Transitions…

- Semester “mid” point (Spring Break)
- From in-person to online classes
- From Part I to Part II of the course
  - Part I: Automata and Languages
  - Part II: Computability Theory
Going forward

- Will continue to use OSBLE+ for course management
- Will use Zoom for lecture meetings
  - Meetings will be recorded and made available for later review
- Will use Zoom for office hours
  - For both instructor and TA office hours
- HW submissions will happen online via OSBLE+
Equivalence & Minimization of DFAs

A topic not explicitly covered in Sipser
Applications of interest

- Comparing two DFAs:
  - \( L(DFA_1) = L(DFA_2) \)?

- How to minimize a DFA?
  1. Remove unreachable states
  2. Identify & condense equivalent states into one
When to call two states in a DFA “equivalent”? 

Two states $p$ and $q$ are said to be equivalent iff:

i) Any string $w$ accepted by starting at $p$ is also accepted by starting at $q$;

AND

ii) Any string $w$ rejected by starting at $p$ is also rejected by starting at $q$.

$\rightarrow p \equiv q$
Computing equivalent states in a DFA

**Table Filling Algorithm**

Pass #0
1. Mark accepting states ≠ non-accepting states

Pass #1
1. Compare every pair of states
2. Distinguish by one symbol transition
3. Mark = or ≠ or blank(tbd)

Pass #2
1. Compare every pair of states
2. Distinguish by up to two symbol transitions (until different or same or tbd)

(keep repeating until table complete)
Table Filling Algorithm - step by step

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>=</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=</td>
</tr>
</tbody>
</table>

Table:

- A = B = C = D = E = F = G = H

Diagram: [Diagram of a network with arrows and nodes labeled A to H]
Table Filling Algorithm - step by step

1. Mark X between accepting vs. non-accepting state
Table Filling Algorithm - step by step

1. Mark X between accepting vs. non-accepting state
2. Look 1-hop away for distinguishing states or strings
Table Filling Algorithm - step by step

1. Mark X between accepting vs. non-accepting state
2. Look 1-hop away for distinguishing states or strings
Table Filling Algorithm - step by step

1. Mark X between accepting vs. non-accepting state
2. Look 1-hop away for distinguishing states or strings

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table Filling Algorithm - step by step

1. Mark X between accepting vs. non-accepting state
2. Look 1-hop away for distinguishing states or strings
Table Filling Algorithm - step by step

1. Mark X between accepting vs. non-accepting state
2. Look 1-hop away for distinguishing states or strings
Table Filling Algorithm - step by step

1. Mark X between accepting vs. non-accepting state
2. Look 1-hop away for distinguishing states or strings
Table Filling Algorithm - step by step

1. Mark X between accepting vs. non-accepting state
2. Pass 1:
   Look 1-hop away for distinguishing states or strings
3. Pass 2:
   Look 1-hop away again for distinguishing states or strings
   continue….
Table Filling Algorithm - step by step

1. Mark X between accepting vs. non-accepting state
2. Pass 1:
   Look 1-hop away for distinguishing states or strings
3. Pass 2:
   Look 1-hop away again for distinguishing states or strings

   Equivalences:
   - A=B
   - C=H
   - D=G
Table Filling Algorithm - step by step

Retrain only one copy for each equivalence set of states

Equivalences:
- A=B
- C=H
- D=G
Q) What happens if the input DFA has more than one final state? Can all final states initially be treated as equivalent to one another?
How to minimize a DFA?

**Goal:** Minimize the number of states in a DFA

**Algorithm:**
1. Eliminate states unreachable from the start state
2. Identify and remove equivalent states
3. Output the resultant DFA

Depth-first traversal from the start state

Table filling algorithm
Are Two DFAs Equivalent?

1. Make a new dummy DFA by just putting together both DFAs
2. Run table-filling algorithm on the unified DFA
3. *IF* the start states of both DFAs are found to be equivalent, 
   *THEN*: $\text{DFA}_1 \equiv \text{DFA}_2$
   *ELSE*: different

Is $q_0 \equiv q_0'$?
: if yes, then $\text{DFA}_1 \equiv \text{DFA}_2$
: else, not equiv.