Spring 2015
Lecture 22: Finite state machines
review: finite state machines

- States
- Transitions on inputs
- Start state and final states
- The language recognized by a machine is the set of strings that reach a final state

<table>
<thead>
<tr>
<th>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>
applications of FSMs (aka finite automata)

• Implementation of regular expression matching in programs like `grep`
• Control structures for sequential logic in digital circuits
• Algorithms for communication and cache-coherence protocols
  – Each agent runs its own FSM
• Design specifications for reactive systems
  – Components are communicating FSMs
applications of FSMs (aka finite automata)

• Formal verification of systems
  – Is an unsafe state reachable?

• Computer games
  – FSMs provide worlds to explore
  – Character AI

• Minimization algorithms for FSMs can be extended to more general models used in
  – Text prediction
  – Speech recognition
start → Wander the Maze → Chase Pac-Man

Wander the Maze:
- Reach Central Base → Return to Base
- Spot Pac-Man → Lose Pac-Man
- Pac-Man Eats Power Pellet → Power Pellet Expires

Chase Pac-Man:
- Pac-Man Eats Power Pellet
- Eaten by Pac-Man

Return to Base:
- Eaten by Pac-Man
Timeout after two maximum segment lifetimes (2*MSL)

- **CLOSED**
  - Passive open
  - Send/SYN
  - SYN/SYN + ACK

- **LISTEN**
  - Passive open
  - Close
  - Send/SYN

- **SYN_RCVD**
  - SYN/SYN + ACK

- **SYN_SENT**
  - SYN/SYN + ACK

- **ESTABLISHED**
  - SYN + ACK/ACK

- **FIN_WAIT_1**
  - ACK
  - FIN + ACK/ACK

- **FIN_WAIT_2**
  - ACK

- **CLOSING**
  - FIN/ACK

- **CLOSE_WAIT**
  - Close/FIN

- **LAST_ACK**
  - Close/FIN

- **TIME_WAIT**
  - FIN/ACK
what language does this machine recognize?

The machine recognizes the language of strings over the alphabet \{0, 1\} where the number of 1s is even if the number of 0s is odd, or the number of 0s is even if the number of 1s is odd.
can we recognize these languages with DFAs?

- $\emptyset$

- $\Sigma^*$

- $\{ x \in \{0,1\}^* : \text{len}(x) > 1 \}$
FSM that accepts binary strings with a 1 three positions from the end
strings over \(\{0, 1, 2\}\)*

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

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

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

\[t_0 \xrightarrow{0} t_1 \xrightarrow{2} t_2 \xrightarrow{1} t_0\]
both: even number of 2’s and sum mod 3 = 0
DFA that accepts strings of a’s, b’s, c’s with no more than 3 a’s
“Remember the last three bits”

3 bit shift register

000 → 001 → 010 → 101 → 110 → 111 → 110 → 101 → 010 → 001 → 000 → …
start and accept states

---

\[
\begin{array}{c}
\text{s}_0 \xrightarrow{0,1} s_1 \xrightarrow{0,1} s_2 \xrightarrow{1} A \\
\end{array}
\]
FSMs with output

"Tug-of-war"

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>s₁</td>
<td>s₁</td>
<td>s₂</td>
</tr>
<tr>
<td>s₂</td>
<td>s₁</td>
<td>s₃</td>
</tr>
<tr>
<td>s₃</td>
<td>s₂</td>
<td>s₄</td>
</tr>
<tr>
<td>s₄</td>
<td>s₃</td>
<td>s₄</td>
</tr>
</tbody>
</table>

L → R → L → R → L → R

S₁ [Beep]  S₂  S₃  S₄ [Beep]
We’re only making $5.50/hour writing regular expressions.

Let’s design a vending machine.

“He does not think like normal people, and as a result his tests are quite difficult. His lectures are amusing and get the material across, but his office hours are not always too helpful. **Beware the vending machine final.**”

**Vending spec:**
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