Regular Language Limits

Write an RE (or $\equiv$ DFA) for the sequence w containing an equal number of occurances of the substring 01 and 10 ($\Sigma = \{0, 1\}$). **Alternatively**, explain why it can't be done.
For instance, 101 is in the language because it has one 10 and one 01, but 1010 is not (**two** 10s, just one 01). λ, 111, and 00000 are also part of the language.

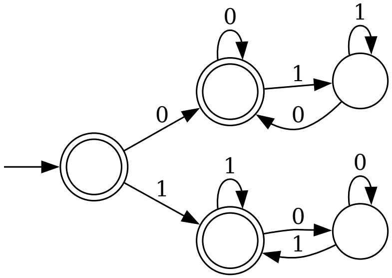
Regular Language Limits

Write an RE (or $\equiv$ DFA) for the sequence w containing an equal number of occurances of the substring 01 and 10 ($\Sigma = \{0, 1\}$). **Alternatively**, explain why it can't be done.
For instance, 101 is in the language because it has one 10 and one 01, but 1010 is not (**two** 10s, just one 01). λ, 111, and 00000 are also part of the language.

Yes we can!

$$\lambda \mid 0^{+}(1^{+}0^{+})* \mid 1^{+}(0^{+}1^{+})*$$

The insight here is that if we begin with a 0 (for instance), then we must always see at least one 0 after 1^{+} is encountered.
IOW: the first and last symbol must be the same.



Regular Language Limits

It's tempting to feel the "equal number of" criteria prevents these patterns from being a **regular language**. But in this case the patterns that are counted are not truely recursive — **they overlap**. The 0 in 101 gets used in both the 10 count and the 01 count.

It is the **arbitrary number** of **recursively nested** structures that regular languages cannot describe.

In a real language theory class, we would be learning the **pumping lemma**. A mathematic tool that can be used to show definitively whether a language is regular or not.

Yes we can!

$$\lambda \mid 0^+(1^+0^+)* \mid 1^+(0^+1^+)*$$

The insight here is that if we begin with a 0 (for instance), then we must always see at least one 0 after 1^+ is encountered.
IOW: the first and last symbol must be the same.