Before We Start

- Any questions?

Plan for today

- Minimization of DFAs

Languages

- Recall.
  - What is a language?
  - What is a class of languages?

Regular Languages

- What we know about regular languages
  - Described using regular expressions
    - Set operations of union, concatenation, Kleene Star
  - Kleene Theorem
    - A language is regular iff there exists a finite automata that accepts the language

Homework

- Homework #1 returned
- Homework #2 Due today
- Homework #3
  - Exercise 4.1.1a,c,e pg 129
  - Exercise 4.2.15 pg 148
  - Exercise 4.3.2 pg 153
  - Exercise 4.3.4 pg 154
  - Exercise 4.4.2 (a,b) pg 164
Minimal Finite Automata

- Motivation
  - Consider the question:
    - Do two finite automata accept the same language?
  - To answer, we introduce the Minimal Finite Automata (MFA)
    - Given a DFA, create a new DFA with the minimal number of states possible that accepts the same language.

- Plan
  - Equivalent states of a DFA
  - Devise an algorithm (based on equivalent states) that creates a minimal DFA from an DFA
  - Some examples

Minimal Finite Automata

- Equivalent States
  - $M = (Q, \Sigma, q_0, \delta, F)$
  - Two states, $p, q \in Q$ are said to be equivalent if
    - For all strings $x \in \Sigma^*$
      - $(p, x)$ is in an accepting state if $(q, x)$ is in an accepting state
        - If $(p, x)$ is an accepting state then $(q, x)$ is an accepting state
        - If $(p, x)$ is not an accepting state then $(q, x)$ is not an accepting state
    - If two states are not equivalent, they are said to be distinguishable.

- In building a MFA, equivalent states can be combined.
### Minimal Finite Automata

**Example**

- States C and G are distinguishable
  - One is accepting, one is not
- States A and G are distinguishable
  - $(A,\{0,1\}) = C$ (accepting)
  - $(G,\{0,1\}) = E$ (not-accepting)

- States B and H are equivalent
  - $\delta(B,1) = \delta(H,1) = C$
  - $\delta(B,0) = \delta(H,0) = G$
  - $\delta(B,0x) = \delta(H,0x) = E$ for any $x$
  - So for any $x$, $(B,x)$ and $(H,x)$ will either both be accepting or both not be accepted.

- States A and E are equivalent
  - $\delta(A,1) = \delta(E,1) = F$
  - $\delta(A,0) = B$, $\delta(E,0) = H$
  - B and H are equivalent
  - $(A,0x)$ and $(E,0x)$ will either both be accepting or both be non-accepting.

### Recursive algorithm to find distinguishable states:

- Consider pairs $\{p,q\}$
- For each pair we will determine whether $p$ is distinguishable from $q$
- Said another way, for each pair $\{p,q\}$ we will determine if $p$ is not equivalent to $q$. 

### Recursive algorithm

- **Base case:**
  - If $p$ is accepting and $q$ is non-accepting then $\{p,q\}$ is distinguishable
- **Induction**
  - For some pair $\{p,q\}$ if
    - $\delta(p,a) = r$ and $\delta(q,a) = s$ and
    - $(r,s)$ is distinguishable then
    - $\{p,q\}$ is distinguishable

---

[Diagram of a finite automaton]
Minimal Finite Automata

• Let’s take a look at this induction step
  – If \( r = \delta(p,a) \) and \( s = \delta(q,a) \) are distinguishable, then there is a string \( x \) such that \( \delta(r,x) \) is accepting and \( \delta(s,x) \) is not, or visa-versa
  – Then for \( x \), \( \delta(p,ax) \) is accepting and \( \delta(q,ax) \) is not, or visa-versa.
  – We found a string, \( ax \) such that \( \delta(p,ax) \) is accepting and \( \delta(q,ax) \) is not (or visa-versa), thus \( \{p,q\} \) are distinguishable.

Minimal Finite Automata

• This algorithm is sometime best visualized by using a table with each table cell representing a pair of states. A mark in a table cell indicates that the two states of the pair are distinguishable.

Minimal Finite Automata

• Distinguishable table

<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>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Minimal Finite Automata

• Restatement of algorithm
  – For all pairs \( \{p,q\} \) such that \( p \) is accepting and \( q \) is not, mark the equivalent cell in the table.
  – Consider each pair \( \{p,q\} \) not yet marked.
    • Determine \( r = \delta(p,a) \) and \( s = \delta(q,a) \) for each \( a \) in \( \Sigma \).
    • If \( \{r,s\} \) is marked, then mark \( \{p,q\} \)
  – Repeat until no further cells are marked during an iteration of the algorithm

Minimal Finite Automata

• Example

\[
\begin{align*}
\delta(A, 0) &= B \\
\delta(A, 1) &= F \\
\delta(B, 0) &= G \\
\delta(B, 1) &= C \\
\delta(C, 0) &= A \\
\delta(C, 1) &= C \\
\delta(D, 0) &= C \\
\delta(D, 1) &= G \\
\delta(E, 0) &= H \\
\delta(E, 1) &= F \\
\delta(F, 0) &= C \\
\delta(F, 1) &= G \\
\delta(G, 0) &= G \\
\delta(G, 1) &= E \\
\delta(H, 0) &= G \\
\delta(H, 1) &= C
\end{align*}
\]
Minimal Finite Automata
• Let’s try on our example

<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>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>x</td>
<td>x</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>
</tr>
<tr>
<td>D</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</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>
</tr>
<tr>
<td>F</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</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>x</td>
</tr>
</tbody>
</table>

Minimal Finite Automata
• Once our table is complete
  – All unmarked cells correspond to state pairs that are not-distinguishable, i.e. they are equivalent
  – Combine equivalent states into one
  – Transitions from equivalent states should map to equivalent states

Minimal Finite Automata
• Combine H and B

Minimal Finite Automata
• Combine E and A

Minimal Finite Automata
• Combine D and F
Minimal Finite Automata

• What have we done?
  – Defined the notion of equivalent states
  – Developed a recursive algorithm to determine which states in an FA are equivalent
  – Combine equivalent states to create FA with minimal number of states.

  – Questions?

Minimal Finite Automata

• Let’s revisit the question:
  – Given 2 specifications of regular languages, do the specifications describe the same language.
    • Create a MFA for each language
    • Compare the MFAs on a state by state basis.

For the mathematically minded

• Let’s go back to our Discrete Math
  – Relation
    • Defines relationship between objects
    • Usually given as an ordered pair, (x, y) where x, y ∈ some Set
  – Equivalence relation
    • Reflective: (a, a)
    • Symmetric: if (a,b) then (b,a)
    • Transitive: if (a,b) and (b,c) then (a,c)

For the mathematically minded

• Equivalence relations
  – The nice thing about equivalence relations
    • It partitions the elements of your set into a number of distinct and disjoint subsets.
    • Each subset is called an equivalence class

For the mathematically minded

• MFA and Equivalence Classes
  – State equivalence can be shown to be an equivalence relation on a language.
  – This relation partitions the strings of L into a number of equivalence classes.
  – Each equivalence class corresponds to a state in the MFA.

Minimal Finite Automata

• Questions?

  • Let’s take a break