Nondeterminism
CSCI 338
DFA vs NFA

**Deterministic Finite Automaton (DFA):**

**Nondeterministic Finite Automaton (NFA):**
DFA vs NFA

Deterministic Finite Automaton (DFA):
• Every state has exactly one transition for every character ($e \in \Sigma$).

Nondeterministic Finite Automaton (NFA):
DFA vs NFA

Deterministic Finite Automaton (DFA):
• Every state has exactly one transition for every character \((e \in \Sigma)\).

Nondeterministic Finite Automaton (NFA):
• Allowed to have multiple (or 0) transitions for each \(e \in \Sigma\).
DFA vs NFA

Deterministic Finite Automaton (DFA):
• Every state has exactly one transition for every character \((e \in \Sigma)\).

Nondeterministic Finite Automaton (NFA):
• Allowed to have multiple (or 0) transitions for each \(e \in \Sigma\).
• Allowed to have transitions that happen without input (\(\varepsilon\)-transition).
Deterministic Finite Automaton (DFA):
• Every state has exactly one transition for every character \((e \in \Sigma)\).
• For the same input, everyone takes the same path to the same final state.

Nondeterministic Finite Automaton (NFA):
• Allowed to have multiple (or 0) transitions for each \(e \in \Sigma\).
• Allowed to have transitions that happen without input (\(\varepsilon\)-transition).
DFA vs NFA

Deterministic Finite Automaton (DFA):

- Every state has exactly one transition for every character \( e \in \Sigma \).
- For the same input, everyone takes the same path to the same final state.

Nondeterministic Finite Automaton (NFA):

- Allowed to have multiple (or 0) transitions for each \( e \in \Sigma \).
- Allowed to have transitions that happen without input (\( \varepsilon \)-transition).
- Processing strings is different.
DFA Processing

\{\omega: \omega \text{ contains the substring 110}\}
DFA Processing

\[ \omega \text{ contains the substring 110} \]

\[ \omega = 1110 \]
DFA Processing

\[ \omega = 1110 \]

\{ \omega: \omega \text{ contains the substring } 110 \}
DFA Processing

\[ \omega : \omega \text{ contains the substring } 110 \]

\[ \omega = 1110 \]
DFA Processing

\[ \omega = 1110 \]

\{ \omega : \omega \text{ contains the substring } 110 \}
DFA Processing

\{ \omega: \omega \text{ contains the substring } 110 \} \quad \omega = 1110
DFA Processing

\[ \omega \text{ contains the substring } 110 \]

\[ \omega = 1110 \]
NFA Processing

\{\omega: \omega \text{ contains the substring } 110\}
NFA Processing

Multiple transition options.

No transition options.

$\{\omega: \omega \text{ contains the substring 110}\}$
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.

\[ \{ \omega : \omega \text{ contains the substring } 110 \} \]
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.

$\omega = 1110$
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.

\[ \omega = 1110 \]

{\( \omega \): \( \omega \) contains the substring 110}
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.

\[ \omega : \omega \text{ contains the substring } 110 \]
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.

\[ \{ \omega : \omega \text{ contains the substring } 110 \} \]

\[ \omega = 1110 \]
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.

\( \omega \): \( \omega \) contains the substring 110

\( \omega = 1110 \)
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.

\{ \omega : \omega \text{ contains the substring } 110 \}
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.

Alternate approach: If it is possible to end on an accept state, accept.

\( \omega: \omega \) contains the substring 110

\( \omega = 1110 \)
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.

\{\omega: \omega \text{ contains the substring 110}\}

\omega = 101
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.

{\omega: \omega \text{ contains the substring 110}}

\omega = 101
$\epsilon$-Transitions

$\epsilon$-transitions can be taken without consuming any characters from the string being processed.
\( \varepsilon \)-Transitions

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.
ε-Transitions

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 ε is encountered, split and take all options without consuming a character from string.

ω = 101
**ε-Transitions**

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 ε is encountered, split and take all options without consuming a character from string.

ω = 101
**ε-Transitions**

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 ε is encountered, split and take all options without consuming a character from string.

\[ \omega = 101 \]
\(\varepsilon\)-Transitions

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.

\[
\omega = 101
\]
\( \varepsilon \)-Transitions

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.

\[
\omega = 101
\]
ε-Transitions

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 ε is encountered, split and take all options without consuming a character from string.

ω = 101
**ε-Transitions**

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.

\[ \omega = 101 \]
\(\varepsilon\)-Transitions

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.

\[1, \varepsilon\]

\[1, 0, 1, \varepsilon\]

\[\omega = 101\]
**ε-Transitions**

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 ε is encountered, split and take all options without consuming a character from string.

ω = 101
NFA Formal Definition

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

![Graph](diagram.png)
NFA Formal Definition

NFAs consist of:

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

Power set of \( Q \).
I.e. set of all subsets.
E.g. \( Q = \{q_1, q_2\} \) => \( \mathcal{P}(Q) = \{\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}\} \)
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, $F \subseteq Q$.

Power set of $Q$.
I.e. set of all subsets.
E.g. $Q = \{q_1, q_2\}$
\[ \Rightarrow \mathcal{P}(Q) = \{\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}\} \]
I.e. $\exists$ 0 or more transitions for each $e \in \Sigma \cup \{\varepsilon\}$ at each state
NFA Practice

What is the NFA that accepts \( \{ \omega: \omega \text{ starts with } 1 \text{ and ends with } 0 \} \)?
What is the NFA that accepts \( \{ \omega: \omega \text{ starts with 1 and ends with 0} \} \)?
What is the NFA that accepts \( \{ \omega : \omega \text{ starts with } 1 \text{ and ends with } 0 \} \)?

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.