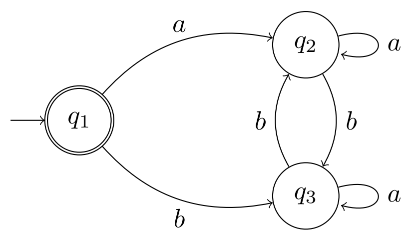
Optimizing automata

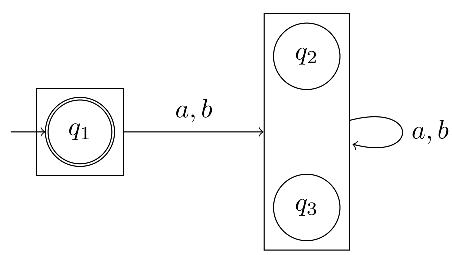
April 6

One advantage of having a clear machine model is that we can reason about optimizations. One optimization we could do for DFAs is to reduce the number of states.

For example, the following DFA clearly recognizes the language $\{\epsilon\}$:



In a sense, the states q_2 and q_3 are equivalent: if we start processing a string x in either of them, we will always get the same answer. So we can lump them together into a single big "metastate":



We can generalize this idea. Let $\sim$ be the equivalence relation on Q defined by

$$q_1 \sim q_2 \text{ iff } \forall x \in \Sigma^*, \widehat{\delta}(q_1, x) \in A \iff \widehat{\delta}(q_2, x) \in A$$

This formalizes the idea that if we start processing x in q_1 or in q_2, we will always get the same answer. If we know $\sim$, we can construct an equivalent machine M_{min} as follows:

- The states Q_{min} are equivalence classes of states of M: $Q_{min} = Q_M / \sim$

- The accepting states of Q_{min} are the equivalence classes of accepting states of M. Note that if $q_1 \in A_M$ and $q_2 \sim q_1$ then $q_2 \in A_M$ (plug ϵ into the definition of $\sim$).
- The initial state of Q_{min} is just $[q_{0M}]$.
- The transition function δ_{min} is given by $\delta_{min}([q], a) = [\delta_M(q, a)]$. This is well-defined (proof by contradiction).