Three ways to think about NFAs

"Outside Observer": is there a path labeled by x from the start state, to the final state (if we know the input in advance can we tell the NFA which decisions to make)

"Perfect Guesser": The NFA has input x, and whenever there is a choice of what to do, it **magically** guesses a transition that will eventually lead to acceptance (if one exists)

"Parallel exploration": The NFA computation runs all possible computations on x in parallel (updating each possible one at every step)

NFA that recognizes "binary strings with a 1 in the third position from the end"

"Perfect Guesser": The NFA has input x, and whenever there is a choice of what to do, it **magically** guesses a transition that will eventually lead to acceptance (if one exists)

Perfect guesser view makes this easier.

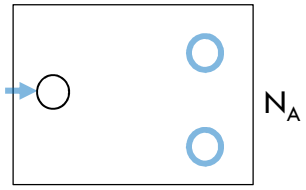
Design an NFA for the language in the title.

Pollev.com/uwcse311

Let $P(A)$ be "There is an NFA whose language is the same as the language for A."

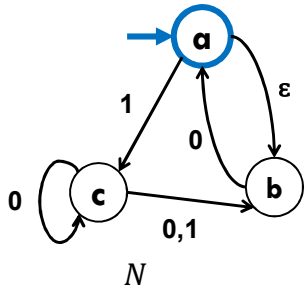
Inductive Hypothesis: Let A, B be arbitrary regular expressions. Suppose $P(A)$ and $P(B)$.

Inductive Step: Case A^*

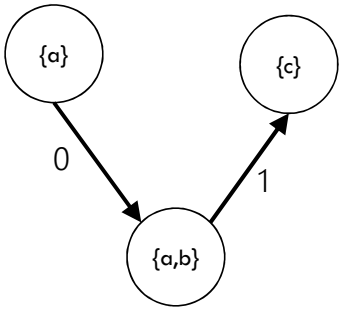


Want a machine that accepts exactly strings matched by A^*.

An example (starting point)



N



D