CSE 311: Foundations of Computing

Lecture 22: Finite State Machines

---

**Comic**

**Panel 1:**
How do bank machines work?

**Panel 2:**
Well, let's say you want 25 dollars. You punch in the amount...

**Panel 3:**
...and behind the machine, there's a guy with a printing press who makes the money and sticks it out this slot.

**Panel 4:**
Sort of like the guy who lives up in our garage and opens the door?

**Panel 5:**
Exactly.
Last class: Strings this machine says are OK?

The set of all binary strings that end in 0
Finite State Machines

- **States**
- **Transitions on input symbols**
- **Start state and final states**
- The “language recognized” by the machine is the set of strings that reach a final state from the start

<table>
<thead>
<tr>
<th>Old State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$s_0$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$s_0$</td>
<td>$s_2$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$s_0$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$s_3$</td>
<td>$s_3$</td>
</tr>
</tbody>
</table>
Finite State Machines

• Each machine designed for strings over some fixed alphabet $\Sigma$.

• Must have a transition defined from each state for every symbol in $\Sigma$.

<table>
<thead>
<tr>
<th>Old State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$s_0$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$s_0$</td>
<td>$s_2$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$s_0$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$s_3$</td>
<td>$s_3$</td>
</tr>
</tbody>
</table>
What language does this machine recognize?

<table>
<thead>
<tr>
<th>Old State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$s_0$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$s_0$</td>
<td>$s_2$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$s_0$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$s_3$</td>
<td>$s_3$</td>
</tr>
</tbody>
</table>
What language does this machine recognize?

The set of all binary strings that contain 111 or don’t end in 1

<table>
<thead>
<tr>
<th>Old State</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₀</td>
<td>s₀</td>
<td>s₁</td>
</tr>
<tr>
<td>s₁</td>
<td>s₀</td>
<td>s₂</td>
</tr>
<tr>
<td>s₂</td>
<td>s₀</td>
<td>s₃</td>
</tr>
<tr>
<td>s₃</td>
<td>s₃</td>
<td>s₃</td>
</tr>
</tbody>
</table>
Strings over \{0, 1, 2\}

\(M_1\): Strings with an even number of 2’s
Strings over \{0, 1, 2\}

$M_1$: Strings with an even number of 2’s
State Machine Design Recipe

Given a language, how do you design a state machine for it?

Create states to remember enough
(about the portion of the input string that it has already seen)
to correctly answer “accept/reject” on the whole string after seeing the rest.

Add labeled edges to show how the memory (state) should be updated for each new symbol.
Strings over \(\{0, 1, 2\}\)

\(M_2\): Strings where the sum of digits mod 3 is 0
Strings over \{0, 1, 2\}

\(M_2\): Strings where the sum of digits mod 3 is 0
Strings over \{0, 1, 2\}

\(M_2\): Strings where the sum of digits mod 3 is 0
What language does this machine recognize?
What language does this machine recognize?

The set of all binary strings with # of 1’s ≡ # of 0’s (mod 2) (both are even or both are odd).

Can you think of a simpler description?
Strings over \( \{0, 1, 2\} \)

\( M_1 \): Strings with an even number of 2’s

\( M_2 \): Strings where the sum of digits mod 3 is 0
Strings over \(\{0,1,2\}\) w/ even number of 2’s and mod 3 sum 0
Strings over \( \{0,1,2\} \) w/ even number of 2’s and mod 3 sum 0
Strings over \{0,1,2\} w/ even number of 2’s OR mod 3 sum 0
The set of binary strings with a 1 in the 3rd position from the start
The set of binary strings with a 1 in the 3\textsuperscript{rd} position from the start
The set of binary strings with a 1 in the 3rd position from the end
3 bit shift register  "Remember the last three bits"
The set of binary strings with a 1 in the 3^{rd} position from the end
The set of binary strings with a 1 in the 3\textsuperscript{rd} position from the end
The beginning versus the end
Adding Output to Finite State Machines

• So far, we have considered finite state machines that just accept/reject strings
  – called “Deterministic Finite Automata” or DFAs

• Now we consider finite state machines with output
  – These are the kinds used as controllers
Enter 15 cents in dimes or nickels
Press S or B for a candy bar
Basic transitions on **N** (nickel), **D** (dime), **B** (butterfinger), **S** (snickers)
Adding output to states: N – Nickel, S – Snickers, B – Butterfinger
Adding additional “unexpected” transitions to cover all symbols for each state