L₁ = {x ∈ {0, 1}⁺ : x has an odd number of 1s}. How to design a regular expression r₁ such that L₁ = L(r₁).

**Directly:** L₀ = {x : x has an even number of 1s} (={L₁}) has the regular expression r₀ = (0⁺1₀¹₀⁺)* (with two levels of nesting). So take r₁ = r₀ • 0¹₀⁺ since adding one 0 to "even" makes "odd".

This gives (0⁺1₀¹₀⁺)*0¹₀⁺ which simplifies to (0⁺1₀¹₀⁺)¹₀⁺ or to (0⁺1₀₁⁺)0¹₀⁺ — see why?

**By Machine:** Design M₁ = 

\[ M = \begin{array}{ccc}
S & q₀ & q₁ \\
q₀ & \epsilon & a & \delta \\
q₁ & a & \epsilon & \delta \\
\end{array} \]

so L(M₁) = L₁.

**By the formula (to be):** given in class, L(M₁) = (0⁺1₀¹⁺)¹₀⁺.

Also 0¹₀⁺(0⁺1₀¹⁺)⁺. How can all these different answers be equally correct? We'll come to this in the last weeks on simplicity.

**Example 2:** Build a machine for r = (00)⁺(11)⁺. The NFA really needs an ε-edge for the concatenation • but not elsewhere:

\[ M = \begin{array}{ccc}
S & qₓ & q_y \\
qₓ & a & b & \epsilon \\
q_y & \epsilon & \epsilon & \delta \\
\end{array} \]

We didn't need to make the start state accepting — it now has an ε-edge to the accepting state f. But what we need to know is S = \{qₓ, q_y\} for the DFA M: