CS 622
Regular Languages Are Closed Under Concatenation
Friday, February 23, 2024
UMass Boston CS
**Announcements**

- HW 3 out
  - Due Mon 3/4 12pm EST (noon)
DFA Extended Transition Function

\[ \hat{\delta} : Q \times \Sigma^* \rightarrow Q \]

- **Domain** (inputs):
  - state \( q \in Q \) (doesn’t have to be start state)
  - string \( w = w_1 w_2 \cdots w_n \) where \( w_i \in \Sigma \)

- **Range** (output):
  - state \( q \in Q \) (doesn’t have to be an accept state)

\[ \delta : Q \times \Sigma \rightarrow Q \text{ is the transition function} \]
\( \hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q) \) is the transition function

\[\begin{align*}
\text{Domain (inputs):} \\
\text{state } q \in Q \text{ (doesn't have to be start state)} \\
\text{string } w = w_1 w_2 \cdots w_n \text{ where } w_i \in \Sigma
\end{align*}\]

\[\begin{align*}
\text{Range (output):} \\
qs \subseteq Q
\end{align*}\]
\( \hat{\delta}: Q \times \Sigma^* \rightarrow \mathcal{P}(Q) \) is the transition function

\( \delta: Q \times \Sigma \rightarrow \mathcal{P}(Q) \) is the transition function

\( \hat{\delta}(q, \varepsilon) = \{q\} \)

**NFA**

Extended Transition Function

- **Domain** (inputs):
  - state \( q \in Q \) (doesn’t have to be start state)
  - string \( w = w_1w_2\cdots w_n \) where \( w_i \in \Sigma \)

- **Range** (output):
  - states \( qs \subseteq Q \)

(Defined recursively)

**Base case**

Recursively Defined Input

needs

Recursive Function

A String is either:

- the **empty string** \((\varepsilon)\), or
- \(xa\) (non-empty string) where
  - \(x\) is a **string**
  - \(a\) is a “char” in \(\Sigma\)
Extended Transition Function

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

- **Domain** (inputs):
  - state \( q \in Q \) (doesn’t have to be start state)
  - string \( w = w_1w_2 \cdots w_n \) where \( w_i \in \Sigma \)
- **Range** (output):
  - states \( qs \subseteq Q \)

(Defined recursively)

Base case \[ \hat{\delta}(q, \varepsilon) = \{q\} \]

Recursive Case

\[ \hat{\delta}(q, w'w_n) = \hat{\delta}(q, w') \]

where \( w' = w_1 \cdots w_{n-1} \)

\[ \hat{\delta}(q, w') = \{q_1, \ldots, q_k\} \]
Extended Transition Function

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

- **Domain** (inputs):
  - state \( q \in Q \) (doesn’t have to be start state)
  - string \( w = w_1 w_2 \cdots w_n \) where \( w_i \in \Sigma \)
- **Range** (output):
  - states \( q_s \subseteq Q \)

(Defined recursively)

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

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

where \( w' = w_1 \cdots w_{n-1} \)

\[ \hat{\delta}(q, w') = \{ q_1, \ldots, q_k \} \]

\[ \delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q) \] is the transition function

We haven’t considered empty transitions!

A String is either:
- the **empty string** (\( \varepsilon \)), or
- \( xa \) (non-empty string) where
  - \( x \) is a **string**
  - \( a \) is a “char” in \( \Sigma \)

Recursively Defined Input needs Recursive Function

For each “second to last” state, take single step on last char

Last char
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
Extended Transition Function

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

- **Domain** (inputs):
  - state \( q \in Q \) (doesn’t have to be start state)
  - string \( w = w_1w_2\cdots w_n \) where \( w_i \in \Sigma \)
- **Range** (output):
  - states \( q_s \subseteq Q \)

(Defined recursively)

Base case \( \hat{\delta}(q, \varepsilon) = \varepsilon\text{-REACHABLE}(q) \)

Recursive Case \( \hat{\delta}(q, w'w_n) = \)

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

\( \bigcup_{i=1}^{k} \delta(q_i, w_n) = \{r_1, \ldots, r_{\ell}\} \)
Extended Transition Function

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

- **Domain** (inputs):
  - state $q \in Q$ (doesn’t have to be start state)
  - string $w = w_1w_2 \cdots w_n$ where $w_i \in \Sigma$
- **Range** (output):
  - states $q_s \subseteq Q$

(Defined recursively)

**Base case**  \( \hat{\delta}(q, \varepsilon) = \varepsilon\text{-REACHABLE}(q) \)

**Recursive Case**  \( \hat{\delta}(q, w'w_n) = \bigcup_{j=1}^{\ell} \varepsilon\text{-REACHABLE}(r_j) \)

Handling \( \varepsilon \) transitions now!

\[ \delta(q, w') = \{q_1, \ldots, q_k\} \]

\[ \bigcup_{i=1}^{k} \delta(q_i, w_n) = \{r_1, \ldots, r_{\ell}\} \]

All chars except last

“second to last” set of states

“last” set of states (no \( \varepsilon \))
Summary: NFA vs DFA Computation

**DFAs**
- Can only be in **one** state
- Transition:
  - **Must read 1 char**
- Acceptance:
  - If **final state is accept state**

**NFAs**
- Can be in **multiple** states
- Transition
  - **Has empty transitions**
- Acceptance:
  - If **one of final states is accept state**
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 \( A_1 \circ A_2 \).

*Proof requires*: Constructing 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

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

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

**Proof** (part of)

Let $DFA \ M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$
DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ 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 $M_1$
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 $a \in \Sigma_{\varepsilon}$,
Concatenation is Closed for Regular Langs

\textbf{Proof} (part of)

Let \( \text{DFA } M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) recognize \( A_1 \)

\( \text{DFA } M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \) recognize \( A_2 \)

Construct \( N = (Q, \Sigma, \delta, q, F_1) \) 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 \( a \in \Sigma \),

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

And: \( \delta(q, \varepsilon) = \emptyset \), for \( q \in Q, q \notin F_1 \)
Concatenation is Closed for Regular Langs

**Proof** (part of)

Let $DFA \ M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$
$DFA \ M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ 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 $a \in \Sigma$, 

$$\delta(q, a) = \begin{cases} 
\{\delta_1(q, a)\} & q \in Q_1 \text{ and } q \notin F_1 \\
\{\delta_1(q, a)\} & q \in F_1 \text{ and } a \neq \epsilon \\
\{q_2\} & q \in F_1 \text{ and } a = \epsilon \\
\{\delta_2(q, a)\} & q \in Q_2.
\end{cases}$$

And: $\delta(q, \epsilon) = \emptyset$, for $q \in Q, q \notin F_1$
Previously

Is Union Closed For Regular Langs?

Proof

**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.

**Justifications**

1. Assumption
2. Def of Reg Lang (Coro)
3. Def of Reg Lang (Coro)
4. Def of DFA
5. See Examples Table
6. Def of Regular Language
7. From stmt #1 and #6

In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$. Q.E.D.
Is Concat Closed For Regular Langs?

Proof?

**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 $M = (Q, \Sigma, \delta, q_0, F)$
5. $M$ recognizes $A_1 \cup A_2$ $A_1 \circ A_2$
6. $A_1 \cup A_2$ $A_1 \circ A_2$ is a regular language
7. The class of regular languages is closed under concatenation operation.

**Justifications**
1. Assumption
2. Def of Reg Lang (Coro)
3. Def of Reg Lang (Coro)
4. Def of NFA
5. See Examples Table
6. Does NFA recognize reg langs?
7. From stmt #1 and #6

Q.E.D.
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 $\{w | M$ accepts $w\}$

Definition: A DFA’s language is a regular language
An NFA’s Language?

• For NFA $N = (Q, \Sigma, \delta, q_0, F)$
  - Intersection ...
  - ... with accept states ...
  - ... is not empty set

• $N$ accepts $w$ if $\hat{\delta}(q_0, w) \cap F \neq \emptyset$
  - i.e., accept if final states contains 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 $\Leftrightarrow$ regular languages
“If and only if” Statements

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

Represents two statements:

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

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

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

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 \( X \)
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. $\delta : Q \times \Sigma_e \rightarrow \mathcal{P}(Q)$ 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*. 
Proving NFAs Recognize Regular Langs

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

Proof:  2 parts

$\Rightarrow$ 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 3)

$\Leftarrow$ If an NFA $N$ recognizes $L$, then $L$ is regular.
⇒ If $L$ is regular, then some NFA $N$ 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</td>
</tr>
<tr>
<td>2. A DFA $M$ recognizes $L$</td>
<td>2. Def of Regular lang (Coro)</td>
</tr>
<tr>
<td>3. Construct NFA $N = \text{convert}(M)$</td>
<td>3. See hw 2 3!</td>
</tr>
<tr>
<td>4. DFA $M$ is equivalent to NFA $N$</td>
<td>4. See Equiv. table!</td>
</tr>
<tr>
<td>5. An NFA $N$ recognizes $L$</td>
<td>5. ???</td>
</tr>
<tr>
<td>6. If $L$ is a regular language, then some NFA $N$ recognizes it</td>
<td>6. By Stmts #1 and # 5</td>
</tr>
</tbody>
</table>

Assume the "if" part ...

... use it to prove "then" part
“Proving” Machine Equivalence (Table)

Let: DFA $M = (Q, \Sigma, \delta, q_0, F)$
NFA $N = \text{convert}(M)$
$\hat{\delta}(q_0, w) \in F$ for some string $w$

<table>
<thead>
<tr>
<th>String</th>
<th>$M$ accepts?</th>
<th>$N$ accepts?</th>
<th>$N$ accepts? Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w$</td>
<td>Yes</td>
<td>???</td>
<td>See justification #1</td>
</tr>
<tr>
<td>$w'$</td>
<td>No</td>
<td>???</td>
<td>See justification #2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If $M$ accepts $w$ ...
Then we know ...

There is some sequence of states: $r_1 \ldots r_n$, where $r_i \in Q$ and $r_1 = q_0$ and $r_n \in F$

Then $N$ accepts?/rejects? $w$ because ...

Justification #1?
There is an accepting sequence of set of states in $N$ ... for string $w$
“Proving” Machine Equivalence (Table)

Let: DFA $M = (Q, \Sigma, \delta, q_0, F)$
NFA $N = \text{convert}(M)$

$\hat{\delta}(q_0, w) \in F$ for some string $w$
$\hat{\delta}(q_0, w') \in F$ for some string $w'$

If $M$ accepts $w'$ ...
Then we know ...

<table>
<thead>
<tr>
<th>String</th>
<th>$M$ accepts?</th>
<th>$N$ accepts?</th>
<th>$N$ accepts? Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w$</td>
<td>Yes</td>
<td>???</td>
<td>See justification #1</td>
</tr>
<tr>
<td>$w'$</td>
<td>No</td>
<td>???</td>
<td>See justification #2?</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Then $N$ accepts?/rejects? $w'$ because ...

Justification #2?
Proving NFAs Recognize Regular Langs

Theorem:
A language \( L \) is regular if and only if some NFA \( N \) recognizes \( L \).

Proof:
\( \Rightarrow \) 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 3)

\( \Leftarrow \) 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

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 \( Q' = \mathcal{P}(Q) \) (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 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 $\varepsilon$-REACHABLE($q$)
  • ... to be all states reachable from $q$ via zero or more empty transitions

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

• **Recursive case:**

$\varepsilon$-REACHABLE($q$) = \{ r | 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 $\rightarrow$ 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 $a \in \Sigma$,
   $$\delta'(R, a) = \bigcup_{s \in S} \varepsilon\text{-REACHABLE}(s)$$

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

S = \bigcup_{r \in R} \delta(r, a)$
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 3)

⇐ 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 $A_1$
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ 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 $a \in \Sigma_\varepsilon$,

$$
\delta(q, a) = \begin{cases}
\{\delta_1(q, a)\} & q \in Q_1 \text{ and } q \notin F_1 \\
\{\delta_1(q, a)\} & q \in F_1 \text{ and } a \neq \varepsilon \\
\{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\
\{\delta_2(q, a)\} & q \in Q_2.
\end{cases}
$$

And: $\delta(q, \varepsilon) = \emptyset$, for $q \in Q$, $q \notin F_1$
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 $N = (Q, \Sigma, \delta, q_1, F_2)$ 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 $a \in \Sigma_e$,

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

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: $F = \{(r_1, r_2) | r_1 \in F_1 \text{ or } r_2 \in F_2\}$
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 $F = F_1 \cup F_2$. 

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 $a \in \Sigma_e$,

\[
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \\
\delta_2(q, a) & q \in Q_2 \\
\{q_{q1}, q_{q2}\} & q = q_0 \text{ and } a = \varepsilon \\
\emptyset & q = q_0 \text{ and } a \neq \varepsilon
\end{cases}
\]
List of Closed Ops for Reg Langs (so far)

- Union

- 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!)
Kleene Star

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

Old accept states $\varepsilon$-transition to old start state
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 $A_1^\varepsilon$.

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 $a \in \Sigma_\varepsilon$,

$$
\delta(q, a) = \begin{cases}
\delta_1(q, a) & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\
\delta_1(q, a) & \text{if } q \in F_1 \text{ and } a \neq \varepsilon \\
\delta_1(q, a) \cup \{q_1\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\
\{q_1\} & \text{if } q = q_0 \text{ and } a = \varepsilon \\
\emptyset & \text{if } q = q_0 \text{ and } a \neq \varepsilon.
\end{cases}
$$
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!
Submit in-class work 2/26

On gradescope