04-0: Non-Determinism

- A Deterministic Finite Automata’s transition function has exactly one transition for each state/symbol pair.
- A Non-Deterministic Finite Automata can have 0, 1 or more transitions for a single state/symbol pair.
- Example: \( L = \{ w \in \{a, b\} : w \text{ starts with } a \} \)
  - Regular expression?
Example: \( L = \{ w \in \{a, b\} : w \text{ starts with } a \} \)

- \( a(a+b)^* \)
Example: \( L = \{ w \in \{a, b\} : w \text{ starts with } a \} \)

- \( a(a+b)^* \)

What happens if a 'b' is seen in state \( q_0 \)?

- The machine “crashes”, and does not accept the string.
Example: \( L = \{w \in \{a, b\} : w \text{ contains the substring } aa\} \)

- Regular Expression?
**04-4: NFA Example**

- Example: $L = \{ w \in \{ a, b \} : w \text{ contains the substring } aa \}$
  - $(a+b)^*aa(a+b)^*$

What happens if a $a$ is seen in state $q_0$?
Example: $L = \{ w \in \{a, b\} : w \text{ contains the substring } aa \}$
- $(a+b)^*aa(a+b)^*$

What happens if a $a$ is seen in state $q_0$?
- Stay in state $q_0$, or go on to state $q_1$
- Multiple Computational Paths (board example)
Example: $L = \{ w \in \{a, b\} : w \text{ contains the substring } aa \}$

Does this machine accept $abaa$?
If there is any computational path that accepts a string, then the machine accepts the string.

Two ways to think about NFAs:
- Magic “Oracle”, which always picks the correct path to take
- Try all possible paths
Example: $L = \{w \in \{a, b\} : w \text{ contains the substring } aa\}$

If a string contains $aa$, will there be a computational path that accepts it?

If a string does not contain $aa$, will there be a computational path that accepts it?
04-9: NFA Definition

- Difference between a DFA and an NFA
  - DFA has exactly only transition for each state/symbol pair
    - \( \delta : (K \times \Sigma) \rightarrow K \)
  - NFA has 0, 1 or more transitions for each state/symbol pair
04-10: NFA Definition

- Difference between a DFA and an NFA
  - DFA has exactly only transition for each state/symbol pair
    - Transition function: \( \delta : (K \times \Sigma) \mapsto K \)
  - NFA has 0, 1 or more transitions for each state/symbol pair
    - Transition relation: \( \Delta \subseteq ((K \times \Sigma) \times K) \)
04-11: **NFA Definition**

- A NFA is a 5-tuple $M = (K, \Sigma, \Delta, s, F)$
  - $K$ Set of states
  - $\Sigma$ Alphabet
  - $\Delta : (K \times \Sigma) \times K$ is a Transition relation
  - $s \in K$ Initial state
  - $F \subseteq K$ Final states
Create an NFA for:

- All strings over \{a, b\} that start with a and end with b
Create an NFA for:

- All strings over \{a, b\} that start with a and end with b

(example computational paths for ababb, abba, bbab)
Create an NFA for:

- All strings over \{0, 1\} that contain 0110 or 1001
Create an NFA for:

- All strings over \{0, 1\} that contain 0110 or 1001
04-16: $\varepsilon$-Transitions

- $\varepsilon$ transition consumes no input
- NFA (with $\varepsilon$ transitions) for $(ab)^*(aab)^*$
Create an NFA (with $\epsilon$-transitions) for:

- All strings over \{a, b, c\} that are missing at least one letter. For example, aabba, cbbc, ccacc $\in L$, while abbc $\notin L$
Create an NFA (with $\epsilon$-transitions) for:

- All strings over \{a, b, c\} that are missing at least one letter. For example, aabba, cbbc, ccacc $\in L$, while abbc $\not\in L$

```
abcb, bbab, abbab, abc
```
Yet More Formalism

- $\epsilon$-closure
  - $\epsilon$-closure$(q)$ = set of all states that can be reached following zero or more $\epsilon$-transitions from state $q$. 
04-20: $\varepsilon$-closure

- $\varepsilon$-closure examples

(quick review: What is $L[M]$?)
04-21: $\varepsilon$-closure

- $\varepsilon$-closure examples

$\varepsilon$-closure example:

$L[M] = \{a, aa, ba, ca, aba\}$
04-22: \( \epsilon \)-closure

- \( \epsilon \)-closure examples

\[
\begin{align*}
\epsilon\text{-closure}(q_0) &= \\
\epsilon\text{-closure}(q_3) &= \\
\epsilon\text{-closure}(q_2) &= \\
\epsilon\text{-closure}(q_5) &= 
\end{align*}
\]
• $\epsilon$-closure examples

- $\epsilon$-closure($q_0$) = \{ $q_0$, $q_1$, $q_4$, $q_5$ \}
- $\epsilon$-closure($q_3$) = \{ $q_3$, $q_4$, $q_5$ \}
- $\epsilon$-closure($q_2$) = \{ $q_2$, $q_6$ \}
- $\epsilon$-closure($q_5$) = \{ $q_5$ \}
04-24: $\epsilon$-closure – Sets

- $\epsilon$-closure
  - $\epsilon$-closure$(q) = \text{set of all states that can be reached following zero or more } \epsilon \text{-transitions from state } q$.
- Can extend $\epsilon$-closure to a set of states
  - $\epsilon$-closure$(\mathcal{S}) = \bigcup \{ A : q \in \mathcal{S} \land \epsilon$-closure$(q) = A \}$
A NFA is a 5-tuple $M = (K, \Sigma, \Delta, s, F)$

- $K$ Set of states
- $\Sigma$ Alphabet
- $\Delta : (K \times (\Sigma \cup \{\epsilon\})) \times K$ is a Transition relation
- $s \in K$ Initial state
- $F \subseteq K$ Final states
Binary relation $\vdash_M$: What machine $M$ yields in one step

- $\vdash_M \subseteq (K \times \Sigma^*) \times (K \times \Sigma^*)$
- $\vdash_M = \{(((q_1, aw), (q_2, w)) : q_1, q_2 \in K, w \in \Sigma^*, a \in \Sigma \cup \{\epsilon\}, ((q_1, a), q_2) \in \Delta_M\}$

Binary relation $\vdash^*_M$: Transitive, reflexive closure of $\vdash_M$
04-27: NFA Languages

- $L[M]$ – Language defined by NFA $M$
  - $L[M] = \{w : (s_M, w) \vdash_M^* (f, \epsilon) \text{ for some } f \in F_M^*\}$
- $L_{NFA} = \{L : \exists \text{NFA } M, L[M] = L\}$
04-28: NFA Examples

- Create an NFA for the language
- Give an NFA for the language $L = \text{All strings over } \{0,1\} \text{ that contain two pairs of adjacent 0’s separated by an even number of symbols. So, 0100110011, 01100101100101, and 01001000 are in the language, but 0100100, 1011001, and 0111011 are not in the language.}$
Create an NFA for the language

Give an NFA for the language $L =$ All strings over $\{0,1\}$ that contain two pairs of adjacent 0’s separated by an even number of symbols. So, 0100110011, 01100101100101, and 01001000 are in the language, but 0100100, 1011001, and 0111011 are not in the language.
Create an NFA for the language

$L = \text{All strings over\ } \{a, b\} \text{ that have an } a \text{ as one of the last 3 characters in the string. So, } a, \text{ baab, bbbab, aabbaaabb } \in L, \text{ but } bb, \text{ baabbb, bbabbbbb } \notin L$
Create an NFA for the language

$L = \text{All strings over \{a,b\} that have an } a \text{ as one of the last 3 characters in the string. So, } a, \text{ baab, bbbab, aabbaaabb } \in L, \text{ but } bb, \text{ baabbb, bbabbbbb } \notin L$