Last time:
- DFAs and NFAs
- Operations on languages

Today:
- Nondeterminism
- Equivalence of NFAs and DFAs
- Closure properties of regular languages
Nondeterministic Finite Automaton (NFA) accepts a string $w$ if there is a way to make it reach an accept state on input $w$. 
Formal Definition

• An **NFA** is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$
  - $Q$ is the set of states
  - $\Sigma$ is the alphabet
  - $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the transition function
  - $q_0 \in Q$ is the start state
  - $F \subseteq Q$ is the set of accept states

• $P(Q)$ is the set of subsets of $Q$ and $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

• $M$ **accepts** a string $w$ if there is a path from $q_0$ to an accept state that $w$ follows.
Example

\[ N = (Q, \Sigma, \delta, q_0, F) \]

\[ Q = \{q_0, q_1, q_2, q_3\} \]
\[ \Sigma = \{0,1\} \]
\[ F = \{q_3\} \]
\[ \delta(q_0,0) = \{q_0\} \]
\[ \delta(q_0,1) = \{q_0, q_1\} \]
\[ \delta(q_1,\varepsilon) = \{q_1, q_2\} \]
\[ \delta(q_2,0) = \emptyset \]
Nondeterminism

Ways to think about nondeterminism
- parallel computation
- tree of possible computations
- guessing and verifying the “right” choice

Deterministic Computation

Nondeterministic Computation

accept or reject

accept

reject

9/12/2017 Sofya Raskhodnikova; based on slides by Nick Hopper
A DFA that recognizes the language \{1\}: 0

An NFA that recognizes the language \{1\}:
Theorem. Every DFA for language \{1\} must have at least 3 states.

Proof:
Theorem. Every NFA has an equivalent DFA.

Corollary: A language is regular iff it is recognized by an NFA.
NFA to DFA Conversion

Input: \( N = (Q, \Sigma, \delta, q_0, F) \)

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

Intuition: Do the computation in parallel, maintaining the set of states where all threads are.

Idea: \( Q' = P(Q) \)
NFA to DFA Conversion

Input: \( N = (Q, \Sigma, \delta, q_0, F) \)

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

\[
Q' = \mathcal{P}(Q) \\
\delta' : Q' \times \Sigma \to Q' \\
\delta'(R, \sigma) = \text{for all } R \subseteq Q \text{ and } \sigma \in \Sigma. \\
q_0' = \text{ } \\
F' = \text{ }
\]
Example: NFA to DFA

1)

a \rightarrow 1 \rightarrow b
Examples NFA to DFA

2)

1 → 2
\(\varepsilon\)  0,1
2 → 3
0
1

3
### NFA to DFA Conversion

**Input:** \(N = (Q, \Sigma, \delta, q_0, F)\)

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

For \(R \subseteq Q\), let \(E(R)\) be the set of states reachable by \(\varepsilon\)-transitions from the states in \(R\).

\[
Q' = P(Q)
\]

\[
\delta' : Q' \times \Sigma \rightarrow Q'
\]

\[
\delta'(R, \sigma) = \bigcup_{r \in R} (\delta(r, \sigma)) \quad \text{for all } R \subseteq Q\text{ and } \sigma \in \Sigma.
\]

\[
q_0' = (\{q_0\})
\]

\[
F' = \{ R \in Q' \mid R \text{ contains some accept state of } N\}
\]
Complement: $\overline{A} = \{ w \mid w \notin A \}$

Union: $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$

Intersection: $A \cap B = \{ w \mid w \in A \text{ and } w \in B \}$

Reverse: $A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A \}$

Concatenation: $A \circ B = \{ vw \mid v \in A \text{ and } w \in B \}$

Star: $A^* = \{ w_1 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$
THEOREM. The class of regular languages is **closed** under all 6 operations.

If A and B are regular, applying any of these operation yields a regular language.
Palindromes

A **palindrome** is a word or a phrase that reads the same forward and backward.

**Examples**

- mom
- madam
- Never odd or even.
- Stressed? No tips? Spit on desserts!
Let $L$ be the set of words in English. Then $L \cap L^R$ is

A. The set of English words in alphabetical order, followed by the same words in reverse alphabetical order.

B. $\{w \mid w \text{ is an English word or an English word written backwards}\}$.

C. $\{w \mid w \text{ is an English word that is a palindrome}\}$.

D. None of the above.
Closure under reverse

**Theorem.** The reverse of a regular language is also regular.

**Proof:** Let $L$ be a regular language and $M$ be a DFA that recognizes it. Construct an NFA $M'$ recognizing $L^R$:

- Define $M'$ as $M$ with the arrows reversed.
- Make the start state of $M$ be the accept state in $M'$.
- Make a new start state that goes to all accept states of $M$ by $\varepsilon$-transitions.
New construction for $A \cup B$

Construct an NFA $M$:

- $L(M_A) = A$
- $L(M_B) = B$

$\epsilon$ transitions allow moving between states without consuming an input symbol.
Concatenation: $A \circ B = \{ vw \mid v \in A \text{ and } w \in B \}$

**Theorem.** If $A$ and $B$ are regular, $A \circ B$ is also regular.

**Proof:** Given DFAs $M_1$ and $M_2$, construct NFA by connecting all accept states in $M_1$ to the start state in $M_2$. 

$L(M_1) = A$

$L(M_2) = B$
Concatenation operation

Concatenation: \( A \circ B = \{ vw \mid v \in A \text{ and } w \in B \} \)

**Theorem.** If \( A \) and \( B \) are regular, \( A \circ B \) is also regular.

**Proof:** Given DFAs \( M_1 \) and \( M_2 \), construct NFA by connecting all accept states in \( M_1 \) to the start state in \( M_2 \).

- Make all states in \( M_1 \) non-accepting.

Diagram: [Diagram showing NFA connections]
Star operation

Star: \( A^* = \{ w_1 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \} \)

**Theorem.** If \( A \) is regular, \( A^* \) is also regular.
The class of regular languages is closed under

Regular operations

Union: $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$

Concatenation: $A \circ B = \{ vw \mid v \in A \text{ and } w \in B \}$

Star: $A^* = \{ w_1 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$

Other operations

Complement: $\neg A = \{ w \mid w \notin A \}$

Intersection: $A \cap B = \{ w \mid w \in A \text{ and } w \in B \}$

Reverse: $A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A \}$