Last time:
• Equivalence of NFAs and DFAs

Today:
• Closure properties of regular languages
• Equivalence of NFAs, DFAs and regular expressions
Regular Operations on languages

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

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

$L(M_A) = A$

$L(M_B) = B$
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 $M_2$. 

![Diagram showing concatenation of languages $L(M_1)=A$ and $L(M_2)=B$]
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.

\[ L(M_1) = A \quad L(M_2) = B \]
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: \( \overline{A} = \{ w \mid w \notin A \} \)

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

Reverse: \( A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A \} \)
Regular expressions

- In a regular expression, we can use
  - **Constants:** ε, Φ, a set Σ, members of Σ.
  - **Regular operations:** *, ∪, ∩

- **Examples:**
  - 0*1* = \{ w | w has a run of 0s followed by a run of 1s\}
  - (0∪1)* = the set of all strings over the alphabet Σ={0,1}
  - 0*1*(ε ∪ Φ)

- **L(R)** = the language regular expression R describes
Precedence

EXAMPLE

\[ R_1 \ast R_2 \cup R_3 = ((R_1 \ast) R_2) \cup R_3 \]
Regular expressions: examples

1) \{ w \mid w \text{ has exactly one character } 1 \} \\
\qquad 0^*10^* \\

2) \{ w \mid w \text{ has length } \geq 3 \text{ and its 3rd symbol is } 0 \} \\
\qquad (0 \cup 1)(0 \cup 1) \ 0 \ (0 \cup 1)^* \\

3) \{ w \mid \text{every odd position of } w \text{ is a } 1 \} \\
\qquad (1(0 \cup 1))^* \ (\varepsilon \cup 1)
Theorem. Every regular expression has an equivalent NFA.

Proof: Induction on the length of regular expression $R$.

Base case: length 1

- $R = \varepsilon$
  
- $R = \emptyset$
  
- $R = \sigma$

Inductive step: follows from closure of the class of regular languages under the regular operations.
What should the induction hypothesis be?

A. Suppose some regular expression of length $k$ can be converted an NFA, for some $k \in \mathbb{N}$.

B. Suppose all regular expressions of length $k$ can be converted an NFA, for some $k \in \mathbb{N}$.

C. Suppose all regular expressions of length at most $k$ can be converted an NFA, for some $k \in \mathbb{N}$.

D. None of the above.
Regular expression to NFA

Transform \((1(0 \cup 1))^*\) to an NFA

Diagram:

- Start state
- Transition on \(\epsilon\) to state 1
- Transition on 1 to state 2
- Transition on 1,0 to states
- Transition on \(\epsilon\) from state 2 to start state

Sofya Raskhodnikova; based on slides by Nick Hopper
Theorem. Every NFA has an equivalent regular expression.

Proof idea:
Transform NFA to a regular expression by removing states and relabeling the arrows with regular expressions.
Generalized NFAs

• Each transition is labeled with a regular expression
• Unique and distinct start and accept states
• No transitions to the start state
• No transitions from the accept state
Generalized NFAs

G accepts w if it finds \( q_0 q_1 \ldots q_k, w_1 \ldots w_k \):

- \( w_i \) is generated by \( R(q_i, q_{i+1}) \)
- \( R(q_s, q) = a^*b \)
- \( R(q, q) = \emptyset \)
- \( w = w_1 w_2 \ldots w_k \)
- \( R(q_0, q_k) \subseteq q, \emptyset_q \)
- \( q_k = q_a \)