Course 8
Equivalence and Minimization of DFAs

The structure and the content of the lecture is based on http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/index.htm
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:

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

$\rightarrow p \equiv q$

AND

2) 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 (i.e. can not distinguish)

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) How the table on the right was obtained? Table Filling Algorithm
Table Filling Algorithm

- Recursive discovery of distinguishable states in a DFA
  - **Base case**: If \( p \) is an accepting state and \( q \) is not accepting then the pair \( \{p,q\} \) is distinguishable.
  - **Induction**: Let \( p, q \) be states s.t. for some input symbol \( a \), \( r = \delta(p, a) \) and \( s = \delta(q, a) \) are known to be distinguishable. Then the pair \( \{p,q\} \) is distinguishable.
Table Filling Algorithm - step by step

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

<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>

0 1 1 1 0 0 1 0
0 1 1 1 1 0 0 0
0 0 1 0 1 1 0 0
1 0 1 0 0 1 1 0
1 0 0 1 0 0 0 1
1 0 0 0 1 1 0 1
1 0 0 0 0 1 1 1
1 0 0 0 0 0 1 1
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 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>X</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>X</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>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</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>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

A =
B =
C =
D =
E =
F =
G =
H =
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….
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….

**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?
DFA Minimization by state equivalence method

0-Equivalence: \{A, B, C, F, G, H, Y\} \rightarrow \{D, E, Y\}
1-Equivalence \{A, B, C, H\}

\[\delta(A, 0) = C\]
\[\delta(B, 0) = D\]
\[\delta(C, 0) = Y\]
\[\delta(D, 0) = E\]
\[\delta(E, 0) = \top\]
\[\delta(F, 0) = E\]
\[\delta(G, 0) = \top\]
\[\delta(H, 0) = E\]

2-Equivalence \{A, B\}

\[\delta(A, 0) = C\]
\[\delta(B, 0) = A\]
\[\delta(C, 0) = C\]
\[\delta(D, 0) = E\]
\[\delta(E, 0) = \top\]
\[\delta(F, 0) = E\]
\[\delta(G, 0) = \top\]

3-Equivalence

\[\delta(A, 0) = C\]
\[\delta(B, 0) = A\]
\[\delta(C, 0) = C\]
\[\delta(D, 0) = E\]
\[\delta(E, 0) = \top\]
\[\delta(F, 0) = E\]
\[\delta(G, 0) = \top\]

Feasibility change!
DFA Minimization with unreachable states

Step 1. Eliminate the unreachable state.

Step 2. Proceed with state equivalence or table filling methods.
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
Summary

- Simplification of DFAs
  - How to remove unreachable states?
  - How to identify and collapse equivalent states?
  - How to minimize a DFA?