A RegEX ⇒ NFA ⇒ FA Example

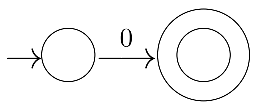
Consider again the RegEX over $\Sigma = \{0, 1\}$ below

$$\alpha = (0^*1^*)^* \tag{1}$$

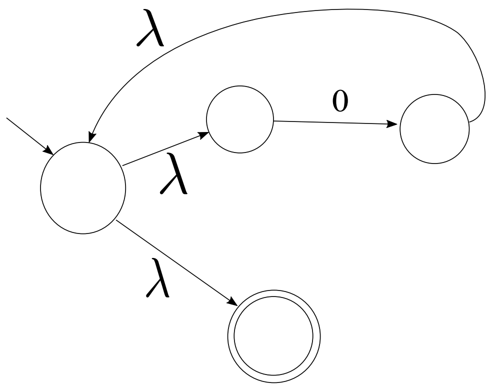
We will first build an NFA from it using a shortcut of Kleene's construction, and then we will apply our *NFA ⇒ FA process* to build an equivalent FA.

If we follow Kleene's proof/construction verbatim, then the first step would be to build an NFA for 0,

0 :



then build the Kleene closure of (2) as



0^* :

One then builds identically the NFA for 1^* in two steps (only the label 0 changes to label 1) and continues with the NFA for the concatenation the NFA for 0^* —above— with the NFA for 1^* (not shown) and finally builds the

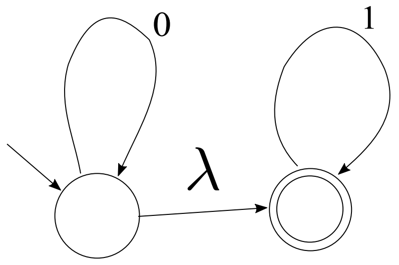
NFA for (1) in the way the proof of Kleene's theorem goes.

In this simple case we proceed guided by the definition of string acceptance in our shortcut construction.

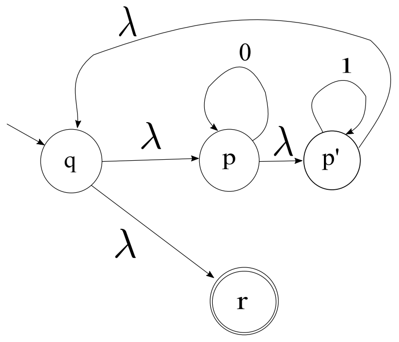
First, the following is clearly an NFA that accepts all strings described by 0^*1^*. The se are all strings of the form

$$\{0^n1^m : n \geq 0 \wedge m \geq 0\} \tag{2}$$

We see by inspection that the NFA below accepts precisely the strings in (2) since *the only possible accepting-path labels* —0^n1^m— that we can get in the design below, and, indeed, *we get all of them*, for al $n \geq 0, m \geq 0$.



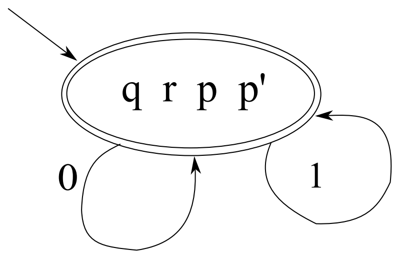
According to the proof of Kleene's theorem we get the NFA for the RegEX (1) as follows from the NFA above:



(2)

where we added state names for the next step, NFA⟹FA.

The FA is the following.



(2)

Note that q is the start state but all of p, r and q' are in $\lambda(q)$. Moreover, see how the 0-successor of the FA state "q, r, p, p'" is computed: We find that only p from the NFA above has a 0 successor, and that is p.

But you can easily compute that $\lambda(p) = \{q, r, p, p'\}$. So on input 0 the FA goes back to $\{q, r, p, p'\}$.

Similarly exactly, the 1-successor of $\{q, r, p, p'\}$ in the FA is $\{q, r, p, p'\}$.

Incidentally, the FA above proves again in a different way that *the language of* the RegEX $(0+1)^*$, which the FA trivially decides is the *same as the language of* the RegEX $(0^*1^*)^*$.