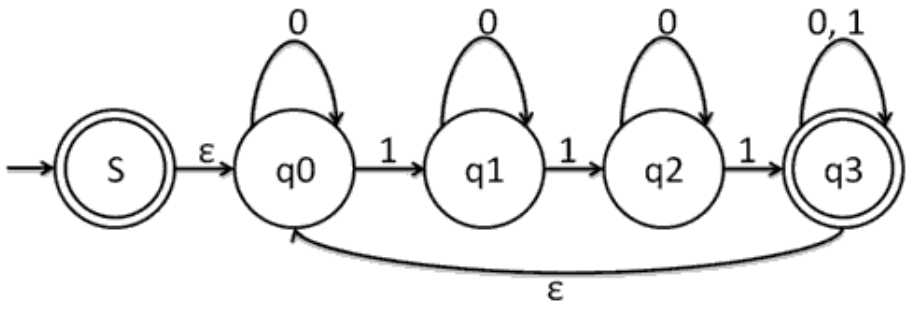
FSMs



Modeling Computation

- Modeling computation helps us to answer questions such as
  - Given a task, can it be done using a computer?
  - If so, how can the task be carried out?

- We can study computation using different structures
  - Grammars
  - Finite State Machines
  - Turing Machines

- In this class, we will touch briefly on Finite State Machines

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Finite State Machines

Many types of machines, including computer components, can be modeled using **finite state machines.**

The basis for many types of programs:

- spell checkers
- grammar checkers
- searching text
- speech recognition
- markup language transformation (HTML, Markdown, LaTeX, etc.)
- network protocols
- ...

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Finite State Machines

Different types of FSMs can model different things.

We will look at two types of FSMs:

1. Produce output
2. Recognize a language / bitstring / sequence of inputs / property

Finite State Machines

All FSMs include:

- Finite set of **states**
- **Starting state**
- **Input alphabet or input actions**
- **Transition function**

The FSM is given a sequence of inputs. The **transition function** determines the next state based on the current state and the current input.

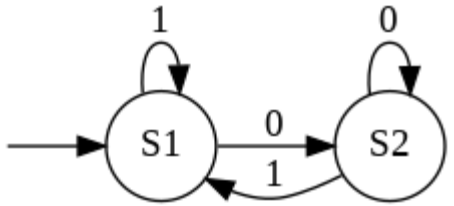
The transition function takes an ordered pair $(state, input)$ and returns a $state$.

Q	Set of states
$q_0 \in Q$	Start state
I	Set of input actions / symbols
$\delta: Q \times I \rightarrow Q$	Transition function

Example:

$$Q = \{S1, S2\}$$
$$q_0 = S1$$
$$I = \{0,1\}$$
$$\delta = \{((S1,0), S2), ((S1,1), S1), ((S2,0), S2), ((S2,1), S1)\}$$

We can also represent a transition function using a state diagram:

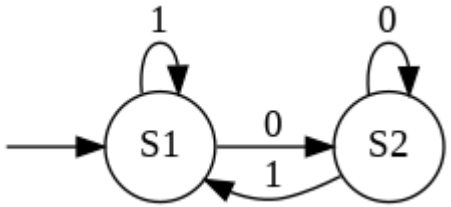


Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Finite State Machines

Example:

$$Q = \{S1, S2\}$$
$$q_0 = S1$$
$$I = \{0,1\}$$
$$\delta = \{((S1,0), S2), ((S1,1), S1), ((S2,0), S2), ((S2,1), S1)\}$$



Example Input: 1001101

Which is the last state given this input?

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

FSM with Output

Some FSMs produce **output** from an output alphabet or output set.

The **output** is determined based on the current state and the current input.

Q	Set of states
$q_0 \in Q$	Start state
I	Set of input actions / symbols
$\boldsymbol{O}$	**Set of outputs**
$\delta: Q \times I \rightarrow Q \times \boldsymbol{O}$	Transition function

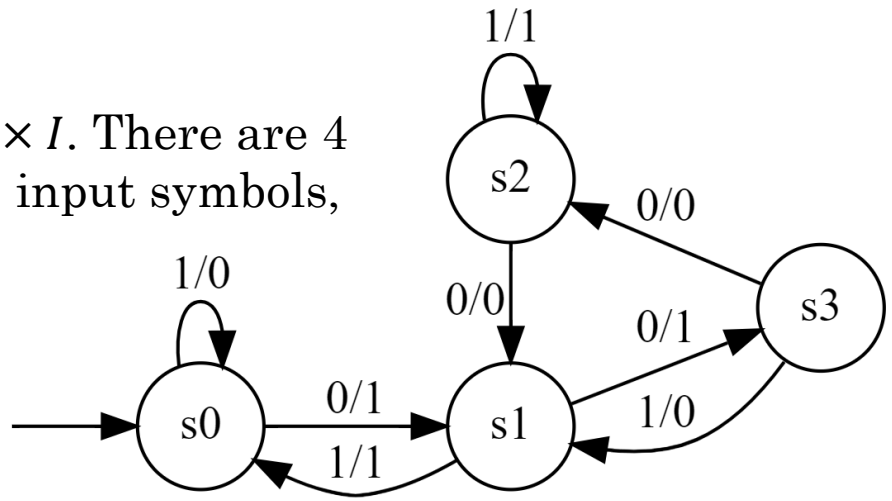
Example:

$$Q = \{s0, s1, s2, s3\}$$
$$q_0 = s0$$
$$I = \{0,1\}$$
$$O = \{0,1\}$$
$$\delta = \{$$
$$(s0,0) \rightarrow (s1,1),$$
$$(s0,1) \rightarrow (s0,0),$$
$$(s1,0) \rightarrow (s3,1),$$
$$(s1,1) \rightarrow (s0,1),$$
$$(s2,0) \rightarrow (s1,0),$$
$$(s2,1) \rightarrow (s2,1),$$
$$(s3,0) \rightarrow (s2,0),$$
$$(s3,1) \rightarrow (s1,0)$$
$$\}$$

How many ordered pairs are in the **domain** of the transition function δ?

The domain is $Q \times I$. There are 4 states in Q and 2 input symbols, so 4x2=8

Go to the whiteboards and draw this FSM



What would be the output on the input of 1010101? 0111111

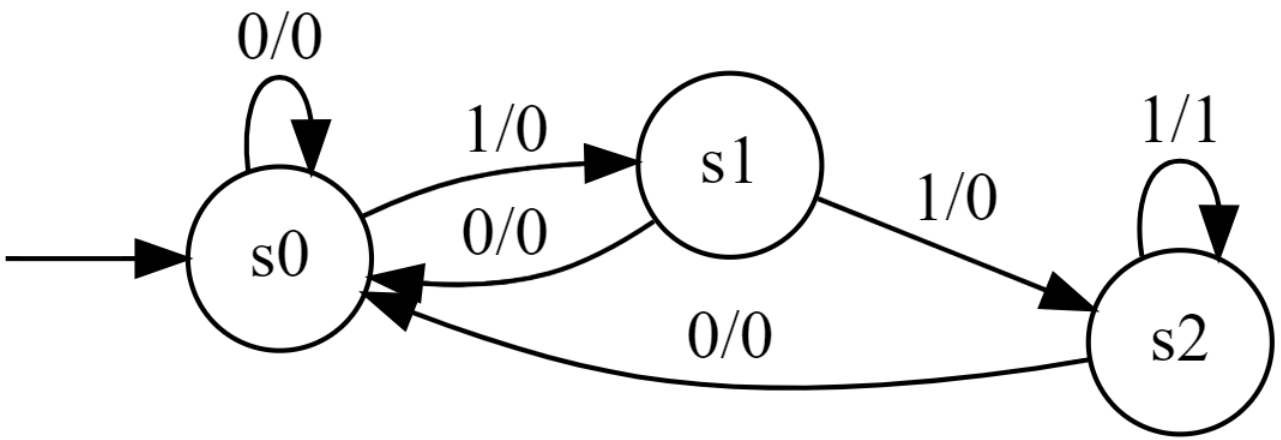
Input of 1100101? 0011010

FSM with Output: Example

Imagine a coding scheme in which the receiver of a message knows there has been a transmission error whenever three consecutive 1s appear. Let's build an FSM that will output a 1 whenever it detects three consecutive 1s and a 0 otherwise.

An FSM can only remember its current state. So how many states will we need to "remember" three bits? **Three**

Let's draw it.



What would be the output on the input of 101101?

000000

On an input of 101110?

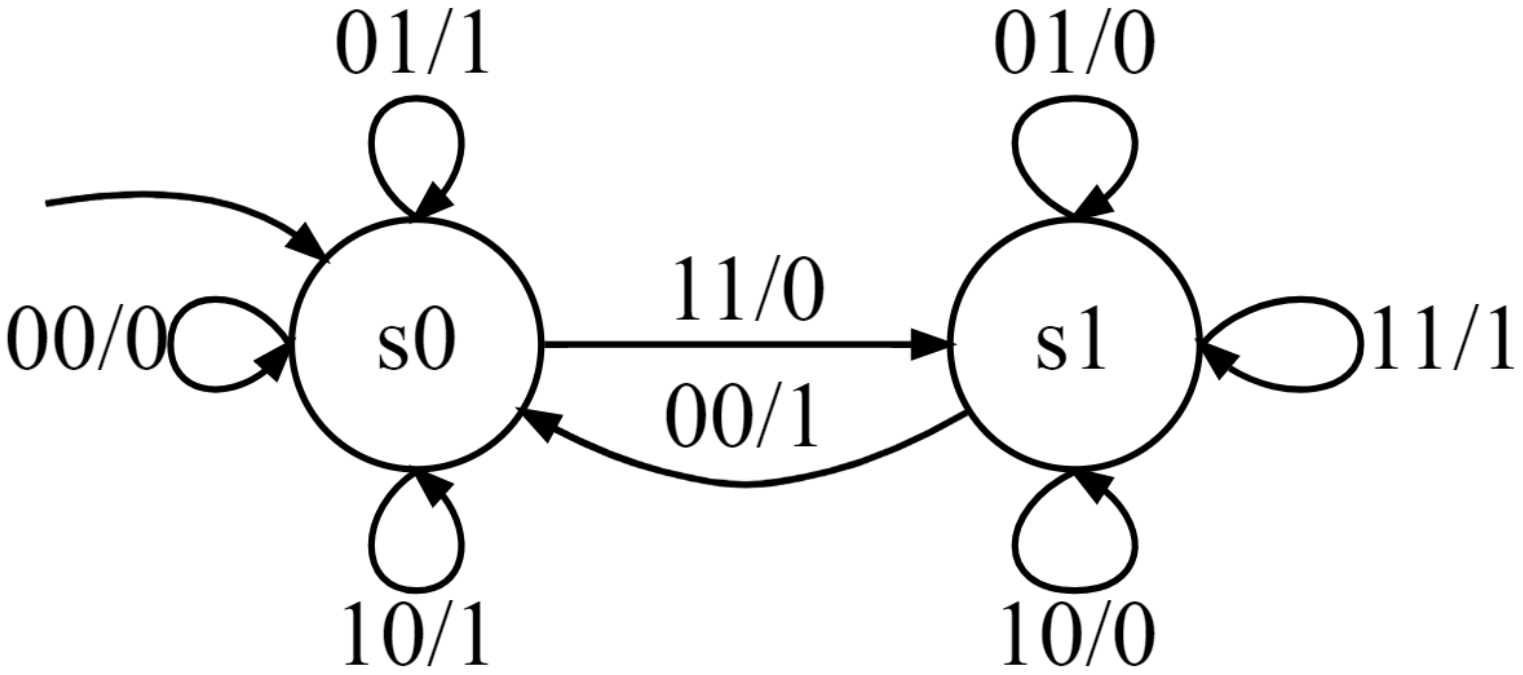
000010

A "1" indicates the receiver detected an error

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

Example: Add two binary numbers

$$
\begin{aligned}
Q &= \{s0, s1\} \\
q_0 &= s0 \\
I &= \{00,01,10,11\} \\
O &= \{0,1\} \\
\delta &= \{ \\
&\quad (s0,00) \rightarrow (s0,0), \\
&\quad (s0,01) \rightarrow (s0,1), \\
&\quad (s0,10) \rightarrow (s0,1), \\
&\quad (s0,11) \rightarrow (s1,0), \\
&\quad (s1,00) \rightarrow (s0,1), \\
&\quad (s1,01) \rightarrow (s1,0), \\
&\quad (s1,10) \rightarrow (s1,0), \\
&\quad (s1,11) \rightarrow (s1,1) \\
&\quad \}
\end{aligned}
$$



This FSM takes an input consisting of 2 bits at a time and outputs one bit. Now we can add two binary numbers.

What is the result of adding these two numbers by hand?

0110
<u>0011</u>
1001

Now trace these through the FSM, taking two bits at a time (one from each string from right to left). Does it produce the same result?

Rosen, Kenneth. *Discrete Mathematics and Its Applications, 7th Edition*. McGraw-Hill, 2012

FSMs that Recognize Properties

Some FSMs recognize whether a sequence of inputs has a particular property.

Instead of producing an output, these FSMs have one or more **accept** states.

If an input string ends on an **accept** state, we say the input has the property recognized by the FSM, or that the input is a valid string in the language of the FSM.

Can be used to represent a **regular language** in a graphical way.

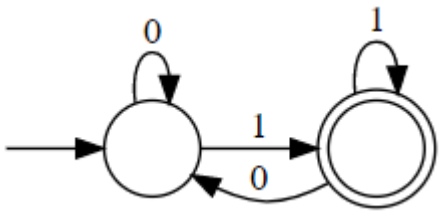
Example:

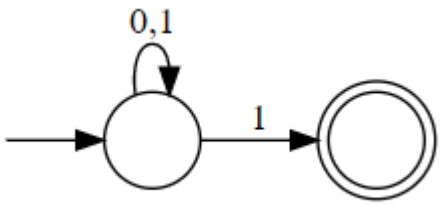
The language of all bitstrings that end with a 1:

"01", "11", "11111", "01010101", ...

Regular expression: (0 | 1)*1

Q	Set of states
$q_0 \in Q$	Start state
I	Set of input actions / symbols
$\boldsymbol{A \subseteq Q}$	**Accepting state(s)**
$\delta: Q \times I \rightarrow Q$	Transition function





These types of FSMs are also known as **Finite State Automata**

Finite State Automata

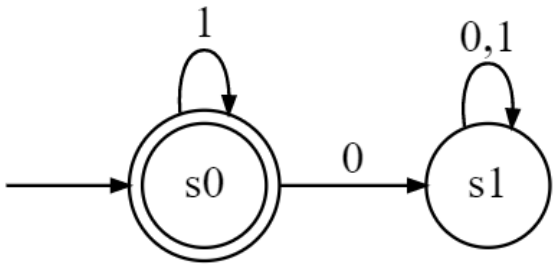
Do not produce output.

Have a set of final states (accepting states).

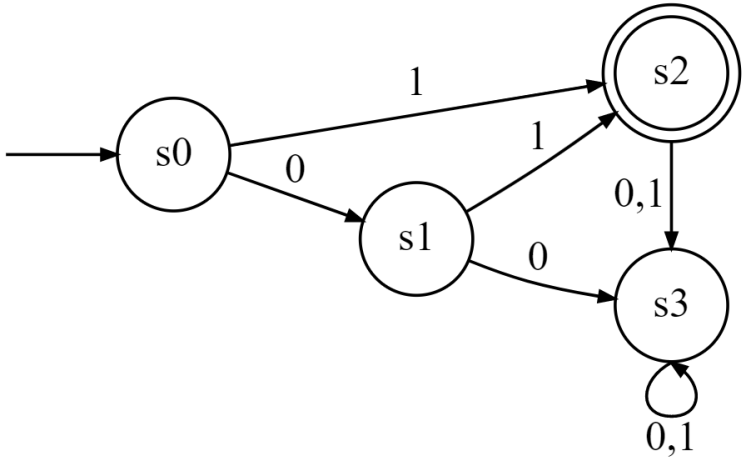
Accepting states are shown using a double circle.

Q	Set of states
$q_0 \in Q$	Start state
I	Set of input actions / symbols
$\boldsymbol{A \subseteq Q}$	**Accepting state(s)**
$\delta: Q \times I \rightarrow Q$	Transition function

Describe the language recognized by these finite state automata.



All bitstrings that contain only 1s



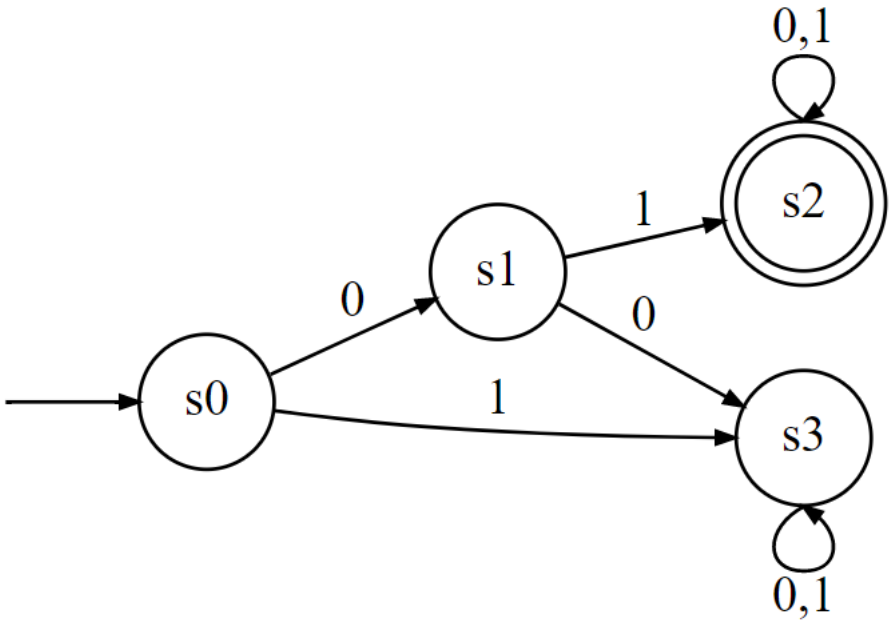
The bitstrings {1, 01}

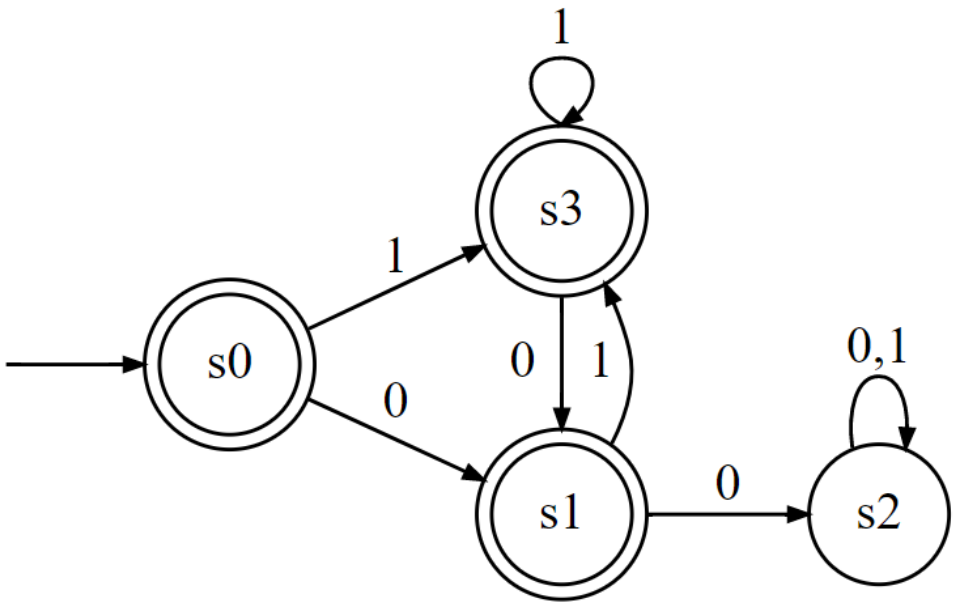
Practice - Finite State Automata

Go to the whiteboards and draw these two finite state automata:

1. The set of bitstrings that begin with 01

2. The set of bitstrings that do not contain two consecutive 0s





FSM Examples in Python

Examples of drawing FSMs using Python