Regular Language Equivalence and DFA Minimization

Equivalence of Two Regular Languages
DFA Minimization
Decision Property: Equivalence

• Given regular languages $L$ and $M$, is $L = M$?
• Algorithm involves constructing the *product DFA* from DFA’s for $L$ and $M$.
• Let these DFA’s have sets of states $Q$ and $R$, respectively.
• Product DFA has set of states $Q \times R$.
  • I.e., pairs $[q, r]$ with $q$ in $Q$, $r$ in $R$. 
Product DFA – Continued

- **Start state** = \([q_0, r_0]\) (the start states of the DFA’s for \(L, M\)).
- **Transitions**: \(\delta([q,r], a) = [\delta_L(q,a), \delta_M(r,a)]\)
  - \(\delta_L, \delta_M\) are the transition functions for the DFA’s of \(L, M\).
  - That is, we simulate the two DFA’s in the two state components of the product DFA.
Example: Product DFA
Equivalence Algorithm

- Make the final states of the product DFA be those states \([q, r]\) such that exactly one of \(q\) and \(r\) is a final state of its own DFA.
- Thus, the product accepts \(w\) iff \(w\) is in exactly one of \(L\) and \(M\).
Example: Equivalence
Equivalence Algorithm – (2)

- The product DFA’s language is empty iff \( L = M \).
- But we already have an algorithm to test whether the language of a DFA is empty.
Decision Property: Containment

• Given regular languages L and M, is \( L \subseteq M \)?
• Algorithm also uses the product automaton.
• How do you define the final states \([q, r]\) of the product so its language is empty iff \( L \subseteq M \)?

\textbf{Answer}: q is final; r is not.
Example: Containment

Note: the only final state is unreachable, so containment holds.
The Minimum-State DFA for a Regular Language

• In principle, since we can test for equivalence of DFA’s we can, given a DFA $A$ find the DFA with the fewest states accepting $L(A)$.

• Test all smaller DFA’s for equivalence with $A$.

• But that’s a terrible algorithm.
Efficient State Minimization

• Construct a table with all pairs of states.
• If you find a string that distinguishes two states (takes exactly one to an accepting state), mark that pair.
• Algorithm is a recursion on the length of the shortest distinguishing string.
State Minimization – (2)

• **Basis**: Mark a pair if exactly one is a final state.

• **Induction**: mark \([q, r]\) if there is some input symbol \(a\) such that \([\delta(q,a), \delta(r,a)]\) is marked.

• After no more marks are possible, the unmarked pairs are equivalent and can be merged into one state.
Transitivity of “Indistinguishable”

• If state $p$ is indistinguishable from $q$, and $q$ is indistinguishable from $r$, then $p$ is indistinguishable from $r$.

• **Proof**: The outcome (accept or don’t) of $p$ and $q$ on input $w$ is the same, and the outcome of $q$ and $r$ on $w$ is the same, then likewise the outcome of $p$ and $r$. 
Constructing the Minimum-State DFA

- Suppose \( q_1, \ldots, q_k \) are indistinguishable states.
- Replace them by one state \( q \).
- Then \( \delta(q_1, a), \ldots, \delta(q_k, a) \) are all indistinguishable states.
  - **Key point**: otherwise, we should have marked at least one more pair.
- Let \( \delta(q, a) = \) the representative state for that group.
### Example: State Minimization

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>{2,4}</td>
<td>{5}</td>
</tr>
<tr>
<td>{2,4}</td>
<td>{2,4,6,8}</td>
<td>{1,3,5,7}</td>
</tr>
<tr>
<td>{5}</td>
<td>{2,4,6,8}</td>
<td>{1,3,7,9}</td>
</tr>
<tr>
<td>{2,4,6,8}</td>
<td>{2,4,6,8}</td>
<td>{1,3,5,7,9}</td>
</tr>
<tr>
<td>{1,3,5,7}</td>
<td>{2,4,6,8}</td>
<td>{1,3,5,7,9}</td>
</tr>
<tr>
<td>* {1,3,7,9}</td>
<td>{2,4,6,8}</td>
<td>{5}</td>
</tr>
<tr>
<td>* {1,3,5,7,9}</td>
<td>{2,4,6,8}</td>
<td>{1,3,5,7,9}</td>
</tr>
</tbody>
</table>

Remember this DFA? It was constructed for the chessboard NFA by the subset construction.

Here it is with more convenient state names.
Example – Continued

Start with marks for the pairs with one of the final states F or G.
Example – Continued

Input r gives no help, because the pair [B, D] is not marked.
But input b distinguishes \{A,B,F\} from \{C,D,E,G\}. For example, [A, C] gets marked because [C, F] is marked.
Example – Continued

[C, D] and [C, E] are marked because of transitions on b to marked pair [F, G].
Example – Continued

[A, B] is marked because of transitions on r to marked pair [B, D].

[D, E] can never be marked, because on both inputs they go to the same state.
Example – Concluded

Replace D and E by H.
Result is the minimum-state DFA.
Eliminating Unreachable States

- Unfortunately, combining indistinguishable states could leave us with unreachable states in the “minimum-state” DFA.
- Thus, before or after, remove states that are not reachable from the start state.