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. \(\delta: Q \times \Sigma \longrightarrow \mathcal{P}(Q)\) 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. \(\delta: Q \times \Sigma \longrightarrow Q\) 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

• HW 2 out
  • Due Sun 2/13 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)

• Tip: Designing a machine = programming
Last Time

Let $N_1$ recognize $A_1$, and $N_2$ recognize $A_2$.

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

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

Allows NFA $N$ to be “in” both machines at once

Does this prove concatenation is closed for regular languages?
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 $w\}$

Definition: A language is a regular language if a DFA recognizes it
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., if the 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: How does an NFA’s language relate to regular languages?
All we know so far: A language is regular if a DFA recognizes it
So is Concatenation Closed for Reg Langs?

• Concatenation of DFAs produces an NFA

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

• To finish the proof that concat is closed ...
  ... we must prove that NFAs also recognize regular languages.

Specifically, we must prove:

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

- $X \Leftrightarrow Y = \text{“} X \text{ if and only if } Y \text{”} = X \text{ iff } Y = X \leftrightarrow Y$
- 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. \(\delta: Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)\) 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 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 design a state in the DFA to be a set of NFA states!

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 \( a \in \Sigma \),  
   \[
   \delta'(R, a) = \bigcup_{r \in R} \delta(r, a)
   \]
   - \( R = \) 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. \( F' = \{R \in Q' | \ R \text{ contains an accept state of } N\} \)
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 $a \in \Sigma$,
   \[
   \delta'(R, a) = \bigcup_{r \in R} \delta(r, a) \cup \varepsilon\text{-REACHABLE}(\delta(r, a))
   \]

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 ...

Requires extending the fn to sets of states (see HW 2)
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

⇐ 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!
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!

• Create machine combining 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** \( M \):
  \[ Q = \{ (r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2 \} = Q_1 \times Q_2 \]
  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:**
  \[ F = \{ (r_1, r_2) | r_1 \in F_1 \text{ or } r_2 \in F_2 \} \]

\[ M \text{ step} = \text{a step in } M_1 + \text{a step in } M_2 \]

**State in** \( M = M_1 \text{ state + } M_2 \text{ state} \)

Accept if either \( M_1 \) or \( M_2 \) accept
Union is Closed for Regular Languages
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 $a \in \Sigma$, 

$$
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \\
\delta_2(q, a) & q \in Q_2 \\
\{q_1 ? q_2\} & q = q_0 \text{ and } a = \varepsilon \\
\emptyset & q = q_0 \text{ and } a \neq \varepsilon
\end{cases}
$$
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 \( N = (Q, \Sigma, \delta, q_1, F_1) \) 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 \( a \in \Sigma_\varepsilon \),

\[
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\
\delta_1(q, a) & q \in F_1 \text{ and } a \neq \varepsilon \\
\delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\
\delta_2(q, a) & q \in Q_2.
\end{cases}
\]
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 $A = \{\text{good, bad}\}$ and $B = \{\text{boy, girl}\}$, then

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

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
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 $A_1^\ast$.

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

2. The state $q_0$ is the new start state.

3. $F = \{q_0\} \cup F_1$

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 $a \in \Sigma$,

$$
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\
\delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\
\delta_1(q, a) \cup \{q_1\} & q \in F_1 \text{ and } a = \epsilon \\
\{q_1\} & q = q_0 \text{ and } a = \epsilon \\
\emptyset & q = q_0 \text{ and } a \neq \epsilon.
\end{cases}
$$
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

- They are sufficient to represent all regular languages!
- I.e., they define regular expressions
So Far: Regular Language Representations

1. State diagram (NFA/DFA)
   ![State Diagram]

2. Formal description
   1. \( Q = \{ q_1, q_2, q_3 \} \),
   2. \( \Sigma = \{ 0, 1 \} \),
   3. \( \delta \) is described as
   
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 )</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( q_3 )</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( q_2 )</td>
</tr>
</tbody>
</table>

3. \( \Sigma^*001\Sigma^* \)

4. \( q_1 \) is the start state, and
5. \( F = \{ q_2 \} \).

A practical application: text search ... it doesn’t fit!

These define a computer (program) that finds strings containing 001

Need a more concise notation
Regular Expressions Are Widely Used

- Perl
- Python
- Java
- Every lang!
Regular Expressions: Formal Definition

A regular expression $R$ is

1. a for some $a$ in the alphabet $\Sigma$, (A lang containing a) length-1 string
2. $\varepsilon$, (A lang containing) the empty string
3. $\emptyset$, The empty set (i.e., a lang containing no strings)
4. $(R_1 \cup R_2)$, where $R_1$ and $R_2$ are regular expressions, union
5. $(R_1 \circ R_2)$, where $R_1$ and $R_2$ are regular expressions, or concat
6. $(R_1^*)$, where $R_1$ is a regular expression, star

Base cases plus union, concat, and Kleene star can express any regular language! (But we have to prove it)
Regular Expression: Concrete Example

Entire reg expr: represents lang whose strings are strings from these langs concat’ed together (implicit concat op)

- the lang \{"0","1"\}
- \((0 \cup 1)0^*\)
- the lang \{"", "0", "00", ...
- the lang \{"0\"
- the lang \{"1\"

- Operator **Precedence:**
  - Paren
  - Star
  - Concat (sometimes implicit)
  - Union

---

*R is a regular expression if \( R \) is*

1. \( a \) for some \( a \) in the alphabet \( \Sigma \),
2. \( \varepsilon \),
3. \( \emptyset \),
4. \((R_1 \cup R_2)\), where \( R_1 \) and \( R_2 \) are regular expressions,
5. \((R_1 \circ R_2)\), where \( R_1 \) and \( R_2 \) are regular expressions, or
6. \((R_1^*)\), where \( R_1 \) is a regular expression.
Check-in Quiz 2/7

On gradescope