Formally, a Nondeterministic Finite Automaton or NFA is defined by a 5-tuple
\[(Q, \Sigma, \delta, S, F)\]

where
\[Q\] : Finite set of States
\[\Sigma\] : Alphabet (finite set of symbols)
\[\delta \in Q \times \Sigma \rightarrow \mathcal{P}(Q)\] : Transition Function
\[S \subseteq Q\] : Initial States
\[F \subseteq Q\] : Accepting (or Final) States

Note:
- The transition function maps a state and an input symbol to zero or more successor states. Thus an NFA has “choice”; hence “nondeterministic”.
- However, nothing ambiguous about the language defined by an NFA! Not the case that some word \(w \in L(A)\) sometimes, and \(w \notin L(A)\) other times for some NFA \(A\).
- How? By considering all possible states simultaneously.

The Subset Construction (1)

Observations:
- An NFA can be in one of a set of states.
- When reading an input symbol, the machine enters one of a new set of states.
- Which are the sets of possible states?
- Each set is a subset of \(Q\), so the set of possible states is (at most) \(\mathcal{P}(Q)\).
- But \(Q\) is finite. Thus \(\mathcal{P}(Q)\) is finite too!
- There may be lots of states as \(|\mathcal{P}(Q)| = 2^{|Q|}\)
- But the number of states is finite!

The Subset Construction (2)

We can thus convert an NFA into a DFA by considering each possible set of NFA states as a single DFA state!