages. You may want to verify that the language of even binary numbers, $\{0, 1\}^* \circ 0$, is indeed regular.