DFA $\equiv$ NFA
Deterministic Finite Automata

Computation with finite memory
Non-Deterministic Finite Automata

Computation with finite memory

and “verified guessing”
From NFAs to DFAs

Input: NFA \( N = (Q, \Sigma, \delta, Q_0, F) \)

Output: DFA \( M = (Q', \Sigma, \delta', q_0', F') \)

To learn if an NFA accepts, we could do the computation in parallel, maintaining the set of all possible states that can be reached.

Idea:
Set \( Q' = 2^Q \)
From NFAs to DFAs: **Subset Construction**

**Input:** NFA \( N = (Q, \Sigma, \delta, Q_0, F) \)

**Output:** DFA \( M = (Q', \Sigma, \delta', q_0', F') \)

\[
Q' = 2^Q \\
\delta' : Q' \times \Sigma \rightarrow Q' \\
\delta'(R, \sigma) = \bigcup_{r \in R} \varepsilon(\delta(r, \sigma)) \\
q_0' = \varepsilon(Q_0) \\
F' = \{ R \in Q' | \exists f \in R \text{ for some } f \in F \}
\]

*For \( S \subseteq Q\), the \( \varepsilon \)-closure of \( S \) is*

\[
\varepsilon(S) = \{ q | q \text{ reachable from some } s \in S \text{ by taking 0 or more } \varepsilon \text{ transitions} \}
\]
Example of the $\varepsilon$-closure

\[ \varepsilon(\{q_0\}) = \{q_0, q_1, q_2\} \]
\[ \varepsilon(\{q_1\}) = \{q_1, q_2\} \]
\[ \varepsilon(\{q_2\}) = \{q_2\} \]
Given: NFA $N = (\{1,2,3\}, \{a,b\}, \delta, \{1\}, \{1\})$

Construct: Equivalent DFA $M = (2^{\{1,2,3\}}, \{a,b\}, \delta', \{1,3\}, \ldots)$

$\varepsilon(\{1\}) = \{1,3\}$
Parting thought:
NFA=DFA, does it mean that non-determinism is free for Finite Automata?