UMB CS622
Closed Operations on Regular Languages
Monday, September 20, 2021
Announcements

• HW1 due yesterday

• HW2 released, due Sun 9/26 11:59pm EST

• **Reminder**: Post HW questions to Piazza
  • Use anonymous post if you don’t want anyone to see

• Midterm / Final exam cancelled
Last Time: NFAs vs DFAs

**DFAs**
- Can only be in **one** state
- Transitions:
  - Always reads one char
  - A state **must have** a transition for every char
- Acceptance:
  - If final state is accept state

**NFAs**
- Can be in **multiple** states
- Transitions:
  - Can read no chars, i.e., empty transition
  - A state **might not have** transitions for every char
- Acceptance:
  - If **one of** final states is accept state
Last Time: NFAs and Regular Languages

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

**Proof:**
$\Rightarrow$ If $A$ is regular, then some NFA $N$ recognizes it
  * Easier
  * We know: if $A$ is regular, then a DFA recognizes it.
  * Convert DFA to an NFA! (see HW2)

$\Leftarrow$ If an NFA $N$ recognizes $A$, then $A$ is regular.
  * Harder
  * We know: a language is regular if a DFA recognizes it.
  * Convert NFA to DFA
Last Time: 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 must be only **one** state.

So design a state in the DFA to be a **set of NFA states**!
Converting NFA to 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: \( N = (Q, \Sigma, \delta, q_0, F) \)

Want to: construct a 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 \( a \in \Sigma \),
   \[ \delta'(R, a) = \bigcup_{r \in R} \delta(r, a) \]
   To compute a single step in the DFA...
   compute next states of each NFA state \( r \) in \( R \),
   then union results together

3. \( q'_0 = \{ q_0 \} \)
   \( R = \) a state in \( M = \) a set of states in \( N \)

4. \( F' = \{ R \in Q' | R \text{ contains an accept state of } N \} \)
NFA→DFA  Proof of Correctness

Let \( N = (Q_N, \Sigma, \delta_N, q_0, F_N) \)
And let \( \text{NFA→DFA}(N) = D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D) \)

**Correctness criteria:** \( \text{LANGUAGEOF}(N) = \text{LANGUAGEOF}(D) \)
  \- i.e., for all strings \( w \), \( N \) accepts \( w \) **if and only if** \( D \) accepts \( w \)

- We will first prove a **stronger** statement: \( \hat{\delta}_D(\{q_0\}, w) = \delta_N(q_0, w) \)
  \- i.e., for all strings \( w \), the DFA and NFA end in the same set of states!

Remember: A state in the DFA is a set of states in the NFA

No empty transitions
NFA→DFA  Proof of Correctness

Let

\[ N = (Q_N, \Sigma, \delta_N, q_0, F_N) \]

And let \( \text{NFA→DFA}(N) = D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D) \)

Theorem: \( \hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w) \)

Proof: (by induction on length of \( w \))

• **Base case** \( w = \epsilon \)
  \[ \hat{\delta}_D(\{q_0\}, \epsilon) \text{ and } \hat{\delta}_N(q_0, \epsilon) = \]

• **Inductive case** \( w = xa \)
  \( a = \text{last char} \)
  • **IH**: \( \hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x) \), call this set of states \( R \)
  • **NFA last step** (from \( \delta_N \) definition)
    \[ \bigcup_{r \in R} \delta_N(r, a) \]
  • **DFA last step** (from NFA→DFA definition)
    \[ \bigcup_{r \in R} \delta_N(r, a) \]

This produces a set bc we defined states to be sets of states

This produces a set bc of the definition of NFAs

Go back and review previous definitions to confirm that they are the same

No empty transitions
Last Time: Adding Empty Transitions

Define the set $\varepsilon$-REACHABLE(q)

... to be all states reachable from q via one or more empty transitions

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

• **Inductive 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: \[ N = (Q, \Sigma, \delta, q_0, F) \]

Want to: construct a DFA \[ M = (Q', \Sigma, \delta', q_0', F') \]

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

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

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

4. \( F' = \{R \in Q'\mid R \text{ contains an accept state of } N\} \)
NFA→DFA\(\varepsilon\) Proof of Correctness

Let \(N = (Q_N, \Sigma, \delta_N, q_0, F_N)\)
And let \(\text{NFA→DFA}(N) = D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)\)

**Correctness criteria:** \(\text{LANGUAGEOF}(N) = \text{LANGUAGEOF}(D)\)
- i.e., for all strings \(w\), \(N\) accepts \(w\) if and only if \(D\) accepts \(w\)

- We will first prove a stronger statement: \(\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)\)
- i.e., for all strings \(w\), the DFA and NFA end in the same set of states!

(Same as before)
\textbf{NFA\textarrow{\varepsilon}\textrightarrow DFA}  

\textbf{Proof of Correctness}

Let \( N = (Q_N, \Sigma, \delta_N, q_0, F_N) \)

And let \( \text{NFA\textarrow{\varepsilon}\textrightarrow DFA}(N) = D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D) \)

\textbf{Theorem:} \( \hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w) \)

\textbf{Proof:} (by induction on length of \( w \))

\begin{itemize}
  \item \textbf{Base case} \( w = \epsilon \) \( \hat{\delta}_D(\{q_0\}, \epsilon) \) and \( \hat{\delta}_N(q_0, \epsilon) \)
  \item \textbf{Inductive case} \( w = xa \) \( \; a = \text{last char} \)
    \begin{itemize}
      \item \textbf{IH:} \( \hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x) \), call this set of states \( R \)
      \item NFA last step (from \( \delta_N \) definition) \( \bigcup_{r \in R} \delta_N(r, a) \)
      \item DFA last step (from NFA\textarrow{\varepsilon}\textrightarrow DFA definition) \( \bigcup_{r \in R} \delta_N(r, a) \)
    \end{itemize}
\end{itemize}
NFA→DFA  Proof of Correctness

Let \[ N = (Q_N, \Sigma, \delta_N, q_0, F_N) \]
And let \[ \text{NFA→DFA}(N) = D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D) \]

**Correctness criteria:** \( \text{LANGUAGEOF}(N) = \text{LANGUAGEOF}(D) \)
- i.e., for all strings \( w \), \( N \) accepts \( w \) **if and only if** \( D \) accepts \( w \)

- We will first prove a **stronger** statement: \[ \hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w) \]
  - i.e., for all strings \( w \), the DFA and NFA end in the same set of states!
Proving that NFAs Recognize Reg Langs

**Theorem:**

A language $A$ is regular if and only if some NFA $N$ recognizes it.

**Proof:**

$\Rightarrow$ If $A$ is regular, then some NFA $N$ recognizes it

- **We know:** If $A$ is regular, then a DFA recognizes it
- So convert that DFA to an NFA

$\Leftarrow$ If an NFA $N$ recognizes $A$, then $A$ is regular

- **We know:** A language is regular if there is a DFA recognizing it
- So convert NFA to DFA ...  
- ... Using NFA$\Rightarrow$DFA algorithm we just defined! $\blacksquare$ (Q.E.D.)

I.e., NFAs also represent regular languages!
Last Time: Combining DFAs

Problem 1: What should be the transition labels?

$M_3$: OR

$q_0$

$M_1$: Check special chars

$M_2$: Check uppercase

Problem 2: Once we enter one of the machines, can’t go back to the other one!

We need a different

This is an NFA! Can be in multiple states, and is still equivalent to a regular language!

Why is this so important!?"
Password Requirements

- Passwords must have a minimum length of ten (10) characters - but more is better!
- Passwords **must include at least 3** different types of characters:
  - upper-case letters (A-Z)
  - lower-case letters (a-z)
  - symbols or special characters (%, &, *, $, etc.)
  - numbers (0-9)
- Passwords cannot contain all or part of your email address
- Passwords cannot be re-used
Review: “Closed” Operations

- Natural numbers = \{0, 1, 2, \ldots\}
  - Closed under addition: if x and y are Natural, then \( z = x + y \) is a Nat
  - Closed under multiplication? yes
  - Closed under subtraction? no
- Integers = \{..., -2, -1, 0, 1, 2, \ldots\}
  - Closed under addition and multiplication
  - Closed under subtraction? yes
  - Closed under division? no
- Rational numbers = \{x \mid x = y/z, y and z are ints\}
  - Closed under division? No?
    - Yes if \( z \neq 0 \)

A set is **closed** under an operation if ... applying it to members of the set returns a member in the set
Why Care About Closed Operations?

- Because it allows \textit{repeatedly} applying an operation to a set

- E.g., Closed operations on regular languages preserves “regularness”

- So result of combining DFAs/NFAs can be combined \textit{again and again}
Operations on Regular Languages

Let $A$ and $B$ be languages. We define the regular operations \textit{union}, \textit{concatenation}, and \textit{star} as follows:

- **Union:** $A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$.
- **Concatenation:** $A \circ B = \{ xy \mid x \in A \text{ and } y \in B \}$.
- **Star:** $A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \}$. 
Union 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

$$A \cup B = \{\text{good, bad, boy, girl}\}$$
Union is Closed for Regular Languages

**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/NFA recognizing it!
- Create machine combining the machines recognizing $A_1$ and $A_2$
  - Should we create a DFA or NFA?
Union is Closed for Regular Languages

Add new start state, and ε-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 $F = F_1 \cup F_2$.
4. Define $\delta$ so that for any $q \in Q$ and any $a \in \Sigma$,

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

   $$\delta(q, a) = \begin{cases} 
   ? & \text{if } q = q_0 \\
   ? & \text{if } q \in Q_1 \\
   ? & \text{if } q \in Q_2 \\
   ? & \text{otherwise}
   \end{cases}$$
Another operation: Concatenation

• Example: Matching street addresses

212 Beacon Street

$M_3$: CONCAT

$M_1$: recognize numbers

$M_2$: recognize words
Concatenation 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

$$A \circ B = \{\text{goodboy, goodgirl, badboy, badgirl}\}$$
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? (like union)
- How does it know when to switch from $N_1$ to $N_2$?
- Can only read input once
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 string” (ie, 0 length str) = transition that reads no input

Enables $N$ to run input on two machines that are at different input positions

$N$ must simultaneously:
- Keep checking with $N_1$ and
- Move 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 $N = (Q, \Sigma, \delta, q_1, F)$ 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 $a \in \Sigma$, 
   \[
   \delta(q, a) = \begin{cases} 
   \delta_1(q, a) & q \in Q_1 \text{ and } q \not\in F_1 \\
   \delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\
   \delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\
   \delta_2(q, a) & q \in Q_2.
   \end{cases}
   \]
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 $N = (Q, \Sigma, \delta, q_1, F_2)$ 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 $a \in \Sigma_e$,

\[
\delta(q, a) = \begin{cases} 
? & a \in \Sigma_e \\
? & a \notin \Sigma_e 
\end{cases}
\]
(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

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

(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 $A_1^\infty$.

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 $a \in \Sigma_e$,

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

**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 $a \in \Sigma_e$,

$$
\delta(q, a) = \begin{cases} 
? & a = \varepsilon \\
? & \text{for any } a \in \Sigma_e \\
? & a \neq \varepsilon 
\end{cases}
$$

Kleene star of a language must accept the empty string!
Many More Closed Operations on Regular Languages!

- Complement
- Intersection
- Difference
- Reversal
- Homomorphism
- (See HW2)
Next Time: Regular Expressions
In-class quiz 9/20

See Gradescope