What we’ll do today

• Prove that DFAs and NFAs are equally expressive
• Use that to prove closure of other regular operations
• Introduction to regular expressions
Recap: Closure

• If you perform an operation on one/more regular languages, is the result also a regular language?

Operations on Regular Languages

Given: Two regular languages A and B

✓ Union: \( A \cup B = \{w \mid w \in A \text{ or } w \in B\} \)

✓ Intersection: \( A \cap B = ? \)

✓ Complementation: \( \overline{A} = \{w \mid w \not\in A\} \)

• Reverse: \( A^R = \{w_1w_2\ldots w_k \mid w_kw_{k-1}\ldots w_1 \in A\} \)

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

• Star:
  \( A^* = \{w_1w_2\ldots w_k \mid k \geq 0 \text{ and each } w_i \in A\} \)
Closure under Reverse

- **Reverse**: \( A^R = \{ w_1 w_2 \ldots w_k \mid w_k w_{k-1} \ldots w_1 \in A \} \)

- Regular languages are closed under reverse. Here’s an attempt to prove it:
  - Given \( M \) that recognizes \( A \)
  - What if you could “run it backwards”?
  - Construct \( M^R \) as \( M \) with all arrows reversed & accept state interchanged with start state
  - \( M^R \) is an NFA

---

A non-deterministic finite automaton (NFA) is also a 5-tuple \( M = (Q, \Sigma, \delta, q_0, F) \)

- \( Q \) is the set of states
- \( \Sigma \) is the alphabet
- \( \delta : Q \times \Sigma \varepsilon \to 2^Q \) is the transition function
- \( q_0 \in Q \) is the start state
- \( F \subseteq Q \) is the set of accept states
- \( 2^Q \) is the set of subsets of \( Q \) and \( \Sigma \varepsilon = \Sigma \cup \{ \varepsilon \} \)
Equivalence

- Two automata are equivalent if their languages are the same
  - For $M_1, M_2$, $L(M_1) = L(M_2)$
- DFAs and NFAs:
  - For every NFA there is an equivalent DFA (we'll prove this) and vice-versa (this is easy, why?)
**Theorem:** Every NFA has an equivalent DFA

Corollary: A language is regular iff it is recognized by an NFA

Corollary: L is regular iff $L^R$ is regular (need to also prove that $M^R$ recognizes $L^R$)

---

From NFA to DFA

- **Proof Hints:**
  - **Proof by construction**
    Given an arbitrary NFA $N$, construct an equivalent DFA $M$
  - **Proof by induction**
    $N$ accepts a word $w$ iff $M$ accepts $w$
FROM NFA TO DFA

Input: \( N = (Q, \Sigma, \delta, q_0, F) \)
Output: \( M = (Q', \Sigma, \delta', q'_0, F') \)

\[ Q' = ? \]

From NFA to DFA

Input: \( N = (Q, \Sigma, \delta, q_0, F) \)
Output: \( M = (Q', \Sigma, \delta', q'_0, F') \)

Idea:

\[ Q' = 2^Q \]

Assume (for now) that there are no \( \epsilon \)-transitions

Each non-stuck path in the computation tree is of equal length

Do a BFS (breadth-first search) on this tree, tracking the “set of states” transitioned to
NFA Example

Run on 1110

From NFA to DFA

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

Idea: \( Q' = 2^Q \)

What if we had \( \varepsilon \)-transitions?
From NFA to DFA

Input: \( N = (Q, \Sigma, \delta, q_0, F) \)
Output: \( M = (Q', \Sigma, \delta', q'_0, F') \)

Idea:
\( Q' = 2^Q \)

What if we had \( \epsilon \)-transitions?
After reading an input symbol, follow \( \epsilon \)-transitions until you can’t any more

Given a set \( S \) in \( 2^Q \), \( E(S) \) is the set of all states reached from \( S \) by following \( \epsilon \)-transitions
- \( E(S) \) is called the \( \epsilon \)-closure of \( S \)

NFA Example

Run on 1110
From NFA to DFA

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

$Q' = 2^Q$

$\delta' : Q' \times \Sigma \rightarrow Q'$

$\delta'(R, \sigma) = \bigcup_{r \in R} E(\delta(r, \sigma))$

$q_0' = ?$

$F' = ?$

(read details of the construction in Sipser)
From NFA N to DFA M

• Construction is complete
• But the proof isn’t: Need to prove
  \( N \) accepts a word \( w \) \iff \( M \) accepts \( w \)

• Use *structural induction* on the length of \( w \), \( |w| \)
  – Base case: \( |w| = 0 \)
  – Induction step: Assume for \( |w| = n \), prove for \( |w| = n+1 \)

Useful Definition

• Let \( w \in \Sigma^* \)
• For an NFA \( N \):  
  – \( \delta(q, w) \) = set of states reached by executing \( N \) on \( w \) starting from \( q \)
  – Note that a state of \( N \) is in \( Q \)
• For the corresponding DFA \( M \):  
  – \( \delta'(q', w) \) = state reached by executing \( M \) on \( w \) starting from \( q' \)
  – Note that a state of \( M \) is in \( 2^Q \)
NFA to DFA: Complexity

• If the original NFA N has n states, how large can the corresponding DFA M be?

– Answer: $2^n$ states
– Exercise: construct an example where N has n states and M has $\Theta(2^n)$ states
Remaining Operations

Given: Two regular languages $A$ and $B$

- **Concatenation:**
  \[ A \cdot B = \{ vw \mid v \in A \text{ and } w \in B \} \]

- **Star:**
  \[ A^* = \{w_1w_2\ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \} \]

Closure under Concatenation

Given DFAs $M_1$ and $M_2$, how can we construct an NFA $N$ for $L(M_1) \cdot L(M_2)$?
Closure under Concatenation

Given DFAs $M_1$ and $M_2$, construct NFA $N$ by connecting all accept states in $M_1$ to start states in $M_2$

- What are accept states of $N$?

Closure under Star

Let $L$ be a regular language and $M$ be a DFA for $L$

How do we construct an NFA $N$ that recognizes $L^*$?
Closure under Star

Why not the following?

Formally:

Input: DFA $M = (Q, \Sigma, \delta, q_1, F)$
Output: NFA $N = (Q', \Sigma, \delta', \{q_0\}, F')$

- $Q' = Q \cup \{q_0\}$
- $F' = F \cup \{q_0\}$

\[
\delta'(q,a) = \begin{cases} 
\{\delta(q,a)\} & \text{if } q \in Q \text{ and } a \neq \varepsilon \\
\{q_1\} & \text{if } q \in F \text{ and } a = \varepsilon \\
\{q_1\} & \text{if } q = q_0 \text{ and } a = \varepsilon \\
\emptyset & \text{if } q = q_0 \text{ and } a \neq \varepsilon \\
\emptyset & \text{else}
\end{cases}
\]
REGULAR LANGUAGES ARE CLOSED UNDER REGULAR OPERATIONS

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

Complementation: $\overline{A} = \{ w \mid w \notin A \}$

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

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

Regular Expressions
What does Language D look like?

D = \{ w | w has equal number of occurrences of 01 and 10\}

w should “toggle” between 0 and 1 an equal number of times

How about: 0, 1, 011, 0110, ε -- are they in D?

\begin{align*}
\Sigma &= \{0, 1\} \\
1 &\cup 0 &\cup &\varepsilon &\cup & (0\Sigma^*0) &\cup & (1\Sigma^*1)
\end{align*}
REGULAR EXPRESSIONS

σ is a regular expression representing \{σ\} 
( σ ∈ Σ )
ε is a regular expression representing \{ε\}
∅ is a regular expression representing ∅

If \( R_1 \) and \( R_2 \) are regular expressions representing \( L_1 \) and \( L_2 \) then:

\( (R_1 R_2) \) represents \( L_1 \cdot L_2 \)
\( (R_1 \cup R_2) \) represents \( L_1 \cup L_2 \)
\( (R_1)^* \) represents \( L_1^* \)

Next Steps

• Read Sipser 1.3 in preparation for next lecture