CSE 311: Foundations of Computing

Fall 2014
Lecture 23: DFA Minimization!
State Minimization

• Many different FSMs (DFAs) for the same problem
• Take a given FSM and try to reduce its state set by combining states
  – Algorithm will always produce the unique minimal equivalent machine (up to renaming of states) but we won’t prove this
State Minimization Algorithm

1. Put states into groups based on their outputs (or whether they are final states or not)

2. Repeat the following until no change happens
   a. If there is a symbol $s$ so that not all states in a group $G$ agree on which group $s$ leads to, split $G$ into smaller groups based on which group the states go to on $s$
Put states into groups based on their outputs (or whether they are final states or not)
State Minimization Example

Put states into groups based on their outputs (or whether they are final states or not).
State Minimization Example

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol $s$ so that not all states in a group $G$ agree on which group $s$ leads to, split $G$ based on which group the states go to on $s$
Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol $s$ so that not all states in a group $G$ agree on which group $s$ leads to, split $G$ based on which group the states go to on $s$
State Minimization Example

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol $s$ so that not all states in a group $G$ agree on which group $s$ leads to, split $G$ based on which group the states go to on $s$
Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol s so that not all states in a group G agree on which group s leads to, split G based on which group the states go to on s
State Minimization Example

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol $s$ so that not all states in a group $G$ agree on which group $s$ leads to, split $G$ based on which group the states go to on $s$
State Minimization Example

Can combine states S0-S4 and S3-S5.

In table replace all S4 with S0 and all S5 with S3.
Minimized Machine

**State Transition Table**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>S0 S1 S2 S3</td>
<td>S3</td>
</tr>
<tr>
<td>S1</td>
<td>S0 S3 S1 S3</td>
<td>S3</td>
</tr>
<tr>
<td>S2</td>
<td>S1 S3 S2 S0</td>
<td>S0</td>
</tr>
<tr>
<td>S3</td>
<td>S1 S0 S0 S3</td>
<td>S3</td>
</tr>
</tbody>
</table>

**Diagram**

- States: S0, S1, S2, S3
- Transitions:
  - S0 → S1 on input 0
  - S1 → S0 on input 0
  - S2 → S3 on input 3
  - S3 → S0 on input 1,3
  - S0 → S0 on input 1,2
  - S1 → S1 on input 1
  - S2 → S2 on input 2
  - S3 → S3 on input 3
another way to look at DFAs

Definition: The label of a path in a DFA is the concatenation of all the labels on its edges in order.

Lemma: x is in the language recognized by a DFA iff x labels a path from the start state to some final state.
nondeterministic finite automaton (NFA)

• Graph with start state, final states, edges labeled by symbols (like DFA) but
  – Not required to have exactly 1 edge out of each state labeled by each symbol— can have 0 or >1
  – Also can have edges labeled by empty string $\varepsilon$

• **Definition:** $x$ is in the language recognized by an NFA if and only if $x$ labels a path from the start state to some final state
goal: NFA to recognize...

Binary strings that have a 1 three positions from the end