Here are some notes on material from section 2.3.

• The Extended Transition Function of an NFA

As with a DFA, we can define the *extended transition function* of an NFA. If the transition function is δ, we usually denote the extended transition function by $\hat{\delta}$. The basis is that

$$\hat{\delta}(q, a) := \{q\}.$$

For the induction step, let S be $\hat{\delta}(q, x)$. Then

$$\hat{\delta}(p, xa) := \bigcup_{p \in S} \delta(p, a).$$

• The Subset Construction

In order to show that DFAs and NFAs have the same computational power, we gave the *subset construction*, which, given an NFA, constructs a DFA that accepts the same language.

- The alphabet of the new DFA is the same as that of the NFA.
- If Q is the set of states of the given NFA, then the set Q' of states of the new DFA is $\mathcal{P}(Q)$, the power set of Q, that is, the set of all subsets of Q. In another words, a state of the new DFA is a *set of states* of the NFA.
- If q_0 is the start state of the NFA, then $\{q_0\}$ is the start state of the new DFA.
- A state in the new DFA is accepting if it *contains* an accepting state of the NFA.
- If δ is the transition function of the NFA, then we define the transition function δ' of the new DFA as follows. Where S is a subset of Q and a is a symbol:

$$\delta'(S, a) := \bigcup_{p \in S} \delta(p, a).$$

• Proving that the Subset Construction Works

The point of the subset construction is to make a DFA that accepts the same language as a given NFA. The main thing to be proven is that the extended transition functions of the two automata are the same. We do this by induction. For the induction step, we need to show that, for each state q in Q, each string x, and each symbol a,

$$\text{if} \quad \hat{\delta}'(\{q\}, x) = \hat{\delta}(q, x), \quad \text{then} \quad \hat{\delta}'(\{q\}, xa) = \hat{\delta}(q, xa).$$

To do this, we use the above definitions, along with the definition of the extended transition function of a DFA.

• Equivalence of DFAs and NFAs

Now, we can show that DFAs and NFAs have the same computational power.