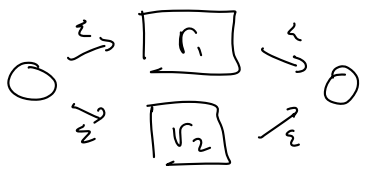
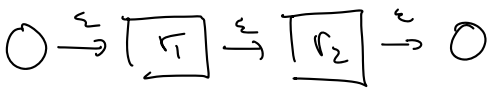
Regex → NFA

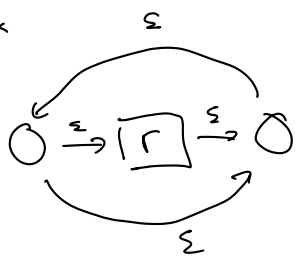
$r_1 + r_2$



$r_1 r_2$

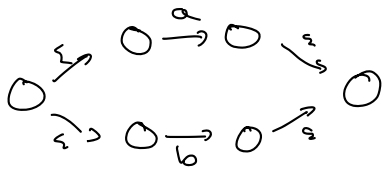


r^*

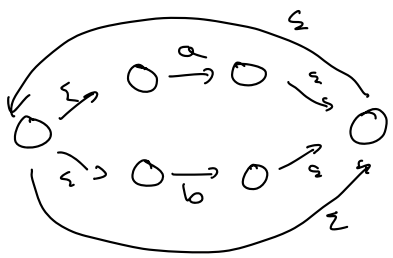


$ab(a+b)^*$

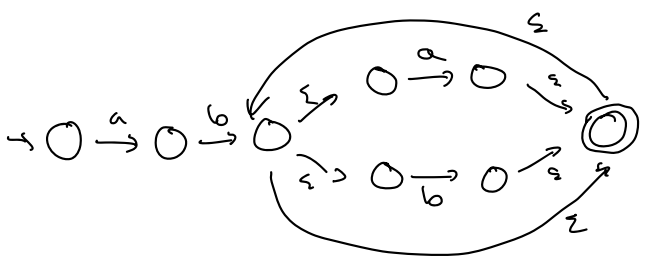
$a+b$:



$(a+b)^*$:

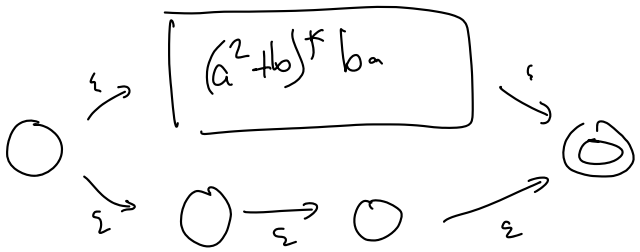


$ab(a+b)^*$:

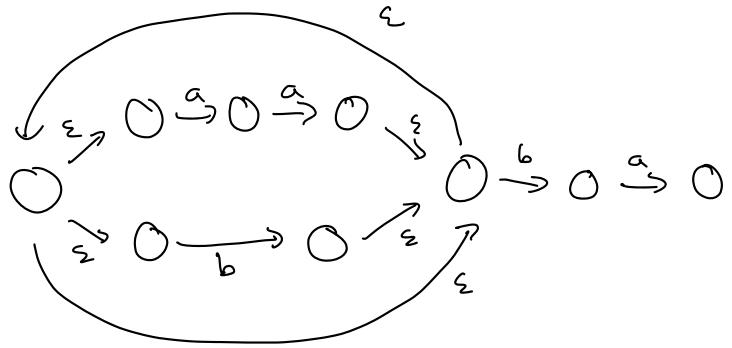


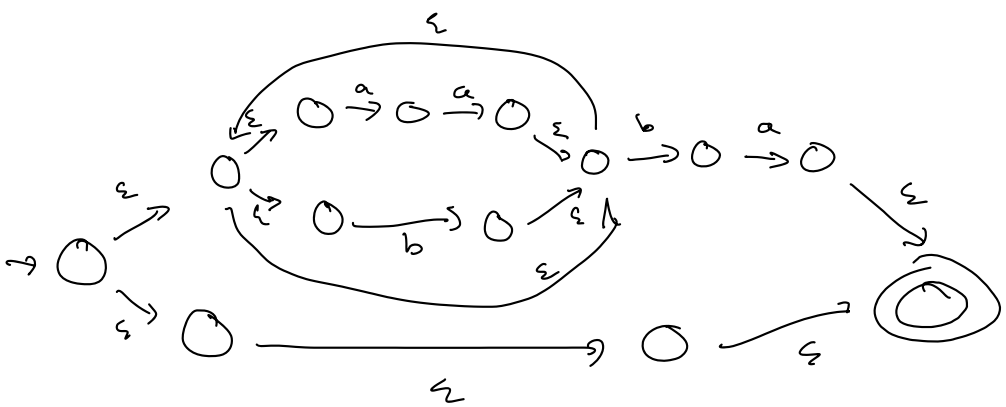
Make an NFA for

$(a^2+b)^* ba + \varepsilon$



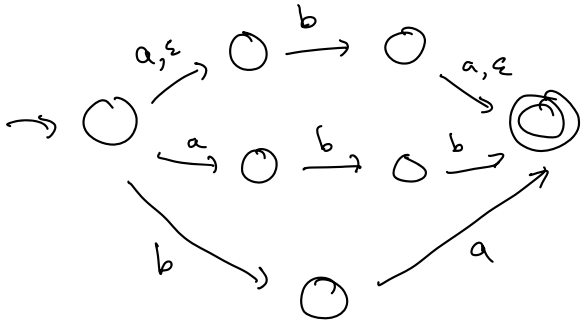
$(a^2+b)^* ba$





T'other way:

NFA → regex



convert to regex.

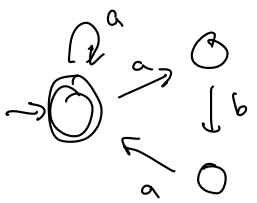
Imagine all possible paths from start to accepting.

3 basic paths, so it'll look like

$(a+\varepsilon)b(a+\varepsilon) + abb + ba$

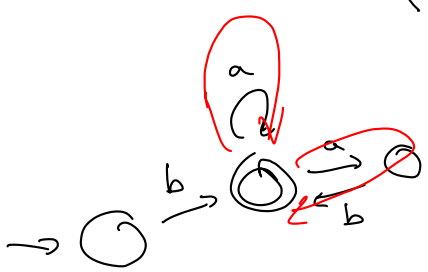
This is harder if there are loops.

Ex



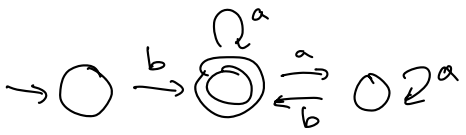
2 loops: a or aba, either can be done several times, in any order

$(a + aba)^*$



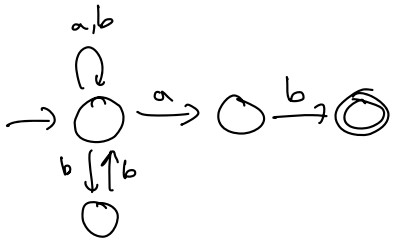
2 loops: a or ab, gotta do b first.

$b(a+ab)^*$



$b(a + aa^*b)^*$

$b(a + \cancel{ab} + aa^*b)^*$



$(a + b)^* ab$

$(a^*ab) + (b^*ab)$

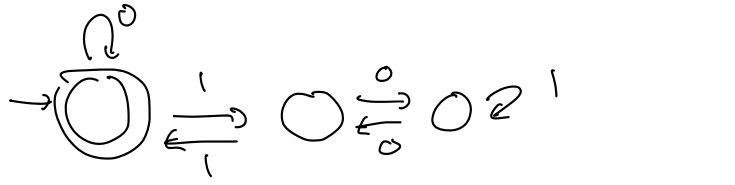
We can convert NFA to regex,
and also regex to NFA

So the langs expressible by regex are
the same as those of NFAs.
These are the regular languages.

So the expressive power of
DFA, NFA, regex are all the same:
The regular languages!

The DFA for binary #s div. by 3.

should be expressible as a regex.



$$(0 + 1(01^*0)^*1)^*$$

Next: non-regular langs!