Minimal State DFA

Theorem: The minimum state DFA for a regular language $L$ is unique up to isomorphism (renaming of states).

This is a corollary of the Myhill Nerode theorem and corresponds to the equivalence classes of $RL$.

Given any DFA $M$ for $L$, let $M'$ be a completely contained with some equivalence class of $RL$ (the machine for $RL$).

For any machine $M$, the number of states of $M'$ is less than or equal to the number of states of $M$.

Consider a machine $M$ which has the same number of states as $M'$.
We want to map the states of \( M \) to \( M' \).

Let \( q \) be a state of \( M \) then clearly for some \( x \in \Sigma^* \)
\[
\delta(q_0, x) = q \quad (\text{otherwise } M \text{ is not min state})
\]

\( q \to [x] \quad (\text{corresponding to } R_i) \)

for consistency verify that for any other \( y \)
\[
\delta(q_0, y) = q \quad [y] = [x]
\]

Since \( x \) and \( y \) belong to the same state they are in the same equiv claw \( \alpha \) \( \delta(q_0, y) = q \quad [y] = [x] \)

To complete the proof show that
\[
\delta(q, a) = \delta([x], a) = [xa]
\]

(this is how \( M' \) was constructed)
Constructing min state DFA

Since this is related to \( R_L \) which is above string \( x, y \) behaving identically with respect to accepting/non-accepting for any concatenate string \( z \), no accepting state can be equivalent to non-accepting state because it must satisfy \( R_L \) for \( z = \varepsilon \).

\[
\begin{array}{c|ccc}
& 0 & 1 & \varepsilon \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{array}
\]

Consider a pair of states \( p, q \)
We cannot club \( p \) and \( q \) if for some string \( z \in \Sigma^* \) \( \delta(p, z) \in Q - F \) and \( \delta(q, z) \in F \) or vice versa

\( \Rightarrow \) If \((p', q')\) are not eqv then \( p, q \) are not eqv if \( \delta(p, a) = p' \) \( \delta(q, a) = q' \)
Define a graph whose vertices correspond to \( Q \times Q \)
and edges correspond to:

Initially, "mark" all pairs known to be not equivalent, viz \( F \times (Q - F) \)

A special case of "coarsest partitioning problem"
Try it out this:

Only \((A, B)\) can be clubbed

\[
\begin{align*}
(A, B) & \xrightarrow{0} (B, B) \\
(A, B) & \xrightarrow{1} (C, C)
\end{align*}
\]

Since the states are same in both cases, they cannot be distinguished by any thing.

Therefore \(A, B\) can be clubbed & the final new state automaton will be

The transitions of the new machine can be thought of as graph-contraction, i.e., once vertices are merged, the edges are between merged vertices.

Argue why we cannot have a situation after merging vertices, i.e., no non-deterministic transitions.
Example

\[ \rightarrow A \rightarrow B \rightarrow C \rightarrow D \]

Paris

\[ \begin{align*}
\{A, B\} & \quad \{A, C\} \\
\{B, C\} & \quad \{B, D\}
\end{align*} \]

\[ \begin{align*}
\{A \ Relative \ AD\} \\
\{C\} & \quad \{D\}
\end{align*} \]

Since one is accepting and the other is not

\[ \{A, C\} \quad \{B, D\} \]

can be merged according to this graph leading to the automaton

\[ \rightarrow \{A, C\} \rightarrow \{B, D\} \]

i.e. the minimum state DFA for \((00)^*\)