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

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 2 in
  • Due Tue 2/14 11:59pm

• HW 3 out
  • Due Sun 2/26 11:59pm
  • Note: extended due date

• Office Hours
  • Woody’s time moved: Tue 4-5:30pm, McCormack 3rd floor, room 139

• No lecture next Monday 2/20

Quiz Preview
• An "if and only if" statement represents two of what kind of statements?
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?
Concatenation Examples

**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 $A_1 \circ A_2$.

- If: $a_1 \in A_1$, $a_2 \in A_2$
- If: $a_3 \in A_1$, $a_4 \notin A_2$
- If: $a_5 \notin A_1$, $a_6 \in A_2$
- If: $a_7 \notin A_1$, $a_8 \notin A_2$
- Then: $a_1 a_2 \in A_1 \circ A_2$ ???
- Then: $a_3 a_4 \in A_1 \circ A_2$ ???
- Then: $a_5 a_6 \in A_1 \circ A_2$ ???
- Then: $a_7 a_8 \in A_1 \circ A_2$ ???
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?

- How does it know when to switch machines?
  - Can only read input once
Let $M_1$ recognize $A_1$, and $M_2$ recognize $A_2$.

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

$N$ is an NFA! It can:
- Keep checking 1st part with $M_1$ and
- Move to $M_2$ to check 2nd part

$\varepsilon$ = “empty transition” = reads no input

Allows $N$ to be in both machines at the same time!
Concatenation is Closed for Regular Langs

**Proof**

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$
Let $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ 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 $M_1$
3. The accept states $F_2$ are the same as the accept states of $M_2$
Concatenation is Closed for Regular Langs

**Proof**

Let \( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) recognize \( A_1 \)

\( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \) recognize \( A_2 \)

Construct \( N = (Q, \Sigma, \delta, q_1, F_1) \) 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 \( M_1 \)
3. The accept states \( F_2 \) are the same as the accept states of \( M_2 \)
4. Define \( \delta \) so that for any \( q \in Q \) and any \( a \in \Sigma \),

\[
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\
\delta_2(q, a) & \text{if } q \in Q_2 \\
\delta_1(q, a) & \text{if } q \in F_1 \text{ and } a \neq \varepsilon \\
\text{?} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\
\text{?} & \text{if } q \notin F_1 \end{cases}
\]

\( \delta(q, \varepsilon) = \emptyset \), for \( q \in Q, q \notin F_1 \)
Flashback: Is Union Closed For Regular Langs?

Statements
1. $A_1$ and $A_2$ are regular languages
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes $A_1$
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes $A_2$
4. Construct DFA $M = (Q, \Sigma, \delta, q_0, F)$
5. $M$ recognizes $A_1 \cup A_2$
6. $A_1 \cup A_2$ is a regular language
7. 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 $A_1 \cup A_2$.

Justifications
1. Assumption
2. Def of Regular Language
3. Def of Regular Language
4. Def of DFA
5. See examples
6. Def of Regular Language
7. From stmt #1 and #6
Is Concat Closed For Regular Langs?

Statements
1. $A_1$ and $A_2$ are regular languages
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes $A_1$
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes $A_2$
4. Construct NFA $N = ???$ (todo)
5. $N$ recognizes $A_1 \cup A_2 \cup A_1 \circ A_2$
6. $A_1 \circ A_2$ is a regular language
7. 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 $A_1 \circ A_2$.

Justifications
1. Assumption
2. Def of Regular Language
3. Def of Regular Language
4. Def of NFA
5. See examples
6. Does NFA recognize regular lang?
7. From stmt #1 and #6
Flashback: A DFA’s Language

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

- $M$ accepts $w$ if $\delta(q_0, w) \in F$

- $M$ recognizes language $\{w \mid M$ accepts $w\}$

Definition: 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., accept if final states contain at least one accept state

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

Q: What kind of languages do NFAs recognize?
Concatenation Closed for Reg Langs?

- Combining DFAs to recognize concatenation of languages ...
  
  ... produces an NFA

- So to prove concatenation is closed ...
  
  ... we must prove that NFAs also recognize regular languages.

Specifically, we must prove:

NFAs ⇔ regular languages
“If and only if” Statements

\[ X \iff Y = \text{“}X \text{ if and only if } Y\text{”} = X \text{ iff } Y = X \iff Y \]

Represents two statements:

1. \( \Rightarrow \) if \( X \), then \( Y \)
   - “forward” direction

2. \( \iff \) if \( Y \), then \( X \)
   - “reverse” direction
How to Prove an “iff” Statement

\[ X \leftrightarrow Y = "X \text{ if and only if } Y" = X \text{ iff } Y = X \leftrightarrow Y \]

Proof has two (If-Then proof) 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 $\rightarrow$ an equivalent NFA! (see HW 2)

⇐ If an NFA $N$ recognizes $L$, then $L$ is regular.
\[ \Rightarrow \text{If } L \text{ is regular, then some NFA } N \text{ recognizes it} \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Justifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( L ) is a regular language</td>
<td>1. Assumption [ Assume the “if” part \ldots ]</td>
</tr>
<tr>
<td>2. A DFA ( M ) recognizes ( L )</td>
<td>2. Def of Regular language</td>
</tr>
<tr>
<td>3. Construct NFA ( N ) equiv to ( M )</td>
<td>3. See hw 2!</td>
</tr>
<tr>
<td>4. An NFA ( N ) recognizes ( L )</td>
<td>4. ???</td>
</tr>
<tr>
<td>5. If ( L ) is a regular language, then some NFA ( N ) recognizes it</td>
<td>5. ByStmts #1 and #4</td>
</tr>
</tbody>
</table>
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 $\rightarrow$ 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 $N$ $\rightarrow$ 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. \(\delta: Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.

Proof idea:
Let each “state” of the DFA = set of states in the NFA
Symbol read

0

1

0

1

1

0

NFA computation can be in multiple states

DFA computation can only be in 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 \( Q' = P(Q) \) (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 DFA state = a set of NFA states

2. For $R \in Q'$ and $a \in \Sigma$,
   $$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$  
   A DFA step = an NFA step for all states in the set

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

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

No empty transitions
Flashback: Adding Empty Transitions

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

(Defined recursively)

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

- **Recursive case:**
  $$\epsilon$${REACHABLE}(q) = \{ r \mid p \in \epsilon$${REACHABLE}(q) \text{ and } r \in \delta(p, \epsilon) \}$$

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 $a \in \Sigma$,
   $$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a) \subseteq \varepsilon\text{-REACHABLE}(\delta(r, a))$$

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

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

With empty transitions

Almost the same, except ...

But this produces a set! We need another "reachable" function (see hw 3!)
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 → 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 $N$ → an equivalent DFA ...
      ... using our NFA to DFA algorithm!
Concatenation is Closed for Regular Langs

**Proof**

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$

Let $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ 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 $M_1$
3. The accept states $F_2$ are the same as the accept states of $M_2$
4. Define $\delta$ so that for any $q \in Q$ and any $a \in \Sigma$, 
   
   \[
   \delta(q, a) = \begin{cases} 
   \delta_1(q, a) & q \in Q_1 \text{ and } q \not\in F_1 \\
   \delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\
   \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\
   \delta_2(q, a) & q \in Q_2.
   \end{cases}
   \]

If a language has an NFA recognizing it, then it is a regular language.
Concat Closed for Reg Langs: Use NFAs Only

**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 $F$ are the same as the accept states of $N_2$
4. Define $\delta$ so that for any $q \in Q$ and any $a \in \Sigma$,

   $\delta(q, a) = \begin{cases} 
   \delta_1(q, a) & q \in Q_1 \text{ and } q \not\in F_1 \\
   \delta_2(q, a) & q \in Q_2 \\
   \delta_1(q, a) & q \in F_1 \text{ and } a \neq \varepsilon \\
   \{q_2\} & q \in F_1 \text{ and } a = \varepsilon 
   \end{cases}$

If language is regular, then it has an NFA recognizing it ...
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: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize $A_1$, 
  $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize $A_2$,

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

- states of $M$: $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$
  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: $F = \{(r_1, r_2) | r_1 \in F_1 \text{ or } r_2 \in F_2\}$

- State in $M = M_1$ state + $M_2$ state

- $M$ step = a step in $M_1$ + a step in $M_2$

- Accept if either $M_1$ or $M_2$ accept
Union is Closed for Regular Languages

Add new start state, and $\varepsilon$-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 $a \in \Sigma_e$, 

$$
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \\
\delta_2(q, a) & q \in Q_2 \\
\{ q_1 \cup q_2 \} & q = q_0 \text{ and } a = \varepsilon \\
\emptyset & q = q_0 \text{ and } a \neq \varepsilon 
\end{cases}
$$

Don’t forget Statements and Justifications!
List of Closed Ops for Reg Langs (so far)

- Union
- Concatentation
  - Kleene Star (repetition) ?
Kleene Star Example

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

If $A = \{\text{good, bad}\}$

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

Note: repeat zero or more times

(this is an infinite language!)
New start (and accept) state,
$\varepsilon$-transitions to old start state

Old accept states
$\varepsilon$-transition to old start state

Kleene Star
In-class exercise:
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 $A_1^*$. 
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 $A_1^\ast$.

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 $a \in \Sigma_e$,

$$
\delta(q, a) = \begin{cases}
\delta_1(q, a) & q \in Q_1 \text{ and } q \not\in F_1 \\
\delta_1(q, a) & q \in F_1 \text{ and } a \neq \varepsilon \\
\delta_1(q, \varepsilon) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\
\{q_1\} & q = q_0 \text{ and } a = \varepsilon \\
\emptyset & q = q_0 \text{ and } a \neq \varepsilon.
\end{cases}
$$
Next Time: Why These Closed Operations?

- Union
- Concat
- Kleene star

All regular languages can be constructed from:
- single-char strings, and
- these three combining operations!
Check-in Quiz 2/15
On gradescope