A regular language is a language with some finite automate that recognizes it

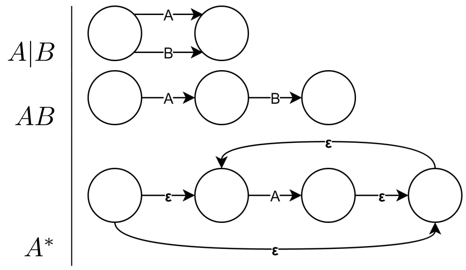
An *NFA* is represented formally by a 5-tuple, $(Q, \Sigma, \Delta, q_0, F)$, consisting of

- a finite set of states Q
- a finite set of input symbols Σ
- a transition function $\Delta : Q \times \Sigma \rightarrow P(Q)$.
- an *initial* (or *start*) state $q_0 \in Q$
- a set of states F distinguished as *accepting* (or *final*) *states* $F \subseteq Q$.

1 Language $\rightarrow$ NFA

A deterministic finite automaton M is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of

- a finite set of states (Q)
- a finite set of input symbols called the alphabet (Σ)
- a transition function $(\delta : Q \times \Sigma \rightarrow Q)$
- an initial or start state $(q_0 \in Q)$
- a set of accept states $(F \subseteq Q)$



2 NFA $\rightarrow$ DFA

1. Create chart of single moves and ϵ^*

	a	b	ϵ^*
1			
2			
3			

2. Create chart of single move followed by ϵ^*

	$a\epsilon^*$	$b\epsilon^*$
1		
2		
3		

3. Draw DFA from second table

4. Add additional moves and trash states to make complete

3 DFA $\rightarrow$ Regular Expression

1. Create new start state

2. Add arrow to each existing state

3. Create new accept state

4. Add arrow from each existing state

5. Find triangle of connections and eliminate states

Assume that A is a regular language. S = ap3
By the pumping lemma, S can be decompose into xyz such that
xyi
xy^iz □ A
|y| > 0
|xy| ≤ P

NFA to GNFA is the elimination of each state for regular expressions