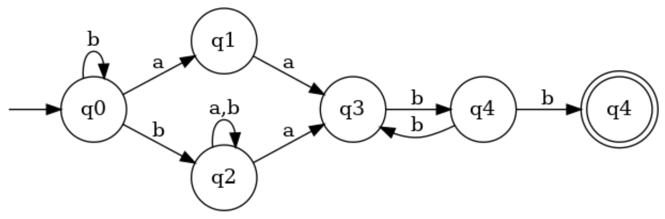
When converting from an NFA to a DFA, don't forget the ϵ-transitions!

#3 said that you must use the construction from the proof of Theorem 3.3 to build the NFA, even if you can easily build one directly from the regular expression. The examples from 3/27 include a description of the construction of a machine for $L(r^*)$. In this construction, ϵ-transitions are added to connect M's final states to a new start state q_0', which is also made a final state. In some cases, the new start state can be omitted and M's final states connected to its start state q_0, which is also made a final state. Both were accepted, as long as the second case occurred in a situation where it was safe.

The NFA-to-RE algorithm discussed in class requires a single final state. For #4b, this needs to be handled first.

Be careful to ensure that all paths through a state have been accounted for before removing that state. Consider the following:



(Having two states labelled q_4 was a typo — for the purposes of discussion, let's call them q_4 and final-q_4.) There's a two-step cycle involving q_3 and q_4, but there's no "middle" state with no other out transitions, so a better way to handle this is to think of the paths through q_4 — condense $q_3 \xrightarrow{b} q_4 \xrightarrow{b}$ final-q_4 to $q_3 \xrightarrow{bb}$ final-q_4 and $q_3 \xrightarrow{b} q_4 \xrightarrow{b} q_3$ to $q_3 \xrightarrow{bb} q_3$, then remove q_4.

Remember that while there is only a single start state, there can be more than one final state. For constructing a machine M^R for language L^R, you can't just swap the start and final states if M has more than one final state. Be sure to handle that case.