DFA Minimization

DFA generated from regular expressions via the construction REG → NFA → DFA is in general not the “smallest” possible DFA.

Some states are unreachable, some are redundant (i.e. have similar behavior to other states).

Example consider DFA

\[
\begin{align*}
q_0 \ a & \rightarrow q_1 & q_0 \ b & \rightarrow q_3 \\
q_1 \ b & \rightarrow q_2 & q_1 \ a & \rightarrow q_2 \\
q_2 \ b & \rightarrow q_2 & q_2 \ a & \rightarrow q_2 \\
q_3 \ a & \rightarrow q_3 & q_3 \ b & \rightarrow q_3
\end{align*}
\]

- $q_0$ start state  
- $q_1,q_2$ final states

States $q_1$ and $q_2$ show identical behavior.
DFA Minimization Idea

- Remove unreachable states, i.e. states which cannot be reached from the start state.
- Build equivalence classes among states via a fixpoint construction.
- Two states \((q,q')\) cannot be equivalent if one is a final state and the other is not.
- If from \((q1,q1')\) we can reach \((q2,q2')\) via \(q1 \ a \rightarrow q2\) and \(q1' \ a \rightarrow q2'\) and we know that \((q2,q2')\) cannot be equivalent, then \((q1,q1')\) cannot be equivalent either.
DFA Minimization Algorithm

Given DFA $M = (\Sigma, Q, \delta, q_0, F)$.

1. Remove unreachable states.

2. Setup marking tables of pairs $(q, q')$ where $q \neq q'$.
   
   (a) Mark all pairs $(q, q')$ where $q \in F$ and $q' \not\in F$ (and vice versa). (These are the states which cannot be equivalent)
   
   (b) For each unmarked pair $(q, q')$ and $a \in \Sigma$ if $(\delta(q, a), \delta(q', a))$ is marked, then mark $(q, q')$.
   
   (c) Repeat until there are no more changes.
3. Combine states.
   For each unmarked \((q, q')\)
   (a) If \(p \cdot a \rightarrow q'\) then add \(p \cdot a \rightarrow q\).
   (b) If \(q' \cdot a \rightarrow p\) then add \(q \cdot a \rightarrow p\).
   (c) Remove \(q'\).
   (d) Remove \(p \cdot a \rightarrow q'\), \(q' \cdot a \rightarrow p\) for all \(p \in Q\) and \(a \in \Sigma\) (i.e. remove \(q'\) and all transitions leading to and from \(q'\)).

4. Resulting DFA is minimal.
Example

q0 start, q1, q2 final states
q0 a -> q1  q0 b -> q3
q1 b -> q2  q1 a -> q2
q2 b -> q2  q2 a -> q2
q3 a -> q3  q3 b -> q3

Marking table (step 2.):
(q0,q1) marked  (q1,q0) marked
(q0,q2) marked  (q2,q0) marked
(q0,q3)        (q3,q0)
(q1,q2)        (q2,q1)
(q1,q3) marked  (q3,q1) marked
(q2,q3) marked  (q3,q2) marked
Combine states (step 3.):

Consider unmarked (q1,q2) we have that

Step 3a.)
q2 a -> q2  q1 a -> q2
q2 b -> q2  q1 b -> q2
therefore add
q2 a -> q1  q1 a -> q1
q2 b -> q1  q1 b -> q1

Step 3b.)
q2 a -> q2  q2 b -> q2
therefore add
q1 a -> q2  q1 b -> q2

Step 3c,d.) remove q2 and its transitions

Resulting DFA:
q0 a -> q1  q0 b -> q3
q1 b -> q1  q1 a -> q1
q3 a -> q3  q3 b -> q3