CSE 311: Foundations of Computing

Lecture 21: DFAs and Finite State Machines with Output

[Comics illustrating a conversation about how bank machines work, involving a printing press worker behind the machine.]
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 \( \{0, 1, 2\} \)

\( \mathcal{M}_1 \): Strings with an even number of 2’s

\( \mathcal{M}_2 \): 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
What language does this machine recognize?

\[ s_0: \text{even} \# \text{of} 1's \text{ and even} \# \text{of} 0's \]

\[ s_3: \text{odd} \# \text{of} 1's \text{ and odd} \# \text{of} 0's \]

- Alternates 1's and 0's
- Strips 1st char is different from last
- Strips with even # of 1's
- Equal # of 1's and 0's

\[ \text{parity of } +1^n = \text{parity of } +0^n \]
What language does this machine recognize?

The set of all binary strings with \(#\text{ of } 1's \equiv \text{ # of } 0's \pmod{2}\) (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?
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 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 3\textsuperscript{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

\[ h \in \text{pos} \]
\[ n \geq 1 \]
\[ h + 2 \]
\[ \sum_{h=1}^{n \geq 2} \]
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