A **nondeterministic finite automaton** is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \(Q\) is a finite set of states,
2. \(\Sigma\) is a finite alphabet,
3. \(\delta: Q \times \Sigma \rightarrow P(Q)\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.

A **finite automaton** is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \(Q\) is a finite set called the states,
2. \(\Sigma\) is a finite set called the **alphabet**, 
3. \(\delta: Q \times \Sigma \rightarrow Q\) is the **transition function**, 
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.
Announcements

• HW 1 in
  • Due Sun 10/25 11:59pm EST

• HW 2 out
  • Due Sun 10/2 11:59pm EST

• Ask HW questions publicly on Piazza
  • So the entire class can participate and benefit from the discussion
  • (Make it anonymous if you want to)

• Recipe: Designing a machine = programming
  • Make examples to understand problem
  • States = what the machine needs to remember
  • Check design with tests
Flashback: Kinds of Mathematical Proof

Deductive Proof

• Start with known facts and statements
• Use logical inference rules to reach new conclusions
An (Important) Inference Rule: Modus Ponens

Premises
• If $P$ then $Q$
• $P$ is true

Conclusion
• $Q$ must also be true
Deductive Proof Example

Prove the following:

• If: If $x \geq 4$, then $2^x \geq x^2$
  
  Given

• And: $x$ is the sum of the squares of four positive integers

• Then: $2^x \geq x^2$
  
  Need to show this
Deductive Proof Example

Prove: If \( x \geq 4 \), then \( 2^x \geq x^2 \) and \( x \) is the sum of the squares of four positive integers, then \( 2^x \geq x^2 \).

Proof:

**Statement**

1. \( x = a^2 + b^2 + c^2 + d^2 \)
2. \( a \geq 1; b \geq 1; c \geq 1; d \geq 1 \)
3. \( a^2 \geq 1; b^2 \geq 1; c^2 \geq 1; d^2 \geq 1 \)
4. \( x \geq 4 \)
5. If \( x \geq 4 \), then \( 2^x \geq x^2 \)
6. \( 2^x \geq x^2 \)

**Justification**

1. Given
2. Given
3. By Step (1) & arithmetic laws
4. (1), (3), and arithmetic
5. Given
6. (4) and (5)
Deductive Proof Example: Regular Lang

**Prove:** The following language \( A = \{ \ldots \} \) is a regular language

**Proof:**

**Statement**

1. If a DFA recognizes a language, then that language is regular
2. DFA \( M = (Q, \Sigma, \delta, q_{\text{start}}, F) \) where \( Q = \ldots, \) etc., recognizes language \( A \)
3. \( A \) is a regular language

**Justification**

1. Definition of a regular language
2. Definition of a DFA and DFA computation rule
3. By Steps (1) and (2)
Deductive Proof Example: Closed Op?

Prove: The operation $OP = \{ \ldots \}$ is closed for regular languages

Proof:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ???</td>
<td>1. ???</td>
</tr>
</tbody>
</table>

- ???

- $OP$ is closed for regular languages

- ???
Last Time: Concatenation of Languages

Let the alphabet $\Sigma$ be the standard 26 letters \{a, b, \ldots, z\}.

If $A = \{\text{good, bad}\}$ and $B = \{\text{boy, girl}\}$, then

$$A \circ B = \{\text{goodboy, goodgirl, badboy, badgirl}\}$$
**Last Time:** Is Concatenation Closed?  

**THEOREM**  
The class of regular languages is closed under the concatenation operation.  

In other words, if $A_1$ and $A_2$ are regular languages then so is $A_1 \circ A_2$.  

- Cannot combine $A_1$ and $A_2$’s machine because:  
  - Don’t know when to switch? (can only read input once)  
- Need a **different machine**!  
- So concatenation **not closed** for regular langs?
DEFINITION

A **nondeterministic finite automaton** is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \(Q\) is a finite set of states,
2. \(\Sigma\) is a finite alphabet,
3. \(\delta: Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.

**NFA transition may not read input**, \(\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}\)

**Transition results in a set of states**
Last Time: NFA Extended Transition Function

Define **extended transition function**: \( \hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q) \)

**Domain:**
- Beginning state \( q \in Q \)
- Input string \( w = w_1w_2 \cdots w_n \) where \( w_i \in \Sigma \)

**Range:**
- Ending set of states

(Defined recursively, on length of input string)

- **Base case:** \( \hat{\delta}(q, \epsilon) = \{q\} \)
- **Recursive case:** \( \hat{\delta}(q, w) = \bigcup_{i=1}^{k} \delta(q_i, w_{n}) \) where: \( \hat{\delta}(q, w_1 \cdots w_{n-1}) = \{q_1, \ldots, q_k\} \)
Last Time: Adding Empty Transitions

• Define the set $\varepsilon$-REACHABLE($q$)
  
  • ... to be all states reachable from $q$ via zero or more empty transitions

(Defined recursively)

• Base case: $q \in \varepsilon$-REACHABLE($q$)

• Inductive case:

$\varepsilon$-REACHABLE($q$) = \{ r \mid p \in \varepsilon$-REACHABLE($q$) and $r \in \delta(p, \varepsilon) \}$

A state is in the reachable set if ...

... there is an empty transition to it from another state in the reachable set
Last Time: NFA Extended Transition Function

Define **extended transition function:**

\[ \hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q) \]

**Domain:**
- Beginning state \( q \in Q \)
- Input string \( w = w_1 w_2 \cdots w_n \) where \( w_i \in \Sigma \)

**Range:**
- Ending set of states

(Defined recursively, on length of input string)

**Base case:** \( \hat{\delta}(q, \epsilon) = \{q\} \)

**Recursive case:** \( \hat{\delta}(q, w) = \bigcup_{i=1}^{k} \hat{\delta}(q_i, w_n) \)

where: \( \hat{\delta}(q, w_1 \cdots w_{n-1}) = \{q_1, \ldots, q_k\} \)

"For all current states, take single step, then follow all empty transitions"
Last Time: Concatenation is Closed?

**Theorem**
The class of regular languages is closed under the concatenation operation.

In other words, if $A_1$ and $A_2$ are regular languages then so is $A_1 \circ A_2$.

**Proof**: Construct a new machine an NFA!
Let $N_1$ recognize $A_1$, and $N_2$ recognize $A_2$.

Want: Construction of $N$ to recognize $A_1 \circ A_2$.

$\varepsilon$ = “empty transition” = reads no input
Allows $N$ to be in both machines at once

$N$ is an NFA! It simultaneously:
- Keeps checking 1st part with $N_1$ and
- Moves to $N_2$ to check 2nd part
Concatenation is Closed for Regular Langs

**Proof**

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize $A_2$.

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state $q_1$ is the same as the start state of $N_1$
3. The accept states $F_2$ are the same as the accept states of $N_2$
Concatenation is Closed for Regular Langs

**Proof**

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize $A_2$.

Construct $N = (Q, \Sigma, \delta, q_1, F)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state $q_1$ is the same as the start state of $N_1$
3. The accept states $F_2$ are the same as the accept states of $N_2$
4. Define $\delta$ so that for any $q \in Q$ and any $a \in \Sigma$, 

$$\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \text{ and } q \not\in F_1 \\
\delta_1(q, a) & q \in F_1 \text{ and } a \neq \varepsilon \\
\delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\
\delta_2(q, a) & q \in Q_2.
\end{cases}$$
Flashback: A DFA’s Language

• For DFA $M = (Q, \Sigma, \delta, q_0, F)$

• $M$ accepts $w$ if $\hat{\delta}(q_0, w) \in F$

• $M$ recognizes language $A$ if $A = \{w | M$ accepts $w\}$

• A DFA’s language is a regular language
An NFA’s Language

• For NFA \( N = (Q, \Sigma, \delta, q_0, F) \)

• \( N \) accepts \( w \) if \( \hat{\delta}(q_0, w) \cap F \neq \emptyset \)
  - i.e., final states have at least one accept state

• Language of \( N = L(N) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset \} \)

Q: An NFA’s language is a regular language
Concatenation Closed for Reg Langs?

• Concatenation of DFAs produces an **NFA**

• **But** a language is only regular if a **DFA** recognizes it

• **So** to finish the proof that concatenation is closed ...  
  ... we must prove that **NFAs also recognize regular languages**.

Specifically, we must prove:

**NFAs ⇔ regular languages**
How to Prove a Statement: $X \Leftrightarrow Y$

$X \Leftrightarrow Y = "X if and only if Y" = X \text{ iff } Y = X \iff Y$

Proof at minimum has 2 required parts:

1. $\Rightarrow$ if $X$, then $Y$
   - “forward” direction
   - assume $X$, then use it to prove $Y$

2. $\Leftarrow$ if $Y$, then $X$
   - “reverse” direction
   - assume $Y$, then use it to prove $X$
Proving NFAs Recognize Regular Langs

Theorem:
A language $L$ is regular if and only if some NFA $N$ recognizes $L$.

Proof:
⇒ If $L$ is regular, then some NFA $N$ recognizes it.
   (Easier)
   • We know: if $L$ is regular, then a DFA exists that recognizes it.
   • So to prove this part: Convert that DFA to an equivalent NFA! (see HW 2)

⇔ If an NFA $N$ recognizes $L$, then $L$ is regular.
   (Harder)
   • We know: for $L$ to be regular, there must be a DFA recognizing it
   • Proof Idea for this part: Convert given NFA $N$ to an equivalent DFA

“equivalent” = “recognizes the same language”
How to convert NFA→DFA?

A finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \(Q\) is a finite set called the states,
2. \(\Sigma\) is a finite set called the alphabet,
3. \(\delta : Q \times \Sigma \rightarrow Q\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.

A nondeterministic finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \(Q\) is a finite set of states,
2. \(\Sigma\) is a finite alphabet,
3. \(\delta : Q \times \Sigma_{\varepsilon} \rightarrow 2^Q\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.

Proof idea:
Let each “state” of the DFA be a set of states in the NFA.
In a DFA, all these states at each step of NFA computation must be only one state.

So encode: a set of NFA states as one DFA state.

This is similar to the proof strategy from “Closure of union” where: a state = a pair of states
Convert **NFA→DFA**, Formally

- Let **NFA** \( N = (Q, \Sigma, \delta, q_0, F) \)

- An equivalent **DFA** \( M \) has states \( Q' = \mathcal{P}(Q) \) (power set of \( Q \))
Example:

The NFA $N_4$

A DFA $D$ that is equivalent to the NFA $N_4$
NFA→DFA

**Have:** NFA $N = (Q, \Sigma, \delta, q_0, F)$

**Want:** DFA $M = (Q', \Sigma, \delta', q_0', F')$

1. $Q' = \mathcal{P}(Q)$  
   A state for $M$ is a set of states in $N$

2. For $R \in Q'$ and $a \in \Sigma$, 
   $$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$  
   $R$ is a state in $M$ = a set of states in $N$

   Next state for DFA state $R$ = next states of each NFA state $r$ in $R$

3. $q_0' = \{q_0\}$

4. $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$

No empty transitions
Flashback: Adding Empty Transitions

• Define the set $\varepsilon$-REACHABLE$(q)$
  • ... to be all states reachable from $q$ via zero or more empty transitions

(Defined recursively)

• **Base case:** $q \in \varepsilon$-REACHABLE$(q)$

• **Inductive case:**

$$\varepsilon$-REACHABLE$(q) = \{ r \mid p \in \varepsilon$-REACHABLE$(q)$ and $r \in \delta(p, \varepsilon) \}$$

A state is in the reachable set if ...

... there is an empty transition to it from another state in the reachable set
NFA→DFA

Have: NFA $N = (Q, \Sigma, \delta, q_0, F)$

Want: DFA $M = (Q', \Sigma, \delta', q_0', F')$

1. $Q' = \mathcal{P}(Q)$

2. For $R \in Q'$ and $a \in \Sigma$,
   \[ \delta'(R, a) = \bigcup_{r \in R} \delta(r, a) \text{ ε-REACHABLE}(\delta(r, a)) \]

3. $q_0' = \{q_0\} \text{ ε-REACHABLE}(q_0)$

4. $F' = \{R \in Q' | R \text{ contains an accept state of } N\}$

With empty transitions
Proving NFAs Recognize Regular Langs

Theorem:
A language $L$ is regular if and only if some NFA $N$ recognizes $L$.

Proof:
⇒ If $L$ is regular, then some NFA $N$ recognizes it.
  • We know: If $L$ is regular, then a DFA recognizes it.
  • We show: How to convert a DFA to an equivalent NFA (proved in hw2)

⇐ If an NFA $N$ recognizes $L$, then $L$ is regular.
  • We know: For $L$ to be regular, there must be a DFA recognizing it
  • We show: How to convert NFA $N$ to an equivalent DFA ...
  • ... using the NFA→DFA algorithm we just defined!
Concatenation is Closed for Regular Langs

**Proof**

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize $A_2$.

Construct $N = (Q, \Sigma, \delta, q_1, F_1 \circ A_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state $q_1$ is the same as the start state of $N_1$
3. The accept states $F_2$ are the same as the accept states of $N_2$
4. Define $\delta$ so that for any $q \in Q$ and any $a \in \Sigma$, $\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\
\delta_1(q, a) & q \in F_1 \text{ and } a \neq \varepsilon \\
\delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\
\delta_2(q, a) & q \in Q_2.
\end{cases}$
Flashback: Union is Closed For Regular Langs

**Theorem**

The class of regular languages is closed under the union operation. In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$.

**Proof:**

- How do we prove that a language is regular?
  - Create a DFA or NFA recognizing it!
- Combine the machines recognizing $A_1$ and $A_2$
  - Should we create a DFA or NFA?
**Flashback:** Union is Closed For Regular Langs

**Proof**

- **Given:**
  \[ M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1), \text{ recognize } A_1, \]
  \[ M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2), \text{ recognize } A_2, \]

- **Construct:** a new machine \( M = (Q, \Sigma, \delta, q_0, F) \) using \( M_1 \) and \( M_2 \)

- **states of** \( M \): 
  \[ Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2 \]
  This set is the **Cartesian product** of sets \( Q_1 \) and \( Q_2 \)

- **\( M \) transition fn:** 
  \[ \delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \]

- **\( M \) start state:** 
  \((q_1, q_2)\)

- **\( M \) accept states:** 
  \[ F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\} \]
Union is Closed for Regular Languages

Add new start state, and $\epsilon$-transitions to old start states
Union is Closed for Regular Languages

PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize $A_2$.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
2. The state $q_0$ is the start state of $N$.
3. The set of accept states $F = F_1 \cup F_2$. 

Alternate Proof, with NFAs
Union is Closed for Regular Languages

**PROOF**

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize $A_2$.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.

2. The state $q_0$ is the start state of $N$.

3. The set of accept states $F = F_1 \cup F_2$.

4. Define $\delta$ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \\
\delta_2(q, a) & q \in Q_2 \\
\{q_1 \cap q_2\} & q = q_0 \text{ and } a = \varepsilon \\
\emptyset & q = q_0 \text{ and } a \neq \varepsilon 
\end{cases}
$$
List of Closed Ops for Reg Langs (so far)

☑️ • Union

☑️ • Concatenation

• Kleene Star (repetition)
Kleene Star Example

Let the alphabet $\Sigma$ be the standard 26 letters $\{a, b, \ldots, z\}$.

If $A = \{\text{good}, \text{bad}\}$ and $B = \{\text{boy}, \text{girl}\}$, then

$$A^* = \{\varepsilon, \text{good}, \text{bad}, \text{goodgood}, \text{goodbad}, \text{badgood}, \text{badbad}, \text{goodgoodgood}, \text{goodgoodbad}, \text{goodbadgood}, \text{goodbadbad}, \ldots \}$$

Note: repeat zero or more times

(this is an infinite language!)
Kleene Star

New start (and accept) state, $\varepsilon$-transitions to old start state

Old accept states $\varepsilon$-transition to old start state
Kleene Star is Closed for Regular Langs

**Theorem**

The class of regular languages is closed under the star operation.
Kleene Star is Closed for Regular Langs

**Proof**  Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$. Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1^*$. 

1. $Q = \{q_0\} \cup Q_1$
2. The state $q_0$ is the new start state.
3. $F = \{q_0\} \cup F_1$

Kleene star of a language must accept the empty string!
Kleene Star is Closed for Regular Langs

**Proof** Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$. Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1^*$.  

1. $Q = \{q_0\} \cup Q_1$
2. The state $q_0$ is the new start state.
3. $F = \{q_0\} \cup F_1$
4. Define $\delta$ so that for any $q \in Q$ and any $a \in \Sigma$,  

$$
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\
\delta_1(q, a) & q \in F_1 \text{ and } a \neq \varepsilon \\
\delta_1(q, \varepsilon) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\
\{q_1\} & q = q_0 \text{ and } a = \varepsilon \\
\emptyset & q = q_0 \text{ and } a \neq \varepsilon.
\end{cases}
$$
Many More Closed Operations on Regular Languages!

- Complement
- Intersection
- Difference
- Reversal
- Homomorphism
- (See HW2)
Why do we care about these ops?

• Union
• Concat
• Kleene star

• The are sufficient to represent all regular languages!
• I.e., they define regular expressions
Check-in Quiz 9/27

On gradescope