Finite Automata
Part One
Computability Theory
What problems can we solve with a computer?
What problems can we solve with a computer?
Computers are Messy

http://www.intel.com/design/intarch/prodbref/272713.htm
Computers are Messy

microSD/SD Card interface with ATmega32  Ver_2.3

by CC Dharmani
www.dharmanitech.com

http://www.dharmanitech.com/
Computers are Messy

4, 8, 16 or 30 SMs
(32, 64, 128 or 240 SPs)

Secondary cache
Secondary cache
Secondary cache

Interconnection

SM
Cache
Multithreading

SP
SP
SP
SP
SP
SFU
SFU
DP
Shared memory

Double-precision SP

Main memory
Main memory
Main memory

Fig 2 Covering Everything from PCs to Supercomputers NVIDIA’s CUDA architecture boasts high scalability. The quantity of processor units (SM) can be varied as needed to flexibly provide performance from PC to supercomputer levels. Tesla 10, with 240 SPs, also has double-precision operation units (SM) added.

http://techon.nikkeibp.co.jp/article/HONSHI/20090119/164259/
Computers are Messy

Computers are Messy

That messiness makes it hard to rigorously say what we intuitively know to be true: that, on some fundamental level, different brands of computers or programming languages are more or less equivalent in what they are capable of doing.

C vs C++

vs Java

vs Python
We need a simpler way of discussing computing machines.
An **automaton** (plural: **automata**) is a mathematical model of a computing device.
Computers are Messy

http://www.intel.com/design/intarch/prodbref/272713.htm
Automata are Clean
Computers are Messy

by CC Dharmani
www.dharmanitech.com

microSD/SD Card interface with ATmega32  Ver_2.3

http://www.dharmanitech.com/
Automata are Clean

\[ q_0 \quad q_1 \quad q_2 \quad q_3 \]

\[ \begin{array}{c}
\text{start} \\
q_0 \\
q_3 \\
q_2 \\
q_1
\end{array} \]

\[ \begin{array}{cccc}
0 & 0 & 0 & \text{accept}
\end{array} \]
Computers are Messy

Fig 2  Covering Everything from PCs to Supercomputers  NVIDIA's CUDA architecture boasts high scalability. The quantity of processor units (SM) can be varied as needed to flexibly provide performance from PC to supercomputer levels. Tesla 10, with 240 SPs, also has double-precision operation units (SM) added.
Automata are Clean

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_1 \rightarrow q_2 \rightarrow q_0 \]

- Start state: \( q_0 \)
- Transitions: 0, 1
Computers are Messy

Automata are Clean

The diagram represents a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are labeled with symbols 0 and 1. The start state is $q_0$.
Why Build Models?

• **Mathematical simplicity.**
  - It is significantly easier to manipulate our abstract models of computers than it is to manipulate actual computers.

• **Intellectual robustness.**
  - If we pick our models correctly, we can make broad, sweeping claims about huge classes of real computers by arguing that they're just special cases of our more general models.
Why Build Models?

- The models of computation we will explore in this class correspond to different conceptions of what a computer could do.

- **Finite automata** (next two weeks) are an abstraction of computers with finite resource constraints.
  - Provide upper bounds for the computing machines that we can actually build.

- **Turing machines** (later) are an abstraction of computers with unbounded resources.
  - Provide upper bounds for what we could ever hope to accomplish.
What problems can we solve with a computer?
What problems can we solve with a computer?

What is a "problem?"
Problems with Problems

• Before we can talk about what problems we can solve, we need a formal definition of a “problem.”

• We want a definition that
  • corresponds to the problems we want to solve,
  • captures a large class of problems, and
  • is mathematically simple to reason about.

• No one definition has all three properties.
Formal Language Theory
Strings

• An **alphabet** is a finite, nonempty set of symbols called **characters**.
  
  • Typically, we use the symbol $\Sigma$ to refer to an alphabet.

• A **string over an alphabet $\Sigma$** is a finite sequence of characters drawn from $\Sigma$.

• Example: If $\Sigma = \{a, b\}$, here are some valid strings over $\Sigma$:
  
  a  aabaaabbabaaabaaabaaaabbb  abbababba

• The **empty string** has no characters and is denoted $\varepsilon$.

• Calling attention to an earlier point: since all strings are finite sequences of characters from $\Sigma$, you cannot have a string of infinite length.
Languages

• A **formal language** is a set of strings.

• We say that $L$ is a **language over $\Sigma$** if it is a set of strings over $\Sigma$.

• Example: The language of palindromes over $\Sigma = \{a, b, c\}$ is the set
  
  $\{\varepsilon, a, b, c, aa, bb, cc, aaa, aba, aca, bab, \ldots \}$

• The set of all strings composed from letters in $\Sigma$ is denoted $\Sigma^*$.

• Formally, we say that $L$ is a language over $\Sigma$ if $L \subseteq \Sigma^*$. 
How many of the following statements are true?

- **Alphabets** are sequences of characters.
- **Languages** are sets of strings.
- **Strings** are sets of characters.
- **Characters** are individual symbols.
- **Languages** are sequences of characters.

Answer at PollEv.com/cs103 or text CS103 to 22333 once to join, then a number.
To Recap

- **Languages** are sets of strings.
- **Strings** are sequences of characters.
- **Characters** are individual symbols.
- **Alphabets** are sets of characters.
The Model

• **Fundamental Question:** For what languages $L$ can you design an automaton that takes as input a string, then determines whether the string is in $L$?

• The answer depends on the choice of $L$, the choice of automaton, and the definition of “determines.”

• In answering this question, we’ll go through a whirlwind tour of models of computation and see how this seemingly abstract question has very real and powerful consequences.
To Summarize

• An **automaton** is an idealized mathematical computing machine.

• A **language** is a set of strings, a **string** is a (finite) sequence of characters, and a **character** is an element of an **alphabet**.

• **Goal:** Figure out in which cases we can build automata for particular languages.
What problems can we solve with a computer?
Finite Automata
A **finite automaton** is a simple type of mathematical machine for determining whether a string is contained within some language.
Each finite automaton consists of a set of *states* connected by *transitions*. 
A Simple Finite Automaton
A Simple Finite Automaton

Each circle represents a state of the automaton.
A Simple Finite Automaton
A Simple Finite Automaton

One special state is designated as the start state.
A Simple Finite Automaton

![Finite Automaton Diagram]

- States: $q_0$, $q_1$, $q_3$, $q_2$
- Transitions:
  - From $q_0$: 0 to $q_1$
  - From $q_1$: 0 to $q_0$, 0 to $q_2$
  - From $q_2$: 0 to $q_3$
  - From $q_3$: 1 to $q_0$, 1 to $q_2$

Start state: $q_0$
A Simple Finite Automaton

\begin{tikzpicture}
  \node[state, initial] (q0) {$q_0$};
  \node[state] (q1) [right of=q0] {$q_1$};
  \node[state] (q2) [below right of=q1] {$q_2$};
  \node[state] (q3) [below left of=q0] {$q_3$};

  \draw[->] (q0) -- node[above] {$0$} (q1);
  \draw[->] (q1) -- node[above] {$0$} (q2);
  \draw[->] (q2) -- node[above] {$0$} (q3);
  \draw[->] (q3) -- node[above] {$0$} (q0);
  \draw[->] (q0) -- node[below] {$1$} (q3);
  \draw[->] (q1) -- node[below] {$1$} (q2);
  \draw[->] (q2) -- node[below] {$1$} (q1);
  \draw[->] (q3) -- node[below] {$1$} (q0);

  \end{tikzpicture}
A Simple Finite Automaton

The automaton is run on an input string and answers "yes" or "no."
A Simple Finite Automaton

\[ \begin{array}{c}
\text{start} \\
 q_0 \\
 q_1 \\
 q_3 \\
 q_2 \\
\end{array} \]
A Simple Finite Automaton

\[
\begin{array}{cccc}
0 & 1 & 0 & 1 & 1 & 1 & 0 \\
q_0 & q_1 & q_2 & q_3 \\
\end{array}
\]
A Simple Finite Automaton

The automaton can be in one state at a time. It begins in the start state.
A Simple Finite Automaton

start

$q_0$ 0 0

$q_1$

$q_0$

$q_3$

$q_2$

0 1 1 0 1 1 0
A Simple Finite Automaton

start

$q_0$ → $q_1$

$q_0$ → $q_3$

$q_3$ → $q_2$

$q_2$ → $q_1$

$q_1$ → $q_0$

Input: 0 1 0 1 1 1 0
The automaton now begins processing characters in the order in which they appear.
A Simple Finite Automaton
A Simple Finite Automaton

\begin{tikzpicture}
\node[state,initial] (q0) at (0,0) {$q_0$};
\node[state] (q1) at (2,0) {$q_1$};
\node[state] (q2) at (0,-2) {$q_2$};
\node[state] (q3) at (2,-2) {$q_3$};
\draw[->,thick,red] (q0) -- node[above] {$0$} (q1);
\draw[->,thick] (q0) -- node[left] {$1$} (q3);
\draw[->,thick] (q3) -- node[above] {$0$} (q2);
\draw[->,thick] (q2) -- node[left] {$1$} (q1);
\end{tikzpicture}
A Simple Finite Automaton

Each arrow in this diagram represents a transition. The automaton always follows the transition corresponding to the current symbol being read.
A Simple Finite Automaton

start

$q_0$  0

$q_1$

$q_2$

$q_3$

0 1 0 1 1 1 0
A Simple Finite Automaton

\begin{center}
\begin{tikzpicture}[node distance = 2cm, inner sep = 1pt]
    \node (q0) [state] {$q_0$};
    \node (q1) [state, right of=q0, fill=yellow] {$q_1$};
    \node (q2) [state, below of=q1] {$q_2$};
    \node (q3) [state, below of=q0] {$q_3$};

    \draw [->] (q0) -- node[above] {0} (q1);
    \draw [->] (q0) -- node[above] {1} (q3);
    \draw [->] (q1) -- node[above] {0} (q2);
    \draw [->] (q1) -- node[below] {1} (q3);
    \draw [->] (q2) -- node[below] {0} (q0);
    \draw [->] (q3) -- node[below] {1} (q1);

    \node at (-2, -2.5) {start};
\end{tikzpicture}
\end{center}

\text{0 1 0 1 1 1 0}
A Simple Finite Automaton

The diagram shows a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are as follows:

- From $q_0$ to $q_1$ on input 0.
- From $q_1$ to $q_0$ on input 0.
- From $q_1$ to $q_2$ on input 0.
- From $q_1$ to $q_1$ on input 1.
- From $q_2$ to $q_3$ on input 0.
- From $q_3$ to $q_2$ on input 0.
- From $q_3$ to $q_0$ on input 0.
- From $q_3$ to $q_2$ on input 0.

The input string is 0 1 0 1 1 1 0.
A Simple Finite Automaton

Start state: $q_0$

Transitions:
- $q_0 \xrightarrow{0} q_1$
- $q_1 \xrightarrow{1} q_2$
- $q_2 \xrightarrow{0} q_3$
- $q_3 \xrightarrow{1} q_0$

Accepting state: $q_2$

Input string: 0 1 0 1 1 1 0
A Simple Finite Automaton

After transitioning, the automaton considers the next symbol in the input.

0 1 0 1 1 1 0
A Simple Finite Automaton

[Diagram of a finite automaton with states q₀, q₁, q₂, q₃ and transitions labeled with 0s and 1s, starting from q₀ and reading input 0101110]
A Simple Finite Automaton
A Simple Finite Automaton

The diagram shows a finite automaton with four states: $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are as follows:

- From $q_0$ to $q_1$ on input 0.
- From $q_0$ to $q_3$ on input 1.
- From $q_1$ to $q_0$ on input 0.
- From $q_1$ to $q_2$ on input 1.
- From $q_2$ to $q_3$ on input 0.
- From $q_3$ to $q_0$ on input 0.

The automaton starts at state $q_0$. The input sequence is 0 1 0 1 1 1 0.
A Simple Finite Automaton

$q_0$ 0 1

$q_1$ 0 0

$q_2$ 1 1 1

$q_3$ 1 1

start

0 1 0 1 1 1 0
A Simple Finite Automaton

$$\begin{array}{c}
q_0 \\
q_1 \\
q_3 \\
q_2
\end{array}$$

$$\begin{array}{cccccccc}
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0
\end{array}$$

Start transition

0 1 0 1 1 1 0

Start state
A Simple Finite Automaton

Graphical representation of an finite automaton with transitions labeled 0 and 1.
A Simple Finite Automaton

\begin{itemize}
\item Start state: \( q_0 \)
\item Transitions:
  \begin{align*}
  \text{On input } 0: & \quad q_0 \rightarrow q_1 \\
  \text{On input } 1: & \quad q_0 \rightarrow q_3 \\
  \text{On input } 0: & \quad q_1 \rightarrow q_2 \\
  \text{On input } 1: & \quad q_3 \rightarrow q_2 \\
  \text{On input } 0: & \quad q_2 \rightarrow q_1 \\
  \end{align*}
\end{itemize}

Input string: 0 1 0 1 1 1 0
A Simple Finite Automaton

\[ \begin{array}{c}
\text{start} \\
q_0 \rightarrow \q_1 \\
q_3 \rightarrow \q_2 \\
q_1 \rightarrow \q_3 \\
q_2 \rightarrow \q_0 \\
\end{array} \]
A Simple Finite Automaton
A Simple Finite Automaton

![Finite Automaton Diagram]

- States: $q_0$, $q_1$, $q_2$, $q_3$
- Transitions:
  - $q_0 \xrightarrow{0} q_1$
  - $q_0 \xrightarrow{1} q_3$
  - $q_1 \xrightarrow{0} q_2$
  - $q_2 \xrightarrow{1} q_3$
  - $q_3 \xrightarrow{0} q_2$

Input: 0101110
A Simple Finite Automaton

![Finite Automaton Diagram]

- Start state: $q_0$
- States: $q_0, q_1, q_2, q_3$
- Transitions:
  - From $q_0$: 0 to $q_1$, 1 to $q_3$
  - From $q_1$: 0 to $q_2$, 1 to $q_0$
  - From $q_2$: 1 to $q_1$, 0 to $q_3$
  - From $q_3$: 0 to $q_2$, 1 to $q_0$

Input: 0101110
A Simple Finite Automaton

The diagram shows a finite automaton with states labeled $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are labeled with input symbols 0 and 1. The automaton starts in state $q_0$ and moves through states $q_1$, $q_2$, and $q_3$.
A Simple Finite Automaton

start

$q_0$ 0

$q_1$ 0

$q_3$ 1 1

$q_2$ 1 1

0 1 0 1 1 1 0
A Simple Finite Automaton

\begin{itemize}
  \item $q_0$ is the start state.
  \item Transitions:
    \begin{itemize}
    \item $q_0 \xrightarrow{0} q_1$
    \item $q_0 \xrightarrow{1} q_3$
    \item $q_1 \xrightarrow{0} q_0$
    \item $q_1 \xrightarrow{1} q_2$
    \item $q_3 \xrightarrow{1} q_2$
    \item $q_3 \xrightarrow{0} q_0$
    \end{itemize}
\end{itemize}

Input sequence: 0 1 0 1 1 1 0

The automaton ends in state $q_0$.
A Simple Finite Automaton

\[ q_0 \rightarrow q_1 \]

\[ q_0 \rightarrow q_2 \]

\[ q_2 \rightarrow q_3 \]

\[ q_3 \rightarrow q_0 \]

\[ q_3 \rightarrow q_2 \]

\[ q_1 \rightarrow q_3 \]

\[ q_1 \rightarrow q_2 \]

\[ q_2 \rightarrow q_1 \]

\[ q_2 \rightarrow q_0 \]

Input: 0 1 0 1 1 1 0
A Simple Finite Automaton

The diagram shows a finite automaton with states $q_0$, $q_1$, $q_3$, and $q_2$. The transitions are labeled with inputs 0 and 1, and the arrows indicate the movement between states. The start state is $q_0$.
A Simple Finite Automaton

![Finite Automaton Diagram]
A Simple Finite Automaton

\[
\begin{array}{c}
\text{start} \\
q_0 \rightarrow 0 \rightarrow q_1 \\
1 \rightarrow q_3 \rightarrow 0 \\
0 \rightarrow q_2 \rightarrow 1 \rightarrow 1 \\
0 \rightarrow q_3 \\
0 \\
\end{array}
\]
A Simple Finite Automaton

\begin{figure}
\centering
\includegraphics[width=\textwidth]{automaton.png}
\end{figure}
A Simple Finite Automaton

\[ q_0 \xrightarrow{0} q_1 \xrightarrow{0} \]

\[ q_3 \xrightarrow{1} q_2 \xrightarrow{1} \]

\[ q_0 \xrightarrow{1} q_3 \xrightarrow{1} q_2 \xrightarrow{0} q_0 \]

Input: 0 1 0 1 1 1 0
A Simple Finite Automaton

The diagram represents a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions include:
- From $q_0$ to $q_1$ on input 0.
- From $q_0$ to $q_3$ on input 1.
- From $q_3$ to $q_0$ on input 0.
- From $q_1$ to $q_2$ on input 0.
- From $q_1$ to $q_1$ on input 1.
- From $q_2$ to $q_2$ on input 1.

The start state is $q_0$.
A Simple Finite Automaton

Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."
A Simple Finite Automaton

Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."

The double circle indicates that this state is an accepting state, so the automaton outputs "yes."

0 1 0 1 1 1 0
A Simple Finite Automaton

Now that the automaton has looked at all this input, it can decide whether to say “yes” or “no.”

The double circle indicates that this state is an accepting state, so the automaton outputs “yes.”
A Simple Finite Automaton

Now that the automaton has looked at all this input, it can decide whether to say "yes" or "no."

The double circle indicates that this state is an accepting state, so the automaton outputs "yes."

0 1 0 1 1 1 0
A Simple Finite Automaton

Diagram:

- Start state: $q_0$
- States: $q_0, q_1, q_2, q_3$
- Transitions:
  - $q_0 \xrightarrow{0} q_1$
  - $q_0 \xrightarrow{1} q_3$
  - $q_1 \xrightarrow{0} q_0$
  - $q_1 \xrightarrow{1} q_2$
  - $q_2 \xrightarrow{1} q_1$
  - $q_2 \xrightarrow{1} q_3$
  - $q_3 \xrightarrow{1} q_1$
  - $q_3 \xrightarrow{1} q_2$
A Simple Finite Automaton

![Finite Automaton Diagram]

- **Start State**: $q_0$
- **States**: $q_0, q_1, q_2, q_3$
- **Transitions**:
  - $q_0$ on 0 to $q_1$
  - $q_0$ on 1 to $q_3$
  - $q_3$ on 1 to $q_1$
  - $q_3$ on 1 to $q_2$
  - $q_2$ on 0 to $q_3$
  - $q_1$ on 0 to $q_2$

- **Input Strings**:
  - $1010000$
A Simple Finite Automaton
A Simple Finite Automaton

\begin{center}
\begin{tikzpicture}
    \node[state,fill=yellow] (q0) at (0,0) {$q_0$};
    \node[state] (q1) at (2,0) {$q_1$};
    \node[state] (q2) at (2,-2) {$q_2$};
    \node[state] (q3) at (0,-2) {$q_3$};

    \draw[->] (q0) edge node[above] {0} (q1);
    \draw[->] (q0) edge node[below] {1} (q3);
    \draw[->] (q1) edge node[below] {0} (q2);
    \draw[->] (q1) edge node[above] {0} (q3);
    \draw[->] (q2) edge node[above] {1} (q0);
    \draw[->] (q2) edge node[below] {1} (q3);
    \draw[->] (q3) edge node[below] {0} (q0);
    \draw[->] (q3) edge node[above] {1} (q1);

    \node at (-1,-1) {start};
\end{tikzpicture}
\end{center}

1 0 1 0 0 0 0
A Simple Finite Automaton
A Simple Finite Automaton

\[ \begin{array}{c}
\text{start} \\
q_0 \quad 0 \\
q_1 \quad 0 \\
q_2 \quad 0 \\
q_3 \quad 0 \\
\end{array} \]
A Simple Finite Automaton
A Simple Finite Automaton

Diagram:

- **Start State**: $q_0$
- States: $q_0$, $q_1$, $q_2$, $q_3$
- Transitions:
  - $q_0$ to $q_1$ on input 0
  - $q_0$ to $q_0$ on input 1
  - $q_1$ to $q_0$ on input 0
  - $q_1$ to $q_1$ on input 1
  - $q_2$ to $q_2$ on input 1
  - $q_2$ to $q_3$ on input 0
  - $q_3$ to $q_2$ on input 0
  - $q_3$ to $q_3$ on input 1

Input String: 1 0 1 0 0 0 0
A Simple Finite Automaton

- States: $q_0, q_1, q_2, q_3$
- Transitions:
  - $q_0 \to q_1$ on input 0
  - $q_1 \to q_0$ on input 1
  - $q_0 \to q_2$ on input 1
  - $q_2 \to q_0$ on input 0
  - $q_2 \to q_3$ on input 0
  - $q_3 \to q_2$ on input 1

- Start state: $q_0$

- Sample input: 1010000
A Simple Finite Automaton

start

$q_0$ — $q_1$

$q_3$ — $q_2$

1 1 1 1 1

1 0 1 0 0 0
A Simple Finite Automaton

\[ q_0 \rightarrow q_1 \rightarrow q_3 \rightarrow q_2 \]

Input: 1010000
A Simple Finite Automaton
A Simple Finite Automaton

\[ \begin{align*}
q_0 & \xrightarrow{0} q_1, \\
q_1 & \xrightarrow{0} q_0, \\
q_0 & \xrightarrow{1} q_3, \\
q_3 & \xrightarrow{1} q_2, \\
q_2 & \xrightarrow{1} q_1, \\
q_1 & \xrightarrow{1} q_3,
\end{align*} \]
A Simple Finite Automaton

![Diagram of a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are labeled with inputs 0 and 1. The start state is $q_0$. The automaton transitions from $q_0$ to $q_1$ on input 0, from $q_1$ to $q_2$ on input 1, from $q_2$ to $q_3$ on input 1, and from $q_3$ to $q_0$ on input 1. The input sequence is 1 0 1 0 0 0 0.]
A Simple Finite Automaton
A Simple Finite Automaton

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_0 \]

Input: 1 0 1 0 0 0 0
A Simple Finite Automaton

![Finite Automaton Diagram]

The diagram represents a finite automaton with states $q_0$, $q_1$, $q_2$, and $q_3$. The transitions are labeled with input symbols 0 and 1. The start state is $q_0$, and the accepted states are $q_1$ and $q_3$. The input sequence is 10110000.
A Simple Finite Automaton

start

$q_0$ 0 0 0

$q_1$

$q_2$

$q_3$

1 1 1 1 1 1

1 0 1 0 0 0 0
A Simple Finite Automaton

\[ \begin{array}{c}
\text{start} \\
\bullet q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \\
\xrightarrow{1} q_3 \quad \xrightarrow{1} q_2 \\
q_3 \xrightarrow{0} q_2 \xrightarrow{0} q_1 \xrightarrow{0} q_0
\end{array} \]
A Simple Finite Automaton

\[
\begin{array}{c}
\text{start} \\
q_0 \\
q_3 \\
q_2 \\
q_1 \\
\end{array}
\]

\[
\begin{array}{c}
0 \\
0 \\
0 \\
0 \\
0 \\
\end{array}
\]

\[
\begin{array}{c}
1 \\
1 \\
1 \\
1 \\
1 \\
\end{array}
\]

\[
\begin{array}{c}
1 \\
0 \\
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{array}
\]

\[
\begin{array}{c}
1 \\
0 \\
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{array}
\]
A Simple Finite Automaton

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_0 \]

Input String: 1 0 1 0 0 0
A Simple Finite Automaton
A Simple Finite Automaton

```
start
q_0
0
1 1
q_3
1
1
q_2
0
0
q_1
0
0
```

1 0 1 0 0 0
A Simple Finite Automaton

![Finite Automaton Diagram]

- Start state: $q_0$
- States: $q_0$, $q_1$, $q_2$, $q_3$
- Transitions:
  - $q_0$ to $q_0$: 0
  - $q_0$ to $q_1$: 1
  - $q_1$ to $q_2$: 1
  - $q_2$ to $q_0$: 0
  - $q_2$ to $q_3$: 0
  - $q_3$ to $q_1$: 0
- Input string: 1 0 1 0 0 0
A Simple Finite Automaton
A Simple Finite Automaton

start

$q_0$ 0

$q_3$

$q_1$

$q_2$

1 1 1 1 1 1

1 0 1 0 0 0
A Simple Finite Automaton

start

$q_0$ 0 $q_1$

$1 1 1 1$

$q_3$ 0 $q_2$

$1 1 0 1 0 0 0$
A Simple Finite Automaton

\begin{center}
\begin{tikzpicture}[node distance=2cm, thick, main node/.style={circle,fill=yellow!30,draw}]
    \node[main node] (q0) {$q_0$};
    \node[main node] (q1) [right of=q0] {$q_1$};
    \node[main node] (q2) [below right of=q1] {$q_2$};
    \node[main node] (q3) [below left of=q0] {$q_3$};

    \path[->]
    (q0) edge node[above] {0} (q1)
    (q1) edge node[below] {0} (q0)
    (q0) edge [loop above] node {0} (q0)
    (q1) edge [loop above] node {0} (q1)
    (q2) edge node[above] {0} (q1)
    (q1) edge node[below] {0} (q2)
    (q2) edge [loop below] node {0} (q2)
    (q3) edge node[above] {1} (q0)
    (q0) edge node[below] {1} (q3)
    (q3) edge [loop below] node {1} (q3)
    (q2) edge node[above] {1} (q3)
    (q3) edge node[below] {1} (q2)
    (q3) edge [loop below] node {1} (q3);
    \end{tikzpicture}
\end{center}
This state is not an accepting state (it's a rejecting state), so the automaton says "no."
A Simple Finite Automaton

This state is not an accepting state (it’s a rejecting state), so the automaton says “no.”
A Simple Finite Automaton

This state is not an accepting state (it's a rejecting state), so the automaton says "no."

1 0 1 0 0 0
A Simple Finite Automaton

\[
\begin{array}{c}
\text{start} \\
q_0 \xrightarrow{0} q_1 \\
q_3 \xrightarrow{1} q_1 \\
q_3 \xrightarrow{1} q_2 \\
q_2 \xrightarrow{1} q_2 \\
q_1 \xrightarrow{0} q_2 \\
q_2 \xrightarrow{0} q_2 \\
q_3 \xrightarrow{0} q_3 \\
q_0 \xrightarrow{1} q_0 \\
q_1 \xrightarrow{0} q_1 \\
q_2 \xrightarrow{1} q_2 \\
q_3 \xrightarrow{1} q_3 \\
\end{array}
\]
A Simple Finite Automaton

Try it yourself! Does this automaton accept (vote A) or reject (vote R)?

Answer at PollEv.com/cs103 or text CS103 to 22333 once to join, then A or R.
The Story So Far

• A **finite automaton** is a collection of **states** joined by **transitions**.

• Some state is designated as the **start state**.

• Some states are designated as **accepting states**.

• The automaton processes a string by beginning in the start state and following the indicated transitions.

• If the automaton ends in an accepting state, it **accepts** the input.

• Otherwise, the automaton **rejects** the input.
Time-Out For Announcements!
Girl Code @Stanford

- This summer, I’ll be running our sixth iteration of Girl Code @Stanford from July 9th – July 20th.
- We invite high-school girls (primarily from low- to middle-income schools in majority-minority areas) to come to campus for two weeks to learn CS, meet researchers, and talk to folks from industry.
- We’re looking for Stanford students to serve as “Workshop Assistants” during the program. We pay competitively (roughly $3,000 over two weeks).
- Interested? Learn more and apply using this link: https://goo.gl/forms/Y76akbVWUYV0NPpR2

All current Stanford students are invited to apply. Feel free to forward this link around!
STANFORD OUT IN STEM PRESENTS

LGBTQ (A+) Study Night

MON. FEB 12 • 5:30-7:30 PM
QSPOT
Back to CS103!
Just Passing Through

![Diagram of a finite automaton with states $q_0$, $q_1$, $q_2$, $q_3$, and $q_4$. The transitions are labeled with 1's and 0's. The start state is $q_0$. ]
Just Passing Through
Just Passing Through

![Diagram of a state transition graph with states $q_0$, $q_1$, $q_2$, $q_3$, and $q_4$ connected by transitions labeled with 0's and 1's, and an input sequence of 1101 on the right side of the image.](image-url)
Just Passing Through

Start

$q_0$

$1$

$0$

$q_1$

$q_2$

$q_3$

$q_4$

Transition:

- $q_0$ to $q_1$: $1$
- $q_0$ to $q_2$: $0$
- $q_1$ to $q_0$: $1$
- $q_1$ to $q_3$: $1$
- $q_2$ to $q_1$: $1$
- $q_2$ to $q_4$: $0$
- $q_3$ to $q_2$: $0$
- $q_3$ to $q_4$: $1$
- $q_4$ to $q_3$: $1$
- $q_4$ to $q_2$: $0$

Input String: 1101
Just Passing Through

Transition diagram:
- Start state: $q_0$
- States: $q_0, q_1, q_2, q_3, q_4$
- Transitions:
  - $q_0$ to $q_1$: 1
  - $q_0$ to $q_2$: 0
  - $q_1$ to $q_3$: 1
  - $q_2$ to $q_3$: 1
  - $q_3$ to $q_4$: 0
  - $q_3$ to $q_4$: 1
  - $q_4$ to $q_1$: 0
  - $q_4$ to $q_2$: 0

Input sequence: 1 1 0 1
Just Passing Through

\[q_\text{start} \rightarrow q_0\]

\[q_0 \rightarrow q_1, q_2\]

\[q_1 \rightarrow q_3, q_4\]

\[q_2 \rightarrow q_3\]

Input: 1101
Just Passing Through

The diagram shows a finite automaton with states $q_0, q_1, q_2, q_3, q_4$. The transitions are labeled with 0s and 1s. The sequence 1101 is shown along the path that the automaton takes from the start state $q_0$.
Just Passing Through

\[
\begin{array}{c}
\text{start} \\
\downarrow \\
q_0 \\
\downarrow 1 \\
q_1 \\
\downarrow 1 \\
q_3 \\
\downarrow 1 \\
\text{1101} \\
\end{array}
\]
Just Passing Through

\[
\begin{array}{c}
\text{start} \\
\rightarrow \\
q_0 \quad 1 \\
\quad 0 \\
q_1 \quad 0 \\
q_2 \quad 1 \\
q_3 \quad 1 \\
q_4 \quad 0 \\
\end{array}
\]
Just Passing Through

The given automaton transitions are as follows:
- From $q_0$ to $q_1$ on input 1.
- From $q_1$ to $q_2$ on input 0.
- From $q_2$ to $q_1$ on input 1.
- From $q_3$ to $q_4$ on input 1.
- From $q_4$ to $q_3$ on input 0.

The sequence 1 1 0 1 makes the automaton transition through states as follows:

1. Start at $q_0$.
2. Move to $q_1$ on input 1.
3. Move to $q_2$ on input 0.
4. Move back to $q_1$ on input 1.
5. Move to $q_4$ on input 0.
6. Move to $q_3$ on input 1.

The final state is $q_3$. The sequence ends at $q_3$. 
Just Passing Through

\[
\begin{array}{c}
\text{start} \\
\rightarrow \\
q_0 \\
\rightarrow 1 \\
\rightarrow 0 \\
q_1 \\
\rightarrow 1 \\
\rightarrow 0 \\
q_2 \\
\rightarrow 1 \\
\rightarrow 0 \\
q_3 \\
\rightarrow 1 \\
\rightarrow 0 \\
q_4 \\
\end{array}
\]

1 1 0 1
Just Passing Through

\[ \text{1101} \]
Just Passing Through

\[\text{start} \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4 \]

\[
\begin{array}{c}
q_0 \\
1 \quad 0 \\
q_1 \\
1 \quad 0 \quad 1 \quad 1 \\
q_2 \\
1 \quad 1 \quad 1 \\
q_3 \\
1 \quad 0 \quad 1 \\
q_4 \\
1 \quad 0 \\
\end{array}
\]
Just Passing Through

```
q0

1
0

q1

0
1

q2

1
0

q3

0
1

q4

1
0

start

1 1 0 1
```
A finite automaton does not accept as soon as it enters an accepting state.

A finite automaton accepts if it ends in an accepting state.
What Does This Accept?

- Start state: $q_0$
- Transitions:
  - From $q_0$: 
    - On input 1, goes to $q_1$.
    - On input 0, goes to $q_2$.
  - From $q_1$: 
    - On input 0, goes back to $q_0$.
    - On input 1, goes to $q_2$.
  - From $q_2$: 
    - On input 1, goes to $q_3$.
    - On input 0, goes to $q_4$.
  - From $q_3$: 
    - On input 1, goes to $q_4$.
    - On input 0, goes to $q_1$.
  - From $q_4$: 
    - On input 0, goes to $q_3$.
    - On input 1, goes to $q_2$. 

What Does This Accept?

Graph:
- Start state: $q_0$
- States: $q_0, q_1, q_2, q_3, q_4$
- Transitions:
  - $q_0$: 1 to $q_1$, 0 to $q_2$
  - $q_1$: 1 to $q_0$, 0 to $q_2$
  - $q_2$: 1 to $q_1$, 0 to $q_3$
  - $q_3$: 1 to $q_2$, 0 to $q_4$
  - $q_4$: 1 to $q_3$, 0 to $q_4$
What Does This Accept?
What Does This Accept?
What Does This Accept?

The diagram represents a finite automaton with states $q_0, q_1, q_2, q_3,$ and $q_4$. The arrows indicate the transitions based on input symbols:

- From $q_0$ to $q_1$: Input 1
- From $q_0$ to $q_2$: Input 0
- From $q_1$ to $q_0$: Input 1
- From $q_1$ to $q_3$: Input 1
- From $q_2$ to $q_4$: Input 0
- From $q_2$ to $q_1$: Input 1
- From $q_3$ to $q_0$: Input 1
- From $q_3$ to $q_2$: Input 0
- From $q_4$ to $q_3$: Input 0
- From $q_4$ to $q_2$: Input 0
What Does This Accept?
What Does This Accept?
No matter where we start in the automaton, after seeing two 1's, we end up in accepting state $q_3$. 
What Does This Accept?

![Diagram of a finite automaton with states and transitions.]

- **States (Q):** $q_0$, $q_1$, $q_2$, $q_3$, $q_4$
- **Initial State (Start):** $q_0$
- **Transitions:**
  - $q_0$ to $q_1$ on 1
  - $q_0$ to $q_2$ on 0
  - $q_1$ to $q_0$ on 0
  - $q_1$ to $q_3$ on 1
  - $q_2$ to $q_0$ on 1
  - $q_2$ to $q_4$ on 0
  - $q_3$ to $q_0$ on 0
  - $q_3$ to $q_1$ on 1
  - $q_3$ to $q_2$ on 1
  - $q_4$ to $q_0$ on 1
  - $q_4$ to $q_2$ on 0
What Does This Accept?
What Does This Accept?

The diagram shows a finite automaton with states $q_0, q_1, q_2, q_3,$ and $q_4$. The arrows indicate transitions for the input symbols 0 and 1. The start state is $q_0$. The automaton accepts certain strings based on the transitions through these states.
What Does This Accept?
What Does This Accept?

![Automaton Diagram]

The automaton starts in state $q_0$. It transitions as follows:
- From $q_0$ on input 1 to $q_1$.
- From $q_0$ on input 0 to $q_2$.
- From $q_1$ on input 0 to $q_3$.
- From $q_1$ on input 1 to $q_2$.
- From $q_2$ on input 1 to $q_4$.
- From $q_2$ on input 0 to $q_3$.
- From $q_3$ on input 1 to $q_4$.
- From $q_3$ on input 0 to $q_4$.
- From $q_4$ on any input to itself.

The automaton accepts strings that end in 0.
What Does This Accept?
No matter where we start in the automaton, after seeing two 0's, we end up in accepting state $q_5$. 

**Diagam:**

- **Start State:** $q_0$
- **States:** $q_1$, $q_2$, $q_3$, $q_4$, $q_5$
- **Transitions:**
  - $q_0$ to $q_1$: 1
  - $q_0$ to $q_2$: 0
  - $q_1$ to $q_0$: 1
  - $q_1$ to $q_3$: 1
  - $q_1$ to $q_2$: 0
  - $q_2$ to $q_0$: 1
  - $q_2$ to $q_1$: 0
  - $q_3$ to $q_1$: 1
  - $q_3$ to $q_3$: 0
  - $q_4$ to $q_4$: 0
  - $q_4$ to $q_2$: 0

**Note:** $q_5$ is the accepting state.
What Does This Accept?
This automaton accepts a string in \( \{0, 1\}^* \) iff the string ends in \( 00 \) or \( 11 \).
The *language of an automaton* is the set of strings that it accepts.

If $D$ is an automaton that processes characters from the alphabet $\Sigma$, then $\mathcal{L}(D)$ is formally defined as

$$\mathcal{L}(D) = \{ w \in \Sigma^* | D \text{ accepts } w \}$$
How many of the following statements are true?

- A *language* of an automaton can have an infinitely long string (or many of them) in it.
- A *language* of an automaton can contain infinitely many strings.
- A *language* of an automaton can contain no strings.

Answer at PollEv.com/cs103 or text CS103 to 22333 once to join, then a number.
A Small Problem

Start: $q_0$

$0 \rightarrow q_0 \rightarrow q_2$

$1 \rightarrow q_1 \rightarrow q_0$

$q_2 \rightarrow q_1$
A Small Problem

\begin{figure}
\begin{center}
\begin{tikzpicture}
\node[state, initial] (q0) at (0,0) {$q_0$};
\node[state, below left of=q0] (q2) {$q_2$};
\node[state, below right of=q0] (q1) {$q_1$};
\path[->]
(q0) edge [loop above] node {0}()
(q0) edge node {0} (q1)
(q0) edge node {1} (q2)
(q2) edge node {1} (q1);
\end{tikzpicture}
\end{center}
\end{figure}
A Small Problem
A Small Problem

![Diagram showing a finite automaton with states $q_0$, $q_1$, and $q_2$, and transitions labeled with inputs 0 and 1. The diagram starts at $q_0$ and moves to $q_1$ with input 0, and to $q_2$ with input 1. There is also a sequence of inputs 0110 indicated.]
A Small Problem

\[ \begin{array}{c}
q_0 \\
\text{start} \\
q_2 \\
\downarrow 1 \\
q_1 \\
\end{array} \]

\[ \begin{array}{c}
0 \\
0 \\
1 \\
0 \\
0 \\
\end{array} \]
A Small Problem
A Small Problem

The diagram represents a finite automaton with states $q_0$, $q_1$, $q_2$, and transitions labeled with 0 and 1. The start state is $q_0$, and the string 01110 is shown with an arrow pointing to the right, indicating the path through the states.
A Small Problem

\[ \begin{align*}
q_0 &\quad 0 \\
q_1 &\quad 0 \\
q_2 &\quad 1
\end{align*} \]
A Small Problem

\[
\begin{align*}
&\text{start} \\
&\downarrow \\
&q_0 \quad 0 \\
&q_2 \quad 1 \\
&\downarrow \\
&q_1 \quad 0 \\
\end{align*}
\]
Another Small Problem

\[ q_0 \xrightarrow{0, 1} q_1 \]

\[ q_1 \xrightarrow{0, 1} q_2 \]

\[ q_2 \xrightarrow{0} q_1 \]
Another Small Problem

\begin{center}
\begin{tikzpicture}
    \node[state, initial] (q0) {$q_0$};
    \node[state, right of=q0] (q1) {$q_1$};
    \node[state, below of=q1] (q2) {$q_2$};
    \path[->]
    (q0) edge node {$0, 1$} (q1)
    (q1) edge node {$0, 1$} (q2)
    (q2) edge node {$0$} (q1)
    (q0) edge[bend left] node {$0, 1$} (q2);
\end{tikzpicture}
\end{center}
Another Small Problem
Another Small Problem

\[ q_0 \rightarrow q_1 \rightarrow q_2 \]

0, 1

0

0, 1

0, 1

\[ 0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \]
Another Small Problem

- Start: $q_0$
  - Transition: 0, 1
- $q_1$
  - Transition: 0, 1
- $q_2$
  - Transition: 0

Input sequence: 0 0 0
Another Small Problem
Another Small Problem

Diagram:
- Start state: $q_0$
- States: $q_0$, $q_1$, $q_2$
- Transitions:
  - $q_0$ to $q_1$ on input 0, 1
  - $q_1$ to $q_2$ on input 0
  - $q_2$ to $q_1$ on input 0, 1

Input string: 0 0 0
Another Small Problem

start $\rightarrow q_0 \xrightarrow{0, 1} q_1 \xrightarrow{0, 1} q_2 \xrightarrow{0} 0000$

I HAVE NO IDEA WHAT I'M DOING
The Need for Formalism

• In order to reason about the limits of what finite automata can and cannot do, we need to formally specify their behavior in all cases.

• All of the following need to be defined or disallowed:
  • What happens if there is no transition out of a state on some input?
  • What happens if there are multiple transitions out of a state on some input?
DFAs

- A **DFA** is a
  - **D**eterministic
  - **F**inite
  - **A**utomaton
- DFAs are the simplest type of automaton that we will see in this course.
DFAs

- A DFA is defined relative to some alphabet $\Sigma$.
- For each state in the DFA, there must be exactly one transition defined for each symbol in $\Sigma$.
  - This is the “deterministic” part of DFA.
- There is a unique start state.
- There are zero or more accepting states.
How many of these are DFAs over \( \{0, 1\} \)?
Is this a DFA?
Is this a DFA?
Is this a DFA?
Designing DFAs

- At each point in its execution, the DFA can only remember what state it is in.

- **DFA Design Tip:** Build each state to correspond to some piece of information you need to remember.
  - Each state acts as a “memento” of what you're supposed to do next.
  - Only finitely many different states means only finitely many different things the machine can remember.
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* | \text{the number of b's in } w \text{ is congruent to two modulo three}\} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* | \text{the number of b's in } w \text{ is congruent to two modulo three} \} \]
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* | \text{the number of b's in } w \text{ is congruent to two modulo three} \}$
Recognizing Languages with DFAs

$L = \{ \ w \in \{a, b\}^*| \text{the number of b's in } w \text{ is congruent to two modulo three} \}$
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* | \text{the number of b's in } w \text{ is congruent to two modulo three} \}$
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* | \text{the number of b's in } w \text{ is congruent to two modulo three } \}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* | \text{the number of } b's \text{ in } w \text{ is congruent to two modulo three} \} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid \text{the number of } b\text{'s in } w \text{ is congruent to two modulo three} \} \]
Recognizing Languages with DFAs

$L = \{ \ w \in \{a, b\}^* | \text{the number of } b's \text{ in } w \text{ is congruent to two modulo three} \ \}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid \text{the number of } b\text{'s in } w \text{ is congruent to two modulo three} \} \]

Each state remembers the remainder of the number of bs seen so far modulo three.
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

$L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \}$
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

\[ L = \{ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \} \]
Recognizing Languages with DFAs

$L = \{ \ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring } \}$
More Elaborate DFAs

\[ L = \{ \ w \in \{ a, *, /\}^* | \ w \text{ represents a C-style comment} \ \} \]

Let’s have the \( a \) symbol be a placeholder for “some character that isn’t a star or slash.”

Try designing a DFA for comments! Here’s some test cases to help you check your work:

**Accepted:**

\[
/**a*/
/***/
/****/
/*aaa*/
/*a*/
/*a*/
\]

**Rejected:**

\[
/***/a/**aa*/
aaa/***/aa
/***/
/***/a/
//aaaa
\]
More Elaborate DFAs

\[ L = \{ w \in \{a, *, /\}^* \mid w \text{ represents a C-style comment} \} \]