Announcements

- Reading assignments
  - 7th Edition, Sections 13.3 and 13.4
  - 6th Edition, Section 12.3 and 12.4

Last lecture highlights

Directed graphs

Let $R$ be a relation on a set $A$. There is a path of length $n$ from $a$ to $b$ if and only if $(a,b) \in R^n$

Let $R$ be a relation on a set $A$. The connectivity relation $R^*$ consists of the pairs $(a,b)$ such that there is a path from $a$ to $b$ in $R$.

Transitive-Reflexive closure: Add the minimum possible number of edges to make the relation transitive and reflexive

The transitive-reflexive closure of a relation $R$ is the connectivity relation $R^*$
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 Finite State Machines (a.k.a. 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 Finite State Machines
(a.k.a. Finite Automata)

- Formal verification of systems
  - Is an unsafe state reachable?
- Computer games
  - FSMs provide worlds to explore
- Minimization algorithms for FSMs can be extended to more general models used in
  - Text prediction
  - Speech recognition

What language does this machine recognize?

- States $s_0$ to $s_3$
- Transitions on inputs
- Start state $s_0$
- Final state $s_3$
Design a DFA that accepts strings with a 1 three positions from the end.

How does the size of a DFA to recognize “10th character is a 1” compare with the size of a DFA to recognize “10th character from the end is 1”?
Strings over \{0, 1, 2\}*

M₁: Strings with an even number of 2’s

\[ \begin{array}{c}
S_0 \\
S_1 \\
\end{array} \]

M₂: Strings where the sum of digits mod 3 is 0

\[ \begin{array}{c}
S_0 \\
S_1 \\
S_2 \\
\end{array} \]

Recognize strings with an even number of 2’s and a mod 3 sum of 0

\[ \begin{array}{c}
S_0s_0 \\
S_1t_0 \\
S_2t_1 \\
\end{array} \]

State machines with output

<table>
<thead>
<tr>
<th>State</th>
<th>L</th>
<th>R</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>s₁</td>
<td>s₂</td>
<td>S₂</td>
</tr>
<tr>
<td>S₂</td>
<td>s₁</td>
<td>s₃</td>
<td>S₃</td>
</tr>
<tr>
<td>S₃</td>
<td>s₂</td>
<td>s₄</td>
<td></td>
</tr>
<tr>
<td>S₄</td>
<td>s₃</td>
<td>s₄</td>
<td>Beep</td>
</tr>
</tbody>
</table>

“Tug-of-war”

Vending Machine

Enter 15 cents in dimes or nickels
Press S or B for a candy bar
Vending Machine, Version 1

Basic transitions on N (nickel), D (dime), B (butterfinger), S (snickers)

Vending Machine, Version 2

Adding output to states: N – Nickel, S – Snickers, B – Butterfinger

Vending Machine, Final Version

Adding additional “unexpected” transitions