CS 360
Naomi Nishimura

Kleene's Theorem

To make the various algorithms clearer, we prove Kleene's Theorem in five steps. Some of these steps, as noted below, have been modified from what appears in the textbook.

A. For any regular expression α, there exists an ϵ-NFA E such that $L(E) = L(\alpha)$ [Section 3.2.3]

B. For any ϵ-NFA E, there exists an NFA N such that $L(N) = L(E)$. [Modified version of Theorem 2.22 in Section 2.5.5, which creates a DFA instead of an NFA]

C. For any NFA N, there exists a DFA D such that $L(D) = L(N)$. [Section 2.3.5] (This is also known as subset construction.)

D. For any DFA D, there exists an ϵ-NFA E such that $L(E) = L(D)$. [Modified version of Theorem 2.12 in Section 2.3.5, which creates an NFA instead of an ϵ-NFA]

E. For any ϵ-NFA E, there exists a regular expression α such that $L(\alpha) = L(E)$. [Modified version of Section 3.2.2, which uses a DFA instead of an ϵ-NFA] (This is also known as state elimination.)

1 Step B

For Step B, we convert an ϵ-NFA into an NFA rather than a DFA. This removes the subset construction from the conversion, making clearer how ϵ-transitions are removed.

In our modified result, we form an NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$ from an ϵ-NFA $E = (Q_E, \Sigma, \delta_E, q_E, F_E)$, as follows. We retain the set of states and start state, so $Q_N = Q_E$ and $q_N = q_E$.

To form the new transition function, we consider the set of states reachable from a state upon the reading of a single input symbol. In the ϵ-NFA, this will be states reached by following any number of ϵ-transitions followed by a transition on the symbol followed by any number of ϵ-transitions. This can be accomplished by taking eclose of the original state followed by a transition on the symbol followed by another eclose. This is in fact exactly how the extended transition function δ_E^* is defined, so we can set $\delta_N(q, a) = \delta_E^*(q, a)$.

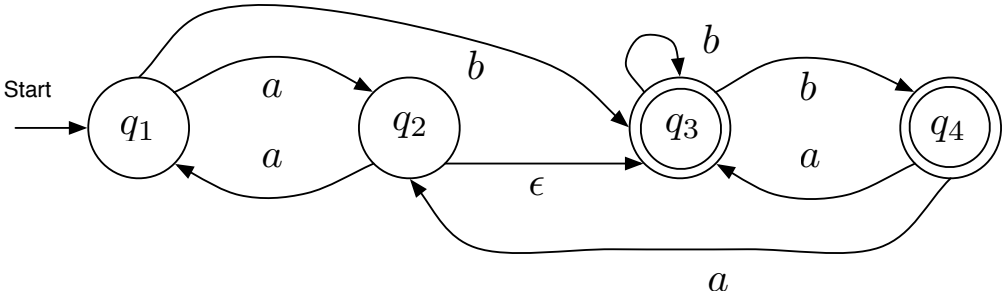
Finally, to define F_N, we observe that F_N should contain every state in F_E plus possibly the start state. The start state should be included if it is possible to reach an accepting state by following zero or more ϵ-transitions. Put another way, the start state should be included if $\text{eclose}(\{q_E\}) \cap F_E \neq \emptyset$, that is $F_N = F_E \cup \{q_E\}$ if $\text{eclose}(\{q_E\}) \cap F_E \neq \emptyset$, and $F_N = F_E$ otherwise.

2 Step E

In Section 3.2.2 of the textbook, an algorithm is given for constructing a regular expression from a DFA. The algorithm presented here (and in class) is simpler to understand, and applies to NFA's and ϵ-NFA's as well.

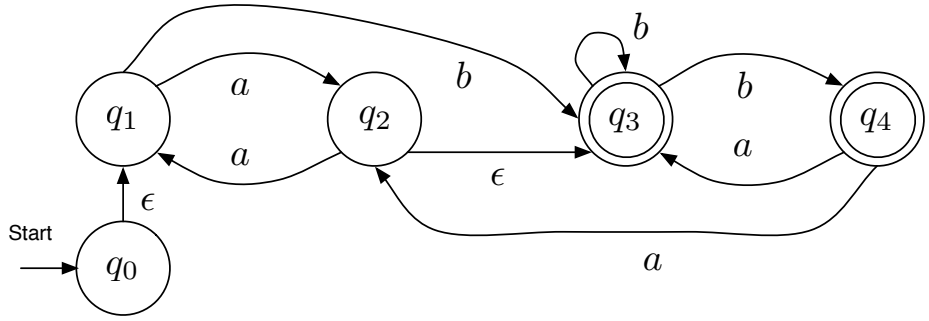
As in the textbook, we will remove states from the automaton, replacing labels of arcs, so that in the end a single regular expression is formed. The single regular expression will be the label on an arc that goes from the start state to the accepting state, and this will be the only arc in the automaton.

The algorithm forms the simpler automaton as follows. In step 1, we modify the automaton to have a start state that is not an accepting state and has no transitions in (either self-loops or from other states). In step 2, we create an equivalent automaton that has a single accepting state with no transitions out. These will be the two states that remain at the end of the algorithm. In step 3, the other states are eliminated, in any order. Details of the algorithm follow, along with a running example, illustrated below.



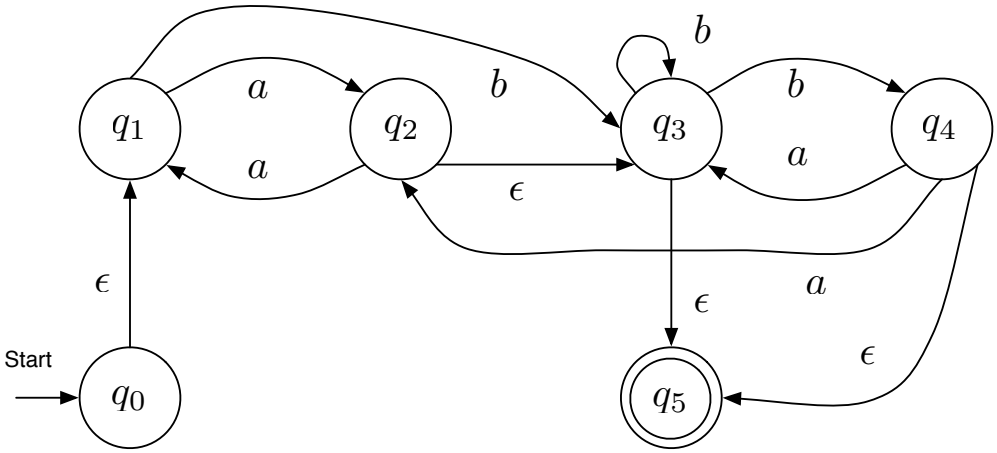
Step 1

If the start state is an accepting state or has transitions in, add a new non-accepting start state and add an ϵ-transition between the new start state and the former start state.



Step 2

If there is more than one accepting state or if the single accepting state has transitions out, add a new accepting state, make all other states non-accepting, and add an ϵ-transition from each former accepting state to the new accepting state.

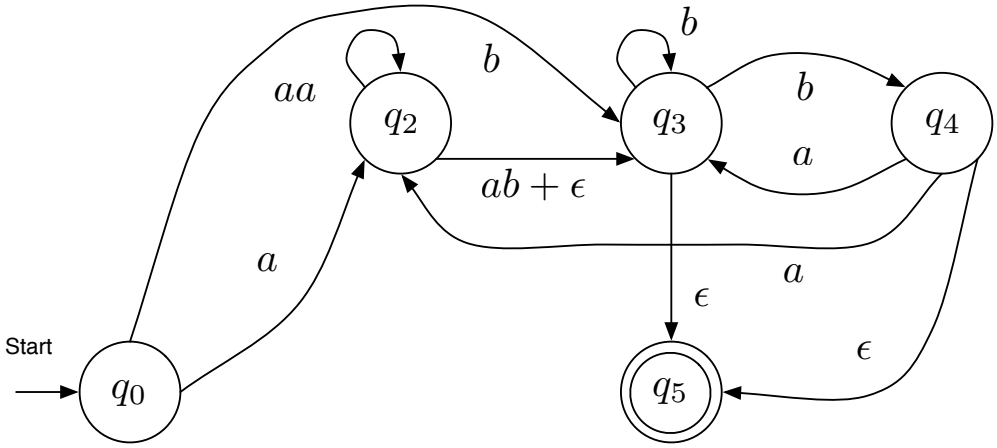


Step 3

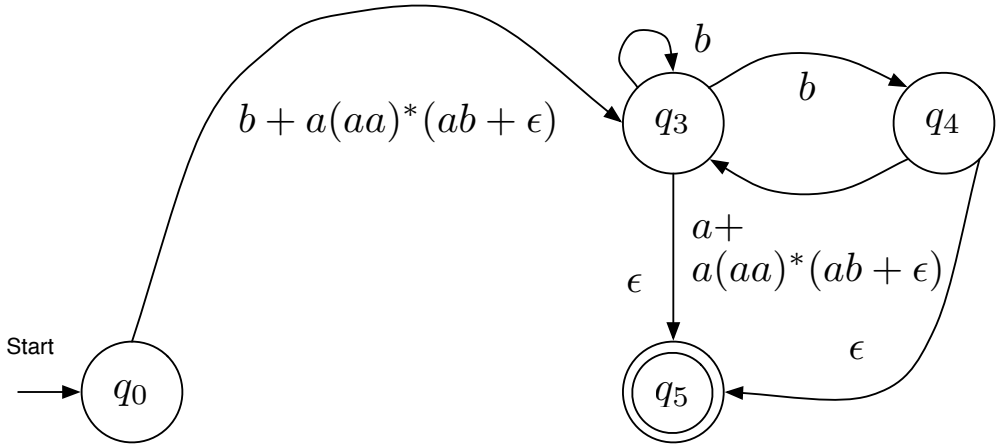
For each non-start non-accepting state in turn, eliminate the state and update transitions according to the procedure given on page 99 of the textbook, Figures 3.7 and 3.8. The following illustrations depict the removal of states q_1, q_2, q_3, and q_4 in that order.

As the transitions into q_1 are from q_0 and q_2 and the transitions out of q_1 are to q_2 and q_3, there will be new or modified transitions from q_0 to q_2 and q_2 and from q_2 to q_2 and q_3.

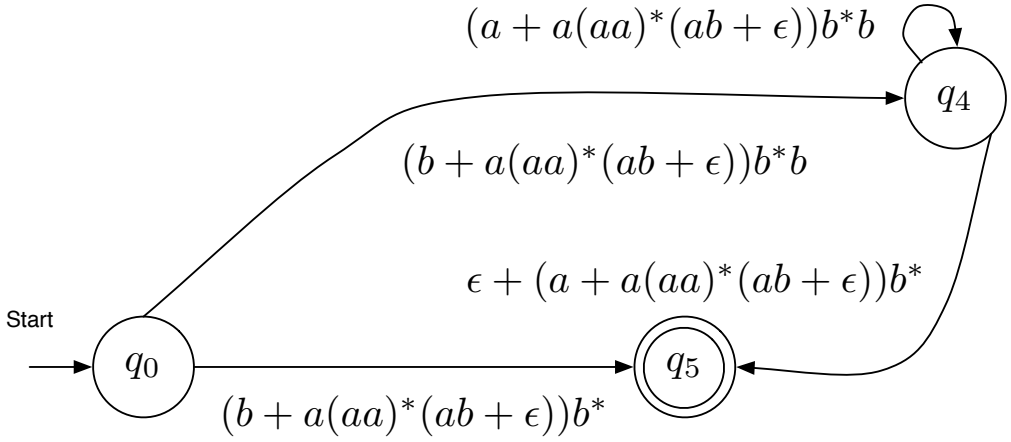
Notice, for example, that there is now a self-loop on q_2 consisting of the concatenation of the previous labels on the transitions from q_2 to q_1 and q_1 to q_2. In addition, observe that the label on the transition from q_2 to q_3 consists of the previous label ϵ and the new label formed by transitions from q_2 to q_1 to q_3.



Since the only transition out of q_2 is to q_3, it is only the transitions into q_3 from q_0 and q_4 that are changed in this step. In both cases, notice that the self-loop appears as $(aa)^*$ in each label.



The elimination of q_3 results in b^* to represent the self-loop. This label, as well as the labels from q_4 to q_3 and q_3 to q_4, result in a self-loop on q_4.



In the final step, we obtain the union of the label from q_0 to q_5 and the label formed by the transition from q_0 to q_4, zero or more uses of the self-loop, and the transition from q_4 to q_5.

$$(b + a(aa)^*(ab + \epsilon))b^* +$$
$$((b + a(aa)^*(ab + \epsilon))b^*b)\,((a + a(aa)^*(ab + \epsilon))b^*b)^*$$
$$(\epsilon + (a + a(aa)^*(ab + \epsilon))b^*)$$