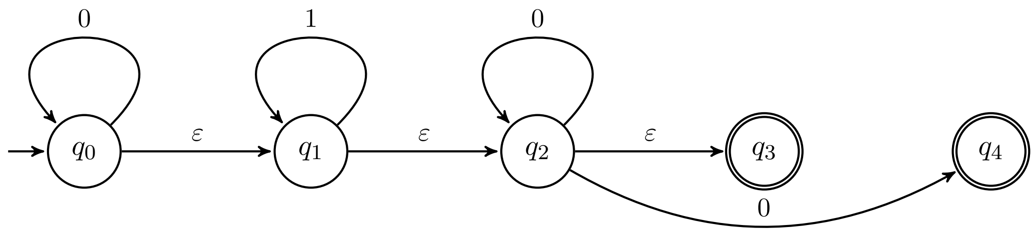
CS 137: Algorithms in the Real World — Spring 2021

Epsilon Closure Example

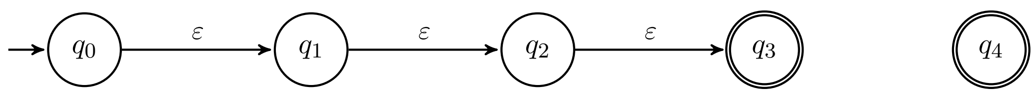
Throughout this document, we will be using the running example of the following NFA to explain `epsilonClosure`:



Step 1: Adding the Necessary Epsilon Transitions

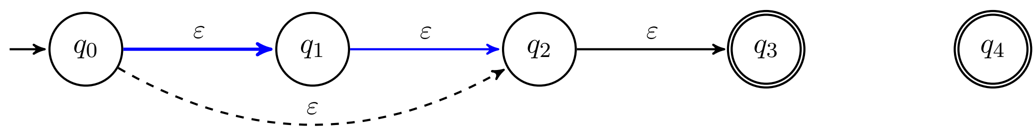
The first step of writing the Epsilon Closure is to add *direct* ε-transitions for any *reachable* ones. Another way of thinking about this is that we're determining, for each state, which states are reachable using *only* ε-transitions. For example, in the example above, q_1, q_2, and q_3 are all reachable from q_0 using only ε-transitions.

For this step, we only consider the ε-transitions:

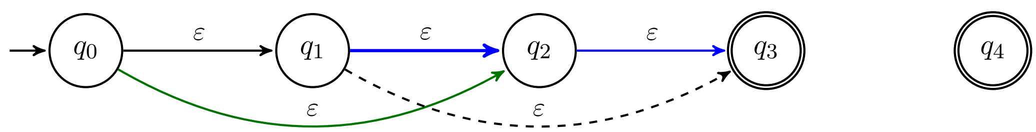


For each transition from a to b, we find all ε-transitions that start at b and add an ε-transition from a to c:

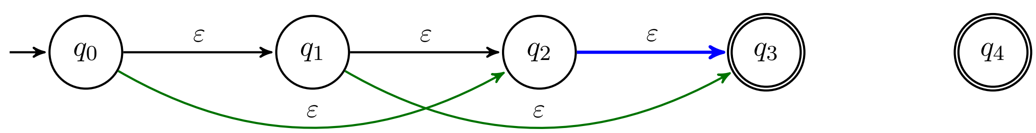
Checking $q_0 \to q_1$:



Checking $q_1 \to q_2$:

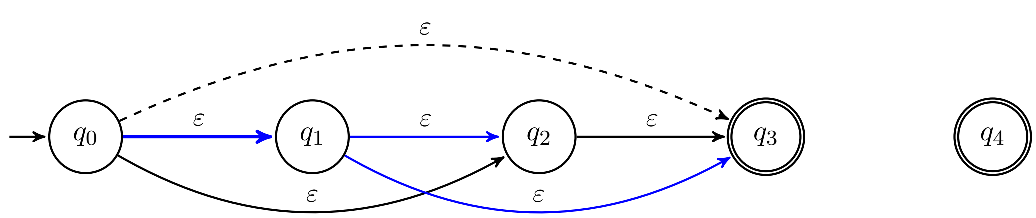


Checking $q_2 \to q_3$:

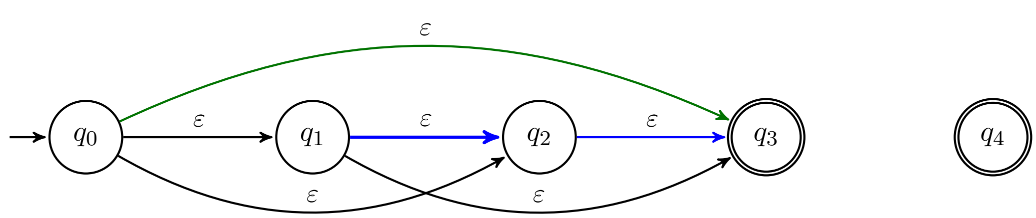


We repeat this loop until the ε-transitions we start with are the same as the ones we end with.

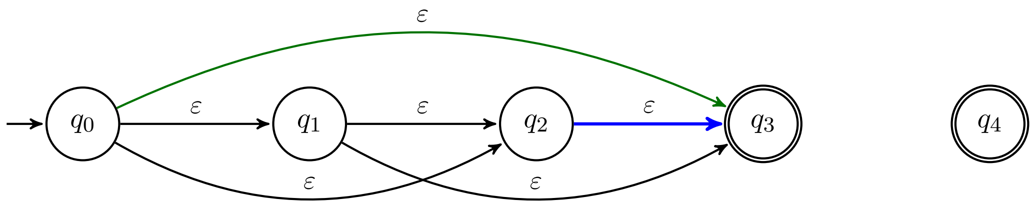
Checking $q_0 \to q_1$:



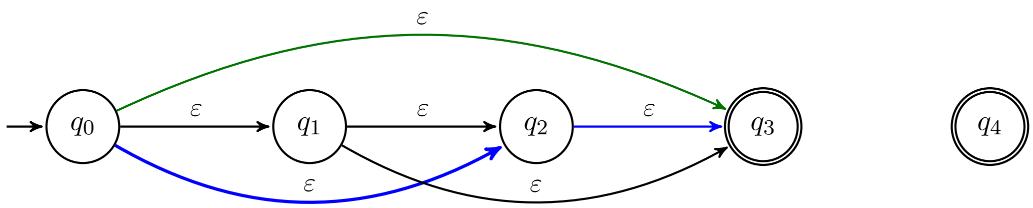
Checking $q_1 \to q_2$:



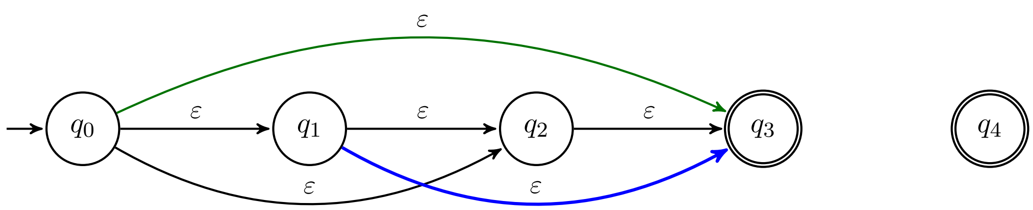
Checking $q_2 \to q_3$:



Checking $q_0 \to q_2$:

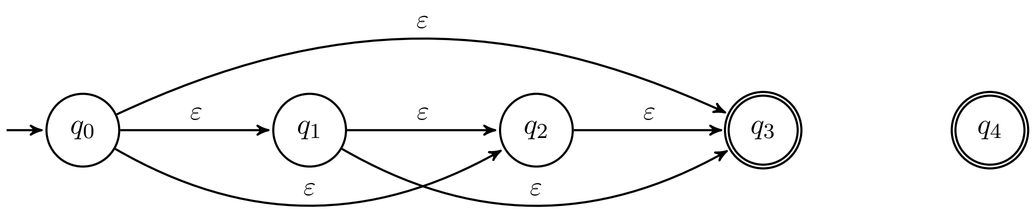


Checking $q_1 \to q_3$:



Notice that the *green* transitions are the ones that were new in this particular iteration of the loop. If we do one more iteration, we will see that no new green transitions are added. That is the cue to stop.

Checking $q_x \to q_y$:



This process can be summed up in the following psuedocode:

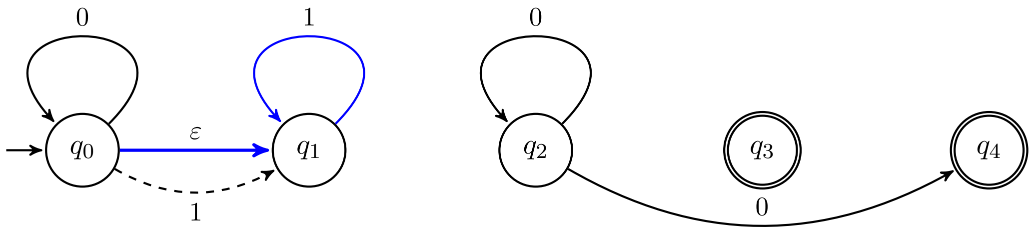
```
old = <empty set>
curr = <set of all epsilon transitions>
while (old != curr) {
   old = curr
   curr = <empty set>
   for (a → b in old) {
      curr.add(a → b)
   }
   for (a → b in old) {
      for (b → c in old) {
         curr.add(a → c)
      }
   }
}
```

Now that we're done with adding ε-transitions, we have a complete set of transitions such that if we can reach one state from another state using ε-transitions, there is a direct transition we can take.

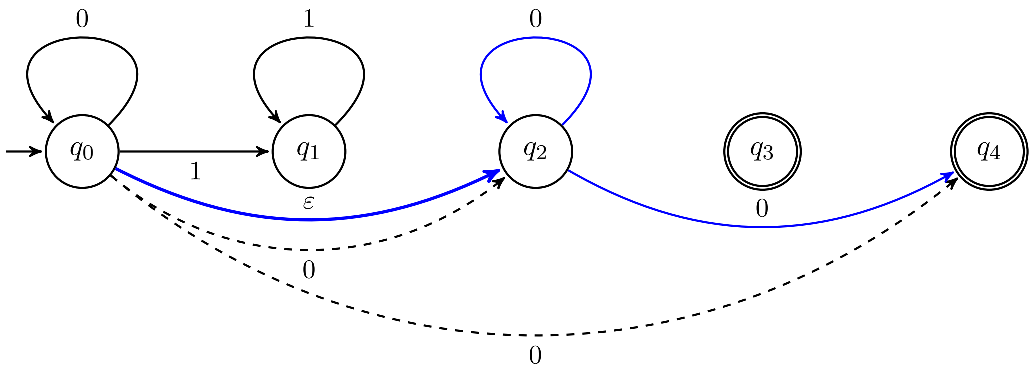
Step 2: Replacing The Epsilon Transitions

The second step consists of completely replacing the ε-transitions so that our NFA only has non-ε-transitions. To do this, we do a process very similar to step one, except we now join an ε-transition to a *non*-ε-transition. The ε-transitions are $q_0 \to q_1$, $q_0 \to q_2$, $q_0 \to q_3$, $q_1 \to q_2$, $q_1 \to q_3$, and $q_2 \to q_3$. We replace them in order:

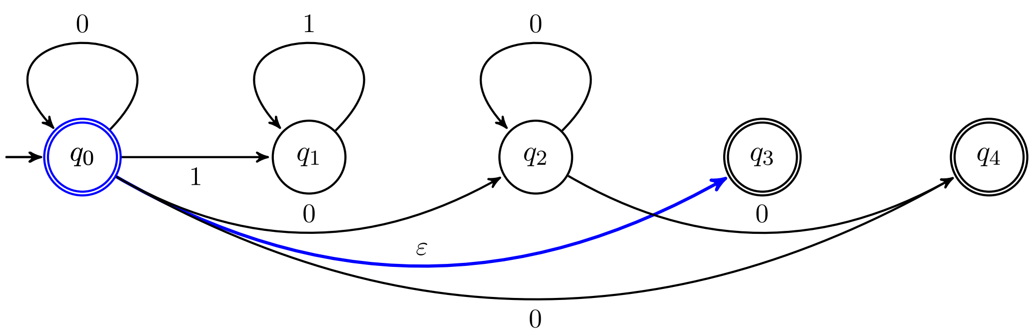
Replacing $q_0 \to q_1$:



Replacing $q_0 \to q_2$:

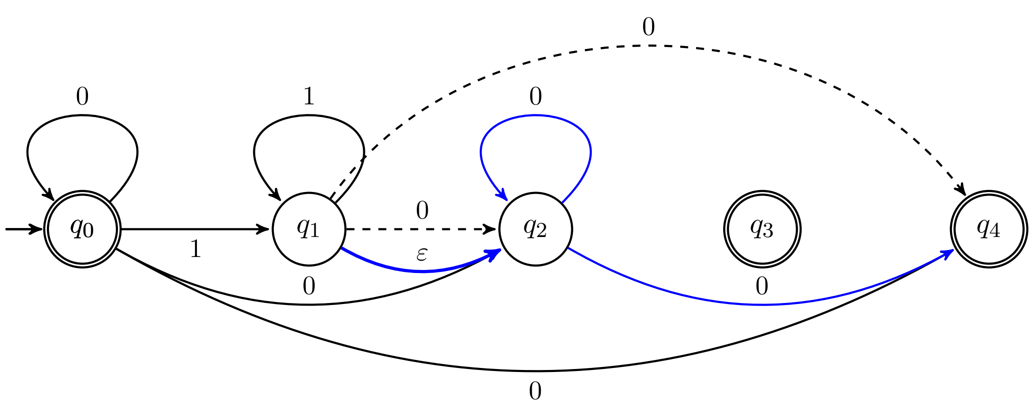


Replacing $q_0 \to q_3$:

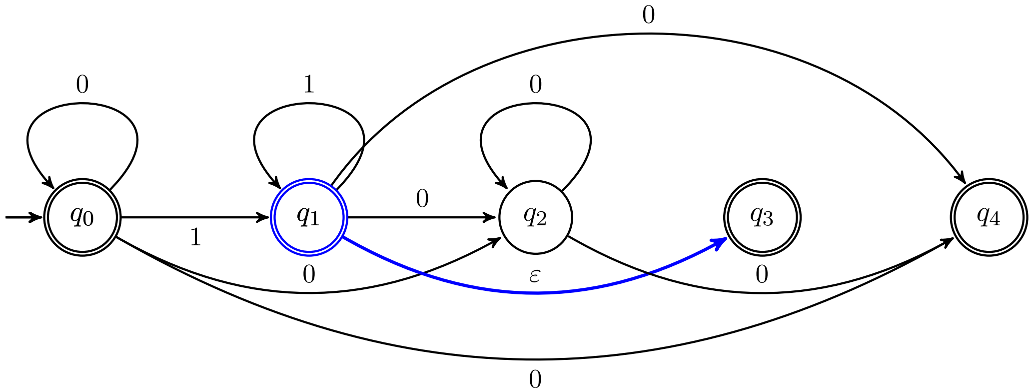


Note that since we can get to a final state on an ε, we need to make q_0 a final state!

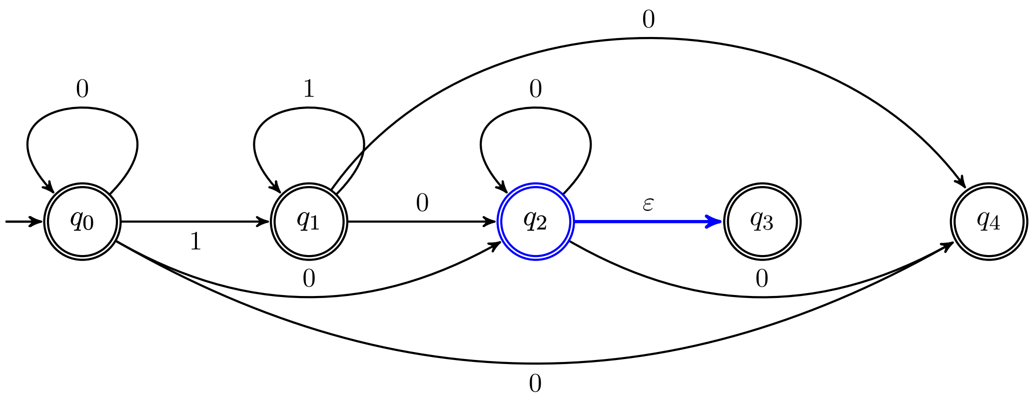
Replacing $q_1 \to q_2$:



Replacing $q_1 \to q_3$:



Replacing $q_2 \to q_3$:



Since we've replaced the ε-transitions one-by-one, we're left with an NFA equivalent to the original one that has no ε-transitions: