Section 12.2 – Finite State Automata

Example 12.2.1 A Simple Vending Machine

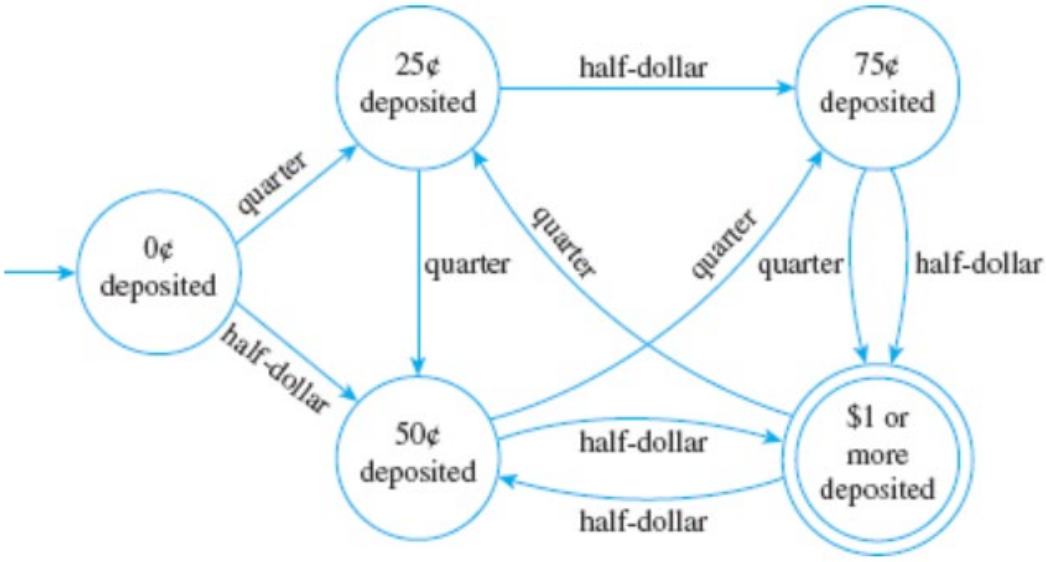
A simple vending machine dispenses bottles of juice that cost \$1 each. The machine accepts quarters and half-dollars only and does not give change. As soon as the amount deposited equals or exceeds \$1 the machine releases a bottle of juice. The next coin deposited starts the process over again. The operation of the machine is represented by the diagram of Figure 12.2.1.

We can represent the operation of this vending machine with a Finite State Automata (FSA). Each input causes the FSA to change states.

In this example, we have two inputs: 25 cent and 50 cent coins.

There are several states for this vending machine: 0 cents, 25 cents, 50 cents, 75 cents, and 100 cents (or \$1).

The state-transition diagram shows each state and how given a current state, and an input, what the resulting state will be.



The state with the arrow, →, 0 cents, is the ***initial state***, and the state with the double circle, 100 cents (or \$1) is the ***accepting state***.

We could also represent the operation of the vending machine with a Next-State Table:

		Input	
		Quarter	Half-Dollar
→	0¢ deposited	25¢ deposited	50¢ deposited
	25¢ deposited	50¢ deposited	75¢ deposited
State	50¢ deposited	75¢ deposited	\$1 or more deposited
	75¢ deposited	\$1 or more deposited	\$1 or more deposited
◎	\$1 or more deposited	25¢ deposited	50¢ deposited

So in general,

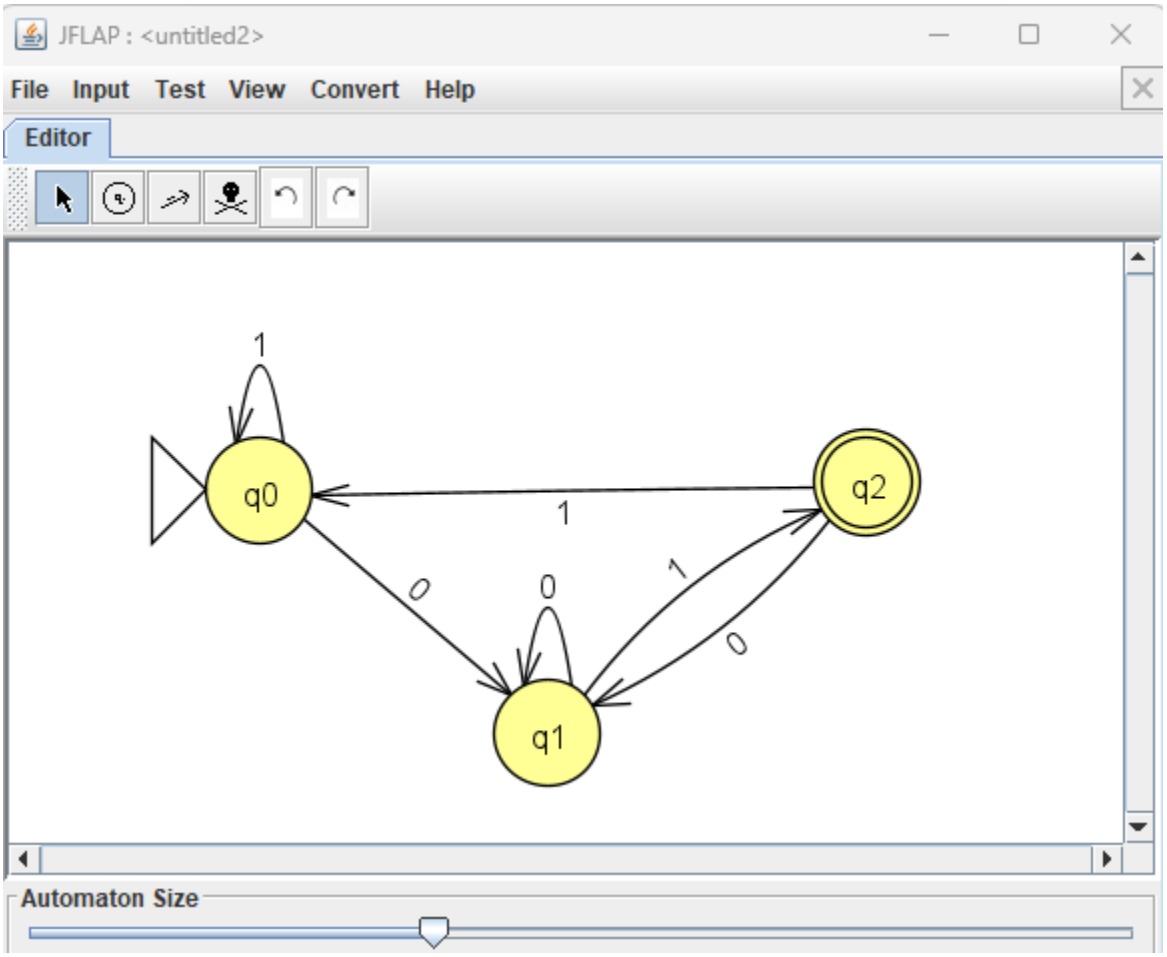
A **finite-state automaton** A consists of five objects:

1. A finite set I, called the **input alphabet**, of input symbols.
2. A finite set S of **states** the automaton can assume.
3. A designated state s_0 called the **initial state**.
4. A designated set of states called the set of **accepting states**.
5. A **next-state function** $N: S \times I \rightarrow S$ that associates a "next-state" to each ordered pair consisting of a "current state" and a "current input." For each state s in S and input symbol m in I, $N(s, m)$ is the state to which A goes if m is input to A when A is in state s.

Consider the language with input alphabet $\sum = \{0\ ,\ 1\}$, that accepts all strings ending in 01.

First, what is the Regular Expression for this language? $(0|1)^*01$

Next, can you create a state-transition diagram for it?



Here we used JFLAP, which is used to explore formal languages and implement FSAs. Read the website for it, https://www.jflap.org/ . It can be used to implement FSAs. It's pretty straightforward to use, simply download the JFLAP . `jar` file.

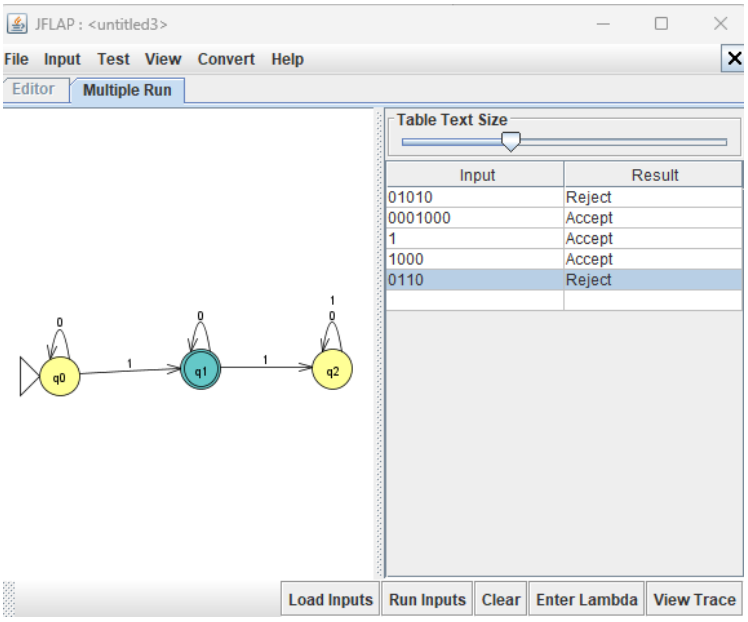
For each of the following are languages with input alphabet $\sum = \{0\ ,\ 1\}$. Give the FSA and Regular Expressions for:

1. The language accepting all strings with exactly one 1.
2. The language accepting all strings ending in 011.
3. The language accepting all strings beginning with 10.

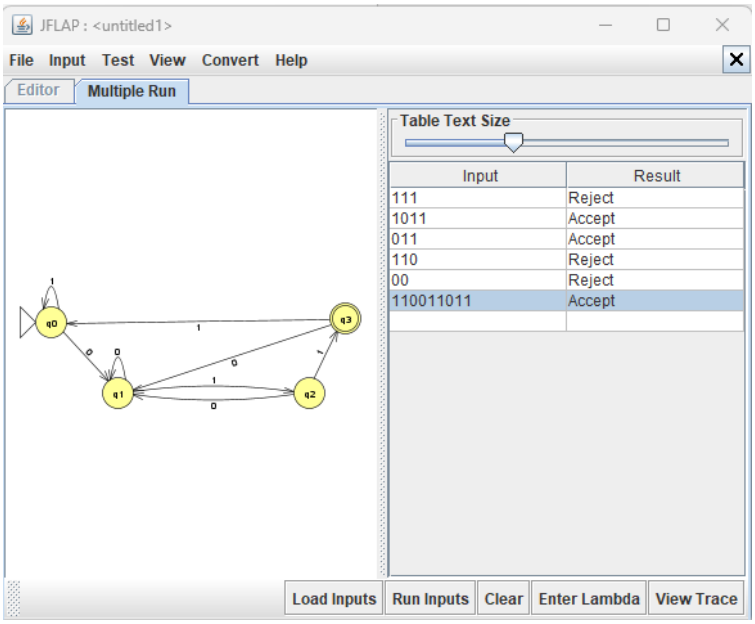
We visualize these with JFLAP, and note that each is run with several test inputs.

==========

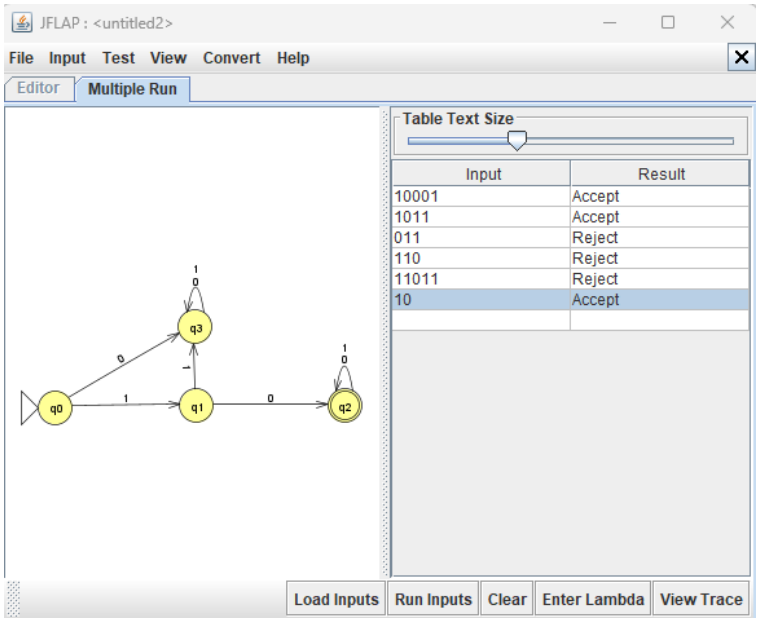
1. The language accepting all strings with exactly one 1.
0^*10^*



2. The language accepting all strings ending in 011.
$(0|1)^*011$

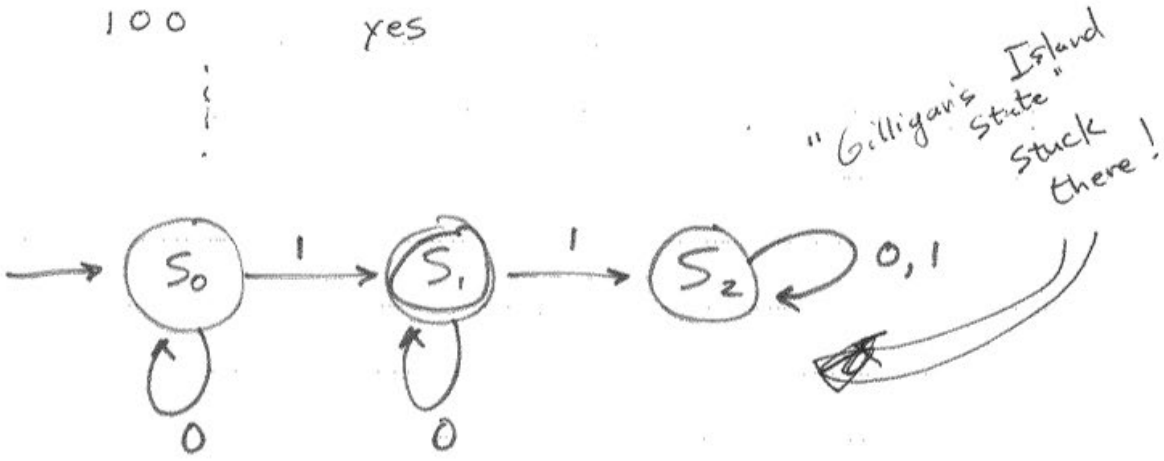


3. The language accepting all strings beginning with 10.
$10(0|1)^*$

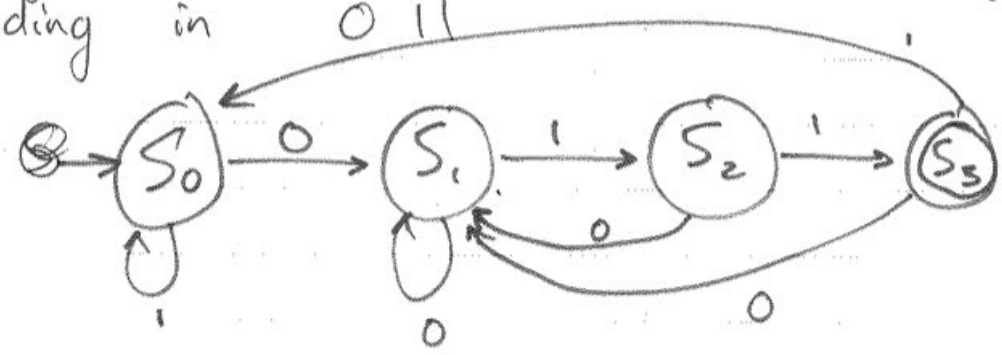


(1) Design an FSA that accepts all strings of 0's & 1's with exactly one 1.

0 111 NO
0000100 yes
100 yes
⋮



(2) Design an FSA which accepts all strings ending in 0 11



(3) Design an FSA that accepts all strings of 0's + 1's beginning w/ 10