NFA
CSCI 338
NFA Processing

1. If a “decision” is encountered, split and take all options.
2. If input symbol does not match any outgoing transitions, that branch dies.
3. If any branch ends in an accept state, accept. If not, reject.
4. If $\varepsilon$ is encountered, split and take all options without consuming a character from string.

$\omega = 101$
NFA Formal Definition

NFAs consist of:

1. Finite set of states, \( Q \).
2. Finite alphabet, \( \Sigma \).
3. Transition function, \( \delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q) \).
4. Start state, \( q_0 \in Q \).
5. Set of accept states, \( F \subseteq Q \).

I.e. set of all subsets.
E.g. \( Q = \{q_1, q_2\} \)
\[ \mathcal{P}(Q) = \{\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}\} \]
I.e. \( \exists 0 \) or more transitions for each \( e \in \Sigma \cup \{\varepsilon\} \) at each state.
What is the NFA that accepts \( \{ \omega: \omega \text{ starts with } 1 \text{ and } \omega \text{ ends with } 0 \} \)?

Only \( \omega \)'s that start with 1 get to \( q_2 \). Any string that gets to \( q_2 \), can get to \( q_3 \) and terminate, if it ends with 0.
Build an NFA for the following language:
{11}.
Build an NFA for the following language: \{11\}.
Build an NFA for the following language:
\[\{11\}. \{\omega: \omega \text{ could be anything except } 11\}.\]
Build an NFA for the following language:
\{11\}. \{\omega: \omega \text{ could be anything except } 11\}.
Build an NFA for the following language:
\(\{11\}\). \(\{\omega: \omega \text{ could be anything except } 11\}\).
Build an NFA for the following language: 
\{\omega: \omega \text{ could be anything except } 11\}.

Swapping accept/reject states in an NFA does NOT yield the complement language.
Build an NFA for the following language: 
\{ \varepsilon \}. 

![Diagram showing an NFA with transitions labeled 0,1 and \varepsilon transitions.]

Build an NFA for the following language:
\{\varepsilon\}.
Build an NFA for the following language:
\[ \{ \omega : \omega \text{ contains an even number of 0's} \} \]
Build an NFA for the following language:
\[ \{ \omega : \omega \text{ contains exactly two 1's} \} \].
Build an NFA for the following language:
\( \{ \omega : \omega \) contains an even number of 0’s \textbf{or} exactly two 1’s\}. 

![Diagram of NFA](image-url)
Build an NFA for the following language:
\[ \{ \omega : \omega \text{ contains an even number of 0's or exactly two 1's} \} \]
Build an NFA with three states for:
{\( \omega : \omega \) has the form \( 0^*1^*0^+ \).}

Proof:

Additional string notation:
- \( 0^* \): Zero or more 0s (e.g. 0, 0000, \( \varepsilon \))
- \( 0^+ \): One or more 0s (e.g. 0, 0000)
Make an NFA with three states for:
\{\omega: \omega \text{ has the form } 0^*1^*0^+.\}

Proof:

Additional string notation:
- $0^*$: Zero or more 0s (e.g. 0, 0000, $\varepsilon$)
- $0^+$: One or more 0s (e.g. 0, 0000)