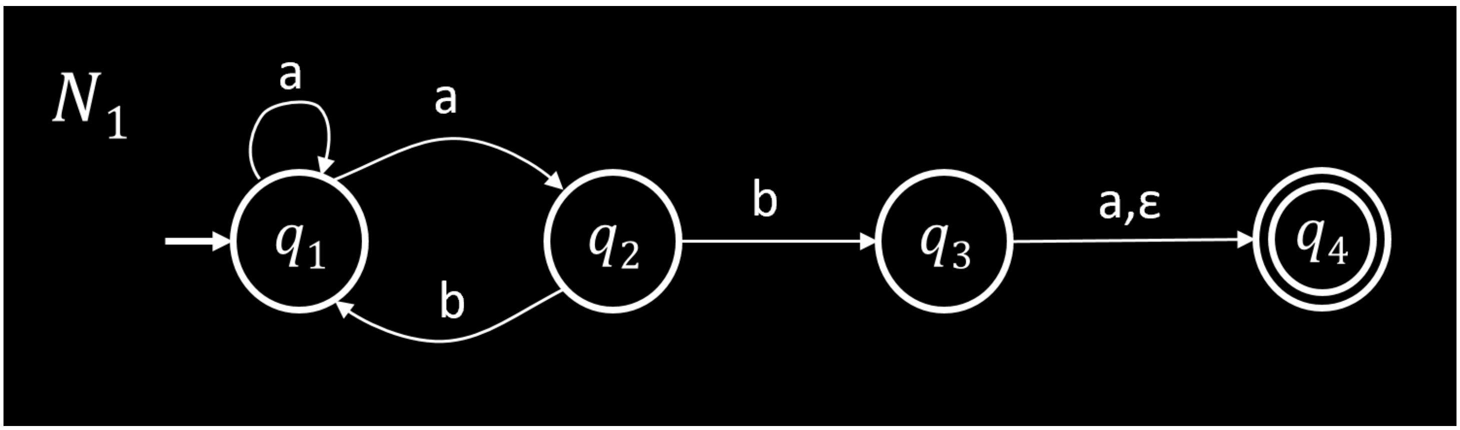
Nondeterministic Finite Automata

So far, we have been talking about deterministic finite automata (DFA)

A Nondeterministic finite automata (NFA) is essentially a DFA but we can have multiple transitions per string

Ways to think about the nondeterminism:

- Computational: fork new parallel thread whenever you have multiple options and accept if any thread reaches accept
- Mathematical: tree with branches and accept if any branch reaches accept
- Magical: guess at each step which way to go, and a machine will always make a right guess that leads to accepting



An empty string ε automatically produces a fork (either you move along it or you don't)

Nondeterministic Finite Automaton

A nondeterministic finite automaton N is a 5-tuple $N = (Q, \Sigma, \delta, q_0, F)$

Same as before except for δ

$$\delta : Q \times \underbrace{\Sigma_\varepsilon}_{\Sigma \cup \{\varepsilon\}} \to \underbrace{\mathcal{P}(Q)}_{\text{power set}} = \{R | R \subseteq Q\}$$

- Power set of Q is the set of all subsets of Q

NFAs and Languages

If $A = L(N)$ for NFA N then A is regular

Proof:
Let $N = (Q, \Sigma, \delta, q_0, F)$ and we wish to construct some DFA $M = (Q', \Sigma, \delta', q_0', F')$. The idea is that we want to capture in our DFA all possible subsets that we could be at in N

- $Q' = (Q) = \text{power set of } Q$
- $\delta'(R, a) = \{q | q \in \delta(r, a) \text{ for some } r \in R\}$
  - $R \in Q'$
  - Needs to be slightly modified to include empty transitions
- $q_0' = \{q_0\}$
- $F' = \{R \in Q' | R \cap F \neq \emptyset\}$

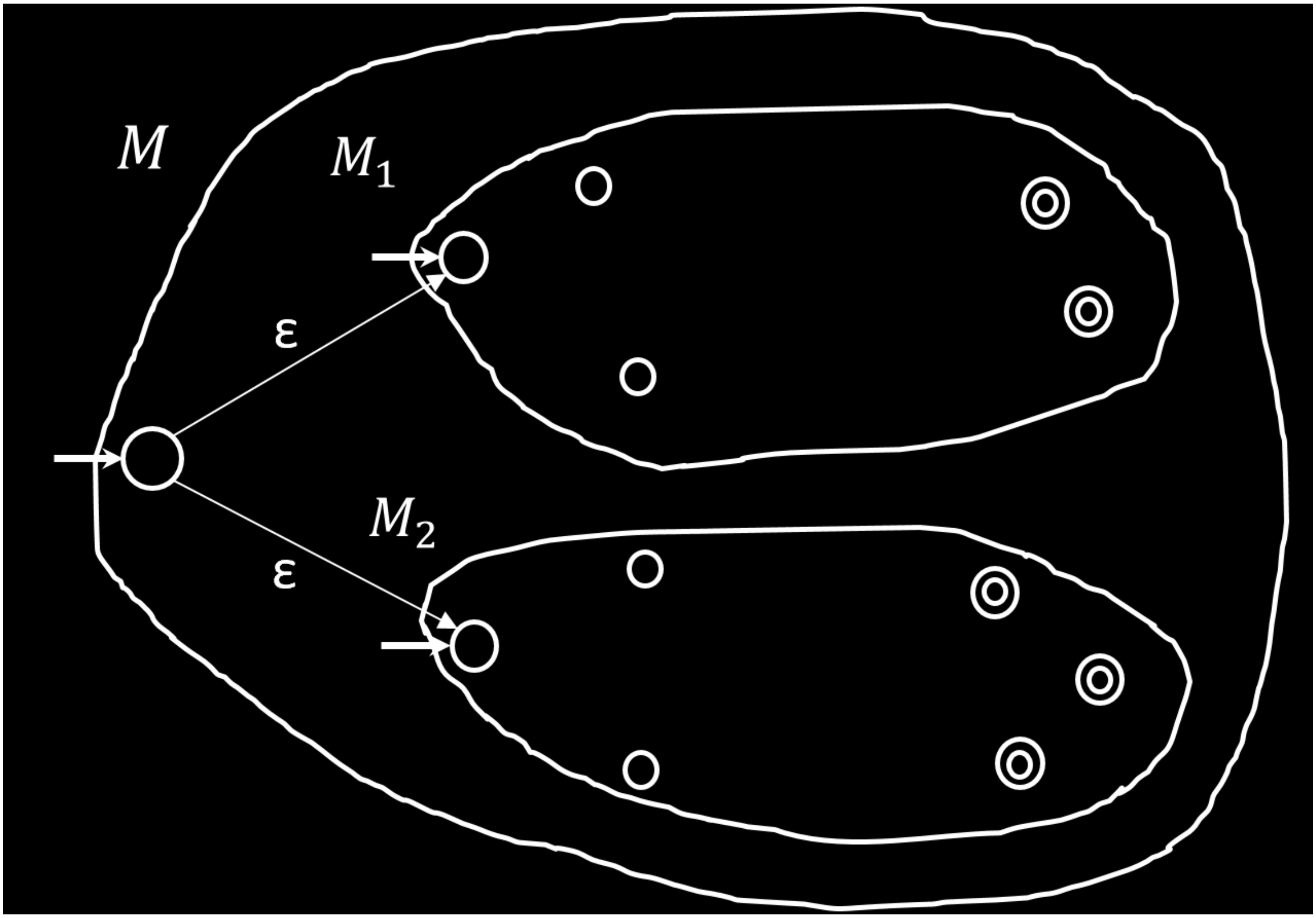
Proving Closure Properties with NFAs

Proposition: Closure under Union

If A_1, A_2 are regular languages, so is $A_1 \cup A_2$

Proof:
We wish to construct an NFA M that accepts both A_1 and A_2

We just construct a new NFA such that it has a single starting state that has two outgoing ε transitions to start state of M_1 and M_2.

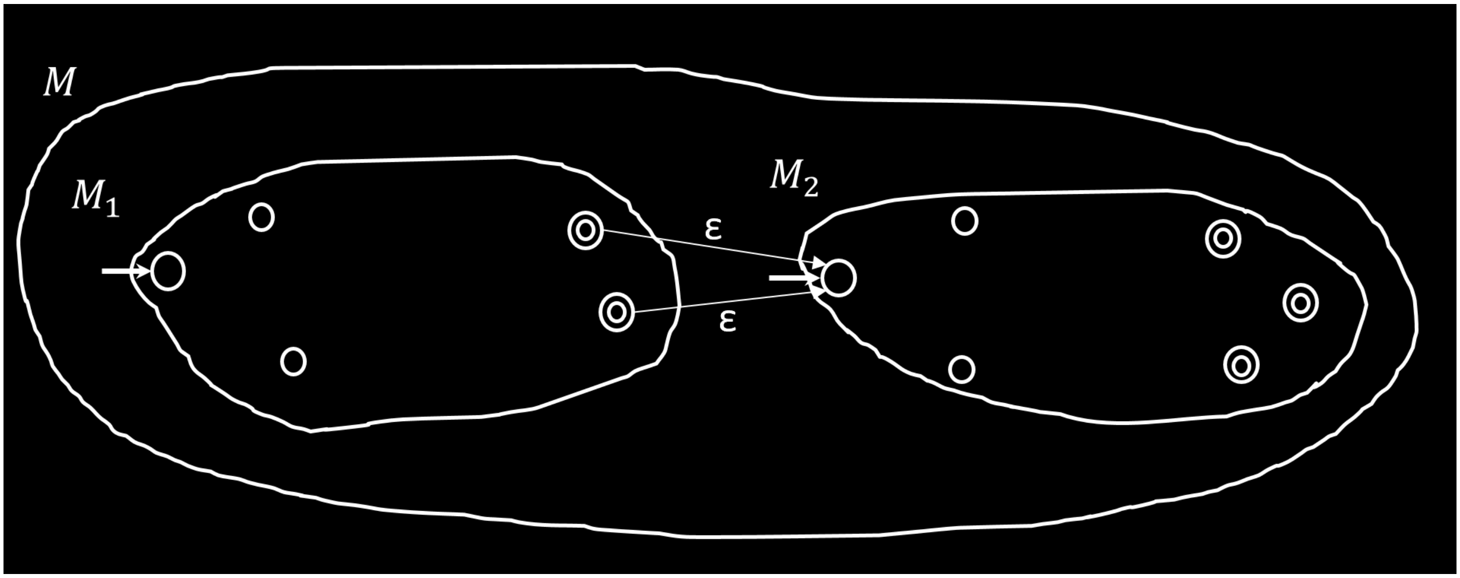


Proposition: Closure under Concatenation

If A_1, A_2 are regular languages, so is $A_1 \circ A_2$

Proof:
We wish to construct an NFA M that accepts all inputs $w = xy$ where $x \in A_1$ and $y \in A_2$

We include M_1, and we make all accept states of M_1 no longer accept. Then we add transitions with empty string from all of those to the start state of M_2.



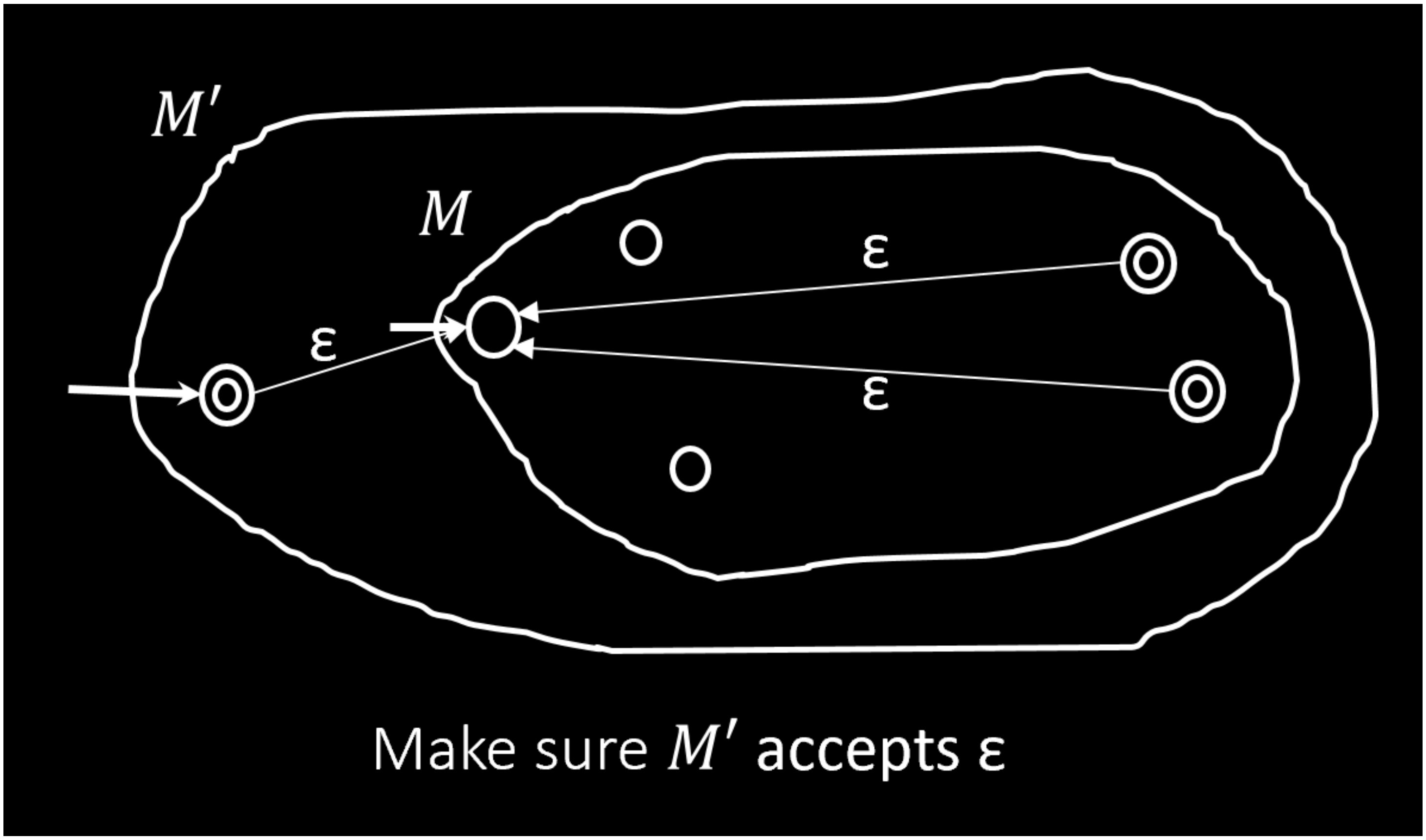
Proposition: Closure under Klein Star

If A is a regular language, so is A^*

Proof:
We wish to construct an NFA M' from M such that $A = L(M)$ and $A^* = L(M')$

We create a start state that leads to the start of M with a ε transition (this ensures that M' accepts ε)

- All accept states of M remain accept states but they all have a ε transition to the start of M

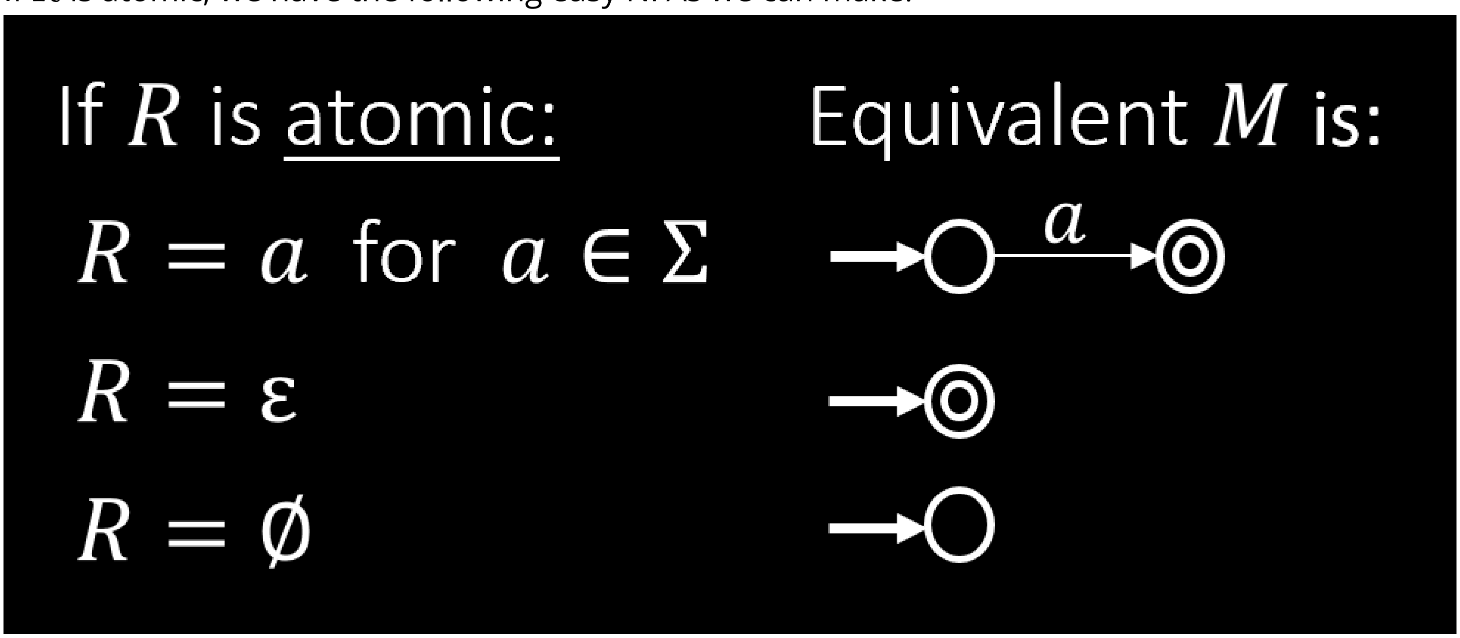


Proposition: Regular Expressions can be Converted to NFAs

Given a regular expression R, we can make a corresponding NFA with the same langauge as R

Proof:

If R is atomic, we have the following easy NFAs we can make:



- Σ could be considered as a single symbol, but it can also be done through composition of atomic expressions so it is fine to not include it

All compositions of atomic expressions with $\cup, \circ, *$ can all be made using the closure properties.

> **Goal**
>
> We've done (the easier) half of showing that regular expressions are equivalent to FAs

To prove that a DFA can be converted to a regular expression, we need another concept first: Generalized Nondeterministic Finite Automata