Lecture 21: DFAs and Finite State Machines with Output

How do bank machines work? WELL, LET'S SAY YOU WANT 25 DOLLARS. YOU PUNCH IN THE AMOUNT...

AND BEHIND THE MACHINE THERE'S A GUY WITH A PRINTING PRESS WHO MAKES THE MONEY AND STICKS IT OUT THIS SLOT.

SORT OF LIKE THE GUY WHO LIVES UP IN OUR GARAGE AND OPENS THE DOOR?

EXACTLY.
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>
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\}

\(M_1\): 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\}

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

\[s_0 \xrightarrow{0,1} s_0 \xrightarrow{2} s_1 \xrightarrow{2} s_1\]

\(M_2\): Strings where the sum of digits mod 3 is 0

\[t_0 \xrightarrow{0} t_1 \xrightarrow{1} t_2 \xrightarrow{0} t_0 \xrightarrow{1} t_1 \xrightarrow{2} t_2 \xrightarrow{2} t_0\]
What language does this machine recognize?
What language does this machine recognize?

The set of all binary strings with \# of 1’s \equiv \# 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

\[
\begin{align*}
\text{s}_0 \text{t}_0 & \\
\text{s}_1 \text{t}_0 & \\
\text{s}_0 \text{t}_1 & \\
\text{s}_1 \text{t}_1 & \\
\text{s}_0 \text{t}_2 & \\
\text{s}_1 \text{t}_2 & 
\end{align*}
\]
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?
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 3\textsuperscript{rd} 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 3\textsuperscript{rd} position from the end
3 bit shift register  “Remember the last three bits”
The set of binary strings with a 1 in the 3rd position from the end
The set of binary strings with a 1 in the 3rd 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 that with output
  – These are the kinds used as controllers
Vending Machine

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