Quote from a student in an upper division CSE class:

“Discussion is really helpful as sometimes when you get lost in other classes, you are lost for the rest of the lecture. Discussion and clicker questions help make students realize when they are getting confused before it is too late and the discussions with classmates help us get back on track.”
Today’s agenda

• More Closure properties of the regular languages

• Introduction to Nondeterministic Finite Automata (NFA’s)

Announcement:
1-1 Sign up sessions with Tutors for extra help (not new HW)

Reminders:
Reading Quiz 2: Due Wed. Apr 6, by 11:59 pm
HW 2: Due Friday, Apr 8, by 11:59 pm
Guaranteed to be on exams

Closure Proofs Revisited
Thm 1.25 The class of regular languages is closed under the union operation.

- **Proof:**
  
- **Given:** Two regular languages $L_1$, $L_2$.
  
- **Want to show:** $L_1 \cup L_2$ is regular.

Because $L_1$ and $L_2$ are regular, we know there exist DFAs

$M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and

$M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$

that recognize $L_1$ and $L_2$.

We construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$, s.t.:
Thm 1.25 The class of regular languages is closed under the union operation.

• Proof continued:
• Given: Two regular languages $L_1$, $L_2$.
• Want to show: $L_1 \cup L_2$ is regular.
• Because $L_1$ and $L_2$ are regular, we know there exist DFAs $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ that recognize $L_1$ and $L_2$.

We construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$, s.t.:
• $Q = Q_1 \times Q_2$
• $\delta(((x,y),c) = (\delta_1(x,c), \delta_2(y,c))$, for $c$ in $\Sigma$ and $(x,y)$ in $Q$
• $q_0 = (q_{01}, q_{02})$
• $F = \{ (x,y) \in Q \mid x \text{ in } F_1 \text{ or } y \text{ in } F_2 \}$
Thm 1.25 The class of regular languages is closed under the union operation.

- Proof Continued:

**Correctness:** We show $M$ recognizes $L_1 \cup L_2$. Let $x$ be a string whose computation in $M$ ends in state $(q,r)$.

If $x$ is *accepted* by $M$, then $q$ is in $F_1$ or $r$ is in $F_2$ by definition of $F$. Therefore $x$ is accepted by $M_1$ or $M_2$, so $x$ is in $L_1 \cup L_2$.

If $x$ is in $L_1 \cup L_2$, then $x$ is accepted by $M_1$ or $M_2$. Thus either $q$ is in $F_1$ or $r$ is in $F_2$. But then $(q,r)$ is in $F$, by definition of $F$, and therefore $x$ is accepted by $M$.

Therefore $x$ is accepted by $M$ IFF $x$ is in $L_1 \cup L_2$.

- **Conclusion:** There is a DFA $M$ that recognizes $L_1 \cup L_2$, so $L_1 \cup L_2$ is regular, and the class of regular languages is closed under union.
Thm 1.25 The class of regular languages is closed under the union operation.

Proof:

Given: Two regular languages $L_1, L_2$.

Want to show: $L_1 \cup L_2$ is regular.

Because $L_1$ and $L_2$ are regular, we know there exist DFAs $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ that recognize $L_1$ and $L_2$. We construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$, s.t.:

- $Q = Q_1 \times Q_2$
- $\delta((x,y),c) = (\delta_1(x,c), \delta_2(y,c))$, for $c \in \Sigma$ and $(x,y) \in Q$
- $q_0 = (q_{01}, q_{02})$
- $F = \{(x,y) \in Q \mid x \in F_1 \text{ or } y \in F_2\}$

Correctness: We show $M$ recognizes $L_1 \cup L_2$. Let $x$ be a string whose computation in $M$ ends in state $(q,r)$. If $x$ is accepted by $M$, then $q$ is in $F_1$ or $r$ is in $F_2$. Therefore $x$ is accepted by $M_1$ or $M_2$, so $x$ is in $L_1 \cup L_2$. If $x$ is not accepted by $M$, then $q$ is not in $F_1$ and $r$ is not in $F_2$, and $x$ is not accepted by either $M_1$ and $M_2$. Therefore $x$ not in $L_1 \cup L_2$.

Conclusion: There is a DFA that recognizes $L_1 \cup L_2$, so $L_1 \cup L_2$ is regular, and the class of regular languages is closed under union.
Thm. The class of regular languages is closed under Intersection

Proof:
Given: Two regular languages \( L_1, L_2 \).
Want to show: \( L_1 \cap L_2 \) is regular.
Because \( L_1 \) and \( L_2 \) are regular, we know there exist DFAs \( M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1) \) and \( M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2) \) that recognize \( L_1 \) and \( L_2 \). We construct a DFA \( M = (Q, \Sigma, \delta, q_0, F) \), s.t.:
- \( Q = Q_1 \times Q_2 \)
- \( \delta((x,y),c) = (\delta_1(x,c), \delta_2(y,c)) \), for \( c \) in \( \Sigma \) and \( (x,y) \) in \( Q \)
- \( q_0 = (s_1, s_2) \)
- \( F = \{ (x,y) \in Q \mid x \in F_1 \text{ and } y \in F_2 \} \)

Correctness: We show that \( M \) recognizes \( L_1 \cap L_2 \). Let \( x \) be a string whose computation in \( M \) ends in state \( (q,r) \). If \( x \) is accepted by \( M \), then \( q \) is in \( F_1 \) and \( r \) is in \( F_2 \). Therefore \( x \) is accepted by \( M_1 \) and \( M_2 \), so \( x \) is in \( L_1 \cap L_2 \). If \( x \) is a string not recognized by \( M \), then either \( q \) is not in \( F_1 \) or \( r \) is not in \( F_2 \). Therefore, \( x \) is not accepted by one of \( M_1 \) and \( M_2 \), and so \( M \) is not in \( L_1 \cap L_2 \).

Conclusion: A DFA recognizes \( L_1 \cap L_2 \), so \( L_1 \cap L_2 \) is regular, and the class of regular languages is closed under intersection.
Thm. The class of regular languages is closed under the union operation.

Proof: (Notice: A regular proof structure!)

What do we know?
L1 and L2 are regular, so by definition, we know there exist DFAs $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ that recognize L1 and L2.

What do we want to show?
L1 U L2 is regular. To be regular, there must be a DFA that recognizes L1 U L2.

We show there is such a DFA:
We construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$, s.t.:

- $Q =$
- $\delta =$
- $q_0 =$
- $F =$

Correctness: Show M recognizes L1 U L2
A DFA recognizes L1 U L2, therefore L1 U L2 is regular. Q.E.D.
Tips for Writing Closure Proofs

A closure proof for regular languages provides an answer to the question, "If I have a regular language, and do [blah] to it, is the new language still regular?"

**GIVEN:** Write down what is known and give name(s) and name their component parts so you can use them later.

**WANT TO SHOW:** Announce what you will prove and your plan for it.

**CONSTRUCTION:**
Let $M' = (Q', \Sigma', \delta', q_0', F')$, where ...

The construction will depend on the problem.

Though *not part of the proof*, a description in English of what you are trying to do is often useful.

**CORRECTNESS:**
Here you prove that your construction actually works.

**CONCLUSION:**
Finish by stating what you have proved.
Example 1

Show that the regular languages are closed under the operation FlipBits, where

\[ \text{FlipBits}(L) = \{w \mid w \text{ is obtained from some } w' \text{ in } L \text{ by flipping each bit in } w' \text{ from 0 to 1, and vice versa}\} \]

**Given**: \(L\) is regular, so recognized by DFA \((Q, \Sigma, \delta, q_0, F)\)

\(\Sigma = \{0, 1\}\)

\(\text{WTS: } \text{FlipBits}(L)\) is regular.

**Construction**: Let \(\text{flip}(0) = 1\), \(\text{flip}(1) = 0\) and for any string \(w = w_1 \ldots w_n\), \(w \in \{0, 1\}^*\)

Let \(\text{flip}(w) = \text{flip}(w_1) \ldots \text{flip}(w_n)\)

Then \(\text{FlipBits}(L) = \{\text{flip}(w) \mid w \in L\} = \{w \mid \text{flip}(w) \in L\}\)

Define \(M' = (Q \Sigma \delta', q_0, F)\) where

\[\delta'(q, a) = \delta(q, \text{flip}(a))\]

**Correctness**: We show \(M'\) recognizes \(\text{FlipBits}(L)\)

\(x = x_1 \ldots x_n \) accepted by \(M' \iff M'\) goes from state \(q_0\) to \(F\) on \(x\)

\(\iff M\) goes from \(q_0\) to \(F\) with \(\text{flip}(x) = \text{flip}(x_1) \ldots \text{flip}(x_n)\)

\(\iff \text{flip}(x) \in L \iff x \in \text{FlipBits}(L)\) (See *)
Example 2

Show that the regular languages are closed under the operation Deletewordswithz, where
Deletewordswithz (L) = \{w | w is in L and w does not contain the letter z\}

Idea: Given DFA M, change all edges labelled "z" to go to a new "dead" state.
Example 3

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes language $B$, swap the accept and non-accept states in $M$ to get a new DFA $M'$.

Claim: The new DFA $M'$ recognizes the complement of $B$.

(Note: the complement of set $B$ is $\Sigma^* - B$).

In HW 2 you will give the full proof that the class of regular languages is closed under complement.

$$M' = (Q, \Sigma, \delta, q_0, Q - F)$$
Example 4

Assume that the Regular Languages are closed under $U$ (union), $\cap$ (intersection), and $-$ (complement). Show that the Regular Languages are also closed under symmetric difference $SD$, where

$$SD(A,B) = (A \setminus B) \cup (B \setminus A)$$

**Proof:**

$A \setminus B = A \cap \overline{B}$ and $B \setminus A = B \cap \overline{A}$.

Since $A, B$ are regular, by closure under $\setminus$, $\overline{A}, \overline{B}$ are regular.

Since $A, B, \overline{A}, \overline{B}$ are regular, by closure under $\cap$,

$A \cap \overline{B}$ and $B \cap \overline{A}$ are regular.

Then by closure under $U$,

$$(A \cap \overline{B}) \cup (B \cap \overline{A}) = SD(A,B)$$ is regular.
Highlights for Regular Language Closure Proofs

• Given a problem in one of the following forms:
  • "If L is a regular language, and we do [blah] to it, is the transformed language still regular?”, and you want to prove that it is regular.
  • Or “Show that the regular languages are closed under [blahblah]”.

• Proofs follow a regular structure:

**GIVEN:** Write down what is known and give names to each of them and their component parts so you can use them later.

**WANT TO SHOW:**
Announce what you will prove and your plan for it.

• You will have to show the new language is regular
  by *either*

• **Constructing** a new $M' = (Q', \Sigma', \delta', q_0', F')$, where $M'$ is constructed from what is given, and show it is correct

• **Applying previously proved closure theorems** for regular languages, if allowed and applicable, and show that the resulting language is the one you wanted to show is regular
Example 5:

Given a language $L$ with alphabet $\Sigma$, and a 1-1, onto function $h: \Sigma \rightarrow \Sigma'$.

$h$ is a symbol-by-symbol translation of $\Sigma$ to $\Sigma'$

We extend $h$ to strings:

$h(a_1...a_k) = h(a_1)...h(a_k) \quad w = a_1...a_k \text{ in } \Sigma^*$

TO SHOW: The regular languages are closed under $h$, where $h(L) = \{ h(w) \mid w \text{ in } L \}$
What about other regular operations?

$L_1 \circ L_2 = \{ xy \mid x \text{ in } L_1 \text{ and } y \text{ in } L_2 \}$

Can we prove closure?

If we start with DFA $M_1$ recognizing $L_1$, and DFA $M_2$ recognizing $L_2$, we have to design DFA to break the input string into two pieces $x$ and $y$ ($x$ for $L_1$ and $y$ for $L_2$)

*But we don’t know how to break it—a guess?*

And * operation is even harder!
To solve this, we introduce NFA’s, a useful extension of DFA’s.
A new model of computation: Nondeterministic Finite Automata

NFA

They’re really good guessers!
And ... provably equivalent to DFA’s
Formal Definition of an NFA

- An NFA $M$ is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:
  - $Q$ is a finite set of states
  - $\Sigma$ is a finite set of characters, the alphabet
  - $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$, the transition function
  - $q_0$, a member of $Q$, the start state
  - $F$, a subset of $Q$, the accept state(s)

**NFA:**

*Allows $\delta(q,x)$ to have more than 1 next state*

*Allows $\varepsilon$ edge to be taken “spontaneously”, without reading input symbol*
NFA or DFA?

NFA....Why?
Informally, on given string $w$:

- Follow any possible edges while reading (consuming) symbol
- At any time, optionally follow $\varepsilon$ edges, without consuming any input symbols
- Accept string $w$ if there is some allowed sequence of transitions that leads to a final state

Is the string 10 accepted?

A. Yes
B. No
DFA’s vs. NFA’s

Deterministic: only 1 edge for a

Nondeterministic: multiple edges for a, possibly edges for empty string $\varepsilon$

Computation trace:

$$q_0 \xrightarrow{a} q_1$$

Path

$q_0$

$q_1$

$q_f$

Tree of all possible paths
Which of the following strings does the NFA N accept?

A. 1
B. 0
C. 01
D. 100
E. More than one of the above

What is $L(N)$?

$= \{ w |$ w ends in 0, $w \in \{0,1\}^* \}$