Converting automata to regular expressions

March 27

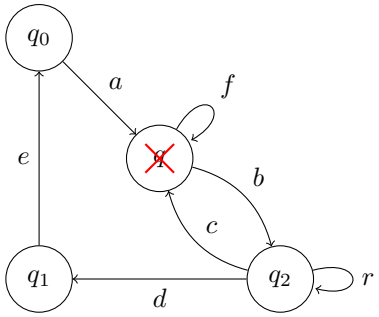
In lecture we completed the proof or Kleene's theorem by showing that every NFA-recognizable language is regular. That is, given an NFA N, we will construct a regular expression r such that $L(r) = L(N)$.

0.1 NFA to regular expression conversion

To show that every NFA-recognizable langauge is regular, we must build a regular expression out of an NFA.

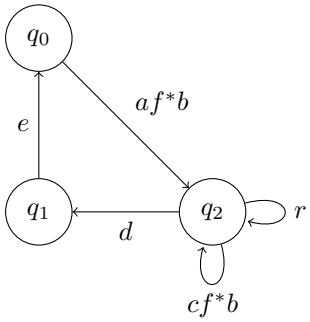
In order to do so, we'll build a "generalized NFA" whose edges are labelled by regular expressions instead of just symbols from Σ. We will start with the given NFA and repeatedly remove states until only the start state and an accepting state remain. We will then be able to read the regular expression from the single transition from the start state to the accepting state.

At each step, we will keep the language of the machine the same. To do so, we must add in new transitions that capture the paths that went through the removed state. For example, suppose we wanted to remove the boxed state q from the following machine:



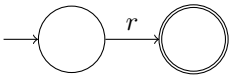
To do so, we must add in edges capturing all of the paths from the remaining nodes, through q, and back to the remaining nodes that are lost. For example, it is possible to transition from q_0 to q_2 by passing through q using any string matching af^*b. So we will add a new transition from q_0 to q_2 annotated with this regular expression. Similarly, we can transition from q_2 back to itself with the string cf^*b, so we will add a self loop to q_2 with the annotation cf^*b. In

general, we must consider every pair of remaining states q', q'', and add a new edge corresponding to any path from q' to q, through a loop on q, and back to q'. In this case, we have the following:



Note that there are now two edges from q_2 to itself. This means that we can transition from q_2 using either a string matching r *or* a string matching cf^*b. This is equivalent to a single transition labelled by $r + cf^*b$. In general we can combine all duplicate transitions this way.

We want to remove states until we are left with a single start and accept state. To avoid having to remove accepting states we will add a single new accepting state with an epsilon transition from each old accepting state into it. We will also add a new start state with an epsilon transition to the old start state; this guarantees that there are no transitions back into the start state. These two modifications allow us to remove states until we reach the following machine:



This machine clearly accepts $L(r)$. It also has the same language as our starting machine, so $L(r) = L(N)$, which was our goal.

Thus every NFA-recognizable language is regular.