CS103 Review Session
Studying for the Final
General strategies

- Write out everything you know and what you’re trying to prove.
- What is the quantifier on the statement you’re trying to prove? What does that tell you about how the proof should be set up?
- What kind of structure are you trying to reason about? (binary relations, sets, functions, etc.) You know how to write proofs about all of these! Use proof templates and formal definitions to guide you.
- What proof strategy are you using? What do you get to assume? If you’re doing an indirect proof, would it be helpful to write out the statement in FOL and negate it?
General strategies

- Make sure you’re using all parts of what’s given to you! Usually there’s a good reason why you need each assumption/condition to get the proof to work.
- Draw pictures! Work backwards! Try a different proof strategy! It’s okay if the first thing you try doesn’t work, just try something else!
Set Theory

- Union
- Intersection
- Difference
- Symmetric difference
- Subset
- Power set
Proofs about sets

To show $A \subseteq B$:

- Pick an arbitrary $x \in A$
- Show that $x \in B$

To show $A = B$:

- Prove $A \subseteq B$
- Prove $B \subseteq A$
Example set theory proof

Let A and B be arbitrary sets. Prove that $A \in p(B)$ if and only if $A \cap B = A$. 
Let $A$ and $B$ be arbitrary sets. Prove that $A \in \mathcal{P}(B)$ if and only if $A \cap B = A$.

**What is $A$?**

**What is $B$?**

**Proof 1:** We will prove both directions of implication. First, we'll prove that if $A \in \mathcal{P}(B)$, then $A \cap B = A$. To do so, we'll prove both $A \cap B \subseteq A$ and $A \subseteq A \cap B$.

Let's begin by showing that $A \cap B \subseteq A$. To do so, pick any $x \in A \cap B$. This means in that $x \in A$, and since our choice of $x$ was arbitrary, we conclude that $A \cap B \subseteq A$, as needed.

Next, we'll show that $A \subseteq A \cap B$. Consider any $x \in A$. We will prove that $x \in A \cap B$. We know $A \in \mathcal{P}(B)$, which means that $A \subseteq B$. Since $x \in A$ and $A \subseteq B$, we see that $x \in B$. Then, since $x \in A$ and $x \in B$, we see that $x \in A \cap B$, as required.

For the other direction of implication, assume that $A \cap B = A$. We will prove that $A \in \mathcal{P}(B)$. To do so, we will prove that $A \subseteq B$. So pick any $x \in A$. Then since $x \in A$ and $A = A \cap B$, we see that $x \in A \cap B$. Therefore, we see that $x \in B$. Since our choice of $x \in A$ was arbitrary, we see that $A \subseteq B$, as required. $\blacksquare$
Types of proofs

- **Universal statements: “for all x…”**
  - Proof: Pick an arbitrary x, and show that the statement is true
  - Disproof: find a counter-example

- **Existential statements: “there is an x such that…”**
  - Proof: Find an example
  - Disproof: Pick an arbitrary x and show that the statement is false

- **Implications “P→Q”**
  - **Directly**: assume P and prove Q
  - **By contrapositive** (!Q → !P): assume !Q and prove !P

- **Proof by contradiction:**
  - Assume !P, arrive at a contradiction
First Order Logic

- Can help unpack or take the negation of statements we are trying to prove
- “All P’s are Q’s”: $\forall x. P(x) \rightarrow Q(x)$
- “No P’s are Q’s”: $\forall x. P(x) \rightarrow \neg Q(x)$
- “Some P’s are Q’s”: $\exists x. P(x) \land Q(x)$
- “Some P’s are not Q’s”: $\exists x. P(x) \land \neg Q(x)$
- “For any choice of $x$, there is some $y$ such that $P(x,y)$ is true”: $\forall x \exists y. P(x,y)$
- “There is some $x$ where for any choice of $y$, $P(x,y)$ is true”: $\exists x \forall y. P(x,y)$
First Order Logic

- \( \forall \) is usually paired with \( \rightarrow \)
- \( \exists \) is usually paired with \( \land \)
- Existential statements are false unless there is a positive example
- Universal statements are true unless there is a counter example
Binary Relations

- Reflexive
  - $\forall a \in A. \ a R a$

- Symmetric
  - $\forall a \in A. \ \forall b \in A. \ a R b \rightarrow b R a$

- Transitive
  - $\forall a \in A. \ \forall b \in A. \ \forall c \in A. \ a R b \land b R c \rightarrow a R c$

- Irreflexive
  - $\forall a \in A. \ a \not\in R a$

- Asymmetric
  - $\forall a \in A. \ \forall b \in A. \ a R b \rightarrow \neg b R a$
Binary Relations

- Reflexive
  - $\forall a \in A. \ a \ R \ a$
  - Proof setup: pick an $a \in A$. Show $aRa$.

- Symmetric
  - $\forall a \in A. \ \forall b \in A. \ aRb \rightarrow bRa$
  - Proof setup: pick an $a \in A$ and $b \in A$ such that $aRb$. Show $bRa$.

- Transitive
  - $\forall a \in A. \ \forall b \in A. \ \forall c \in A. \ aRb \land bRc \rightarrow aRc$
  - Proof setup: pick an $a,b,c \in A$ such that $aRb \land bRc$. Show that $aRc$. 
Binary Relations

- **Irreflexive**
  - $\forall a \in A. \neg R a$
  - **Proof setup**: pick an $a \in A$. Show $a \not\in R a$.

- **Asymmetric**
  - $\forall a \in A. \forall b \in A. a R b \rightarrow \neg b \in R a$
  - **Proof setup**: pick an $a \in A$ and $b \in A$ such that $a R b$. Show $b \not\in R a$. 
Binary Relations

- Equivalence Relations
  - Reflexive, symmetric, and transitive
- StrictOrders
  - Irreflexive, asymmetric, and transitive
  - OR equivalently, irreflexive and transitive
  - OR equivalently, asymmetric and transitive
Example binary relations proof

If \( R_1 \) is a binary relation over a set \( A_1 \) and \( R_2 \) is a binary relation over a set \( A_2 \), then an embedding of \( R_1 \) in \( R_2 \) is a function \( f : A_1 \to A_2 \) such that

\[
\forall a \in A_1. \ \forall b \in A_1. \ (aR_1 b \leftrightarrow f(a) R_2 f(b)).
\]

If there’s an embedding of a relation \( R_1 \) in a relation \( R_2 \), we say that \( R_1 \) can be embedded in \( R_2 \).

Let \( R_1 \) be a binary relation over a set \( A_1 \) and let \( R_2 \) be a strict order over some set \( A_2 \). Prove that if \( R_1 \) can be embedded in \( R_2 \), then \( R_1 \) is a strict order.

**IRREFLEXIVE**

\[
\forall a \in A. \ aRa.
\]

No element is related to itself.

**Proof setup:**

Pick an \( a \in A \). Prove \( \neg aRa \).

What set is the relation defined over?

*(Where should we be picking our arbitrary element from?)*
Example binary relations proof

Proof 1: Let $f : A_1 \rightarrow A_2$ be an embedding of $R_1$ in $R_2$. We will show that $R_1$ is a strict order by proving that it is irreflexive and transitive.

First, we'll show that $R_1$ is irreflexive. Consider any $a \in A_1$. Since $R_2$ is a strict order, we know that $R_2$ is irreflexive, so $f(a)R_2f(a)$. Then, since $f$ is an embedding of $R_1$ in $R_2$, we see that $aR_1a$, as required.

Next, we'll show that $R_1$ is transitive. To do so, consider any $a, b, c \in A_1$ where $aR_1b$ and $bR_1c$. Since $f$ is an embedding of $R_1$ in $R_2$, we then see that $f(a)R_2f(b)$ and $f(b)R_2f(c)$. Then, since $R_2$ is a strict order, it's transitive, and so $f(a)R_2f(c)$. Finally, since $f$ is an embedding of $R_1$ in $R_2$, we use the reverse direction of the implication to conclude that $aR_1c$, as required. ■
Functions

● All functions: $f: A \rightarrow B$
  ○ Every input maps to some output
    ■ For all $a$ in $A$, there exists $b$ in $B$ such that $f(a) = b$.
  ○ Functions are deterministic: equal inputs produce equal outputs
    ■ For all $a_1, a_2$ in $A$ if $a_1 = a_2$, then $f(a_1) = f(a_2)$.
Functions

- **Injective functions**
  - Different inputs produce different outputs
  - For all \( a_1, a_2 \) in \( A \), \( a_1 \neq a_2 \) → \( f(a_1) \neq f(a_2) \).
  - For all \( a_1, a_2 \) in \( A \), \( f(a_1) = f(a_2) \) → \( a_1 = a_2 \).
Functions

- **Surjective functions**
  - For every possible output, there exists at least one possible input that produces it.
  - For all $b$ in $B$, there exists an $a$ in $A$ such that $f(a) = b$. 

![Diagram of surjection and not a surjection]
Functions

- Bijective functions
  - Functions that are both injective and surjective
Example of function proof

Imagine you have a function \( f : A \rightarrow B \) from some set \( A \) to some set \( B \). We can use \( f \) to construct a new function called the \textit{lift of} \( f \), denoted \( \text{lift}_f \), from \( \wp(A) \) to \( \wp(B) \). Specifically \( \text{lift}_f : \wp(A) \rightarrow \wp(B) \) is defined as follows:

\[
\text{lift}_f(S) = \{ y \mid \exists x \in S. f(x) = y \}
\]

Let \( A \) and \( B \) be sets. Prove that if \( f : A \rightarrow B \) is injective, then \( \text{lift}_f \) is injective.

What function are we trying to prove things about?
What is the domain of that function?
Example of function proof

**Proof 1:** Let \( f : A \to B \) be an injective function. We will prove that \( \text{lift}_f \) is injective as well. To do so, consider any \( S_1, S_2 \in \varnothing(A) \) where \( S_1 \neq S_2 \). We will prove that \( \text{lift}_f(S_1) \neq \text{lift}_f(S_2) \).

Since \( S_1 \neq S_2 \), we know that either \( S_1 \not\subseteq S_2 \) or that \( S_2 \not\subseteq S_1 \). Without loss of generality, assume \( S_1 \not\subseteq S_2 \), which means that there is some \( a \in S_1 \) where \( a \not\in S_2 \).

First, notice that since \( a \in S_1 \), we see that \( f(a) \in \text{lift}_f(S_1) \). We now claim that \( f(a) \not\in \text{lift}_f(S_2) \). To see this, suppose for the sake of contradiction that \( f(a) \in S_2 \). This means that there must be some \( a' \in S_2 \) such that \( f(a') = f(a) \). Since \( f \) is injective, that tells us that \( a' = a \), and since \( a' \in S_2 \), we see that \( a \in S_2 \) as well. But this is impossible, since we know that \( a \not\in S_2 \). We’ve reached a contradiction, so our assumption was wrong and \( f(a) \not\in \text{lift}_f(S_2) \).

Since \( f(a) \in \text{lift}_f(S_1) \) but \( f(a) \not\in \text{lift}_f(S_2) \), we see \( \text{lift}_f(S_1) \neq \text{lift}_f(S_2) \), which is what we needed to show. \( \blacksquare \)
5 minute break!
The Pigeonhole Principle
Pigeonhole Principle Refresher

- The **generalized pigeonhole principle** says that if you distribute $m$ objects into $n$ bins, then
  - some bin will have at least $\lceil \frac{m}{n} \rceil$ objects in it, and
  - some bin will have at most $\lfloor \frac{m}{n} \rfloor$ objects in it.

$\lceil \frac{m}{n} \rceil$ means “$m/n$, rounded up.”
$\lfloor \frac{m}{n} \rfloor$ means “$m/n$, rounded down.”
Pigeonhole Principle Clues + Tips

1) Look for "at most", "at least", "less than", or "more than" in the problem statement. It's not a guarantee but often pigeonhole principle problems use these terms.
Pigeonhole Principle Clues + Tips

2) Try writing out all the nouns mentioned in the problem statement and their quantity (if known).

Let’s begin with some new definitions. First, we’ll say that a matching in a graph \( G = (V, E) \) is a set \( M \subseteq E \) of edges in \( G \) such that no two edges in \( M \) share an endpoint. The size of a matching is the number of edges it contains. The matching number of a graph \( G \), denoted \( v(G) \), is the size of the largest matching in \( G \).

Now, let’s introduce a variation on a definition we’ve seen before. A \( k \)-edge coloring of a graph \( G = (V, E) \) is a way of coloring each of the edges in \( G \) one of \( k \) different colors so that no two edges that share an endpoint are assigned the same color. The chromatic index of a graph \( G \), denoted \( \chi'_k(G) \), is the minimum number of colors needed in any edge coloring of \( G \).

Let \( G \) be an undirected graph with exactly \( n^2 + 1 \) edges for some natural number \( n \geq 1 \). Prove that either \( \chi'_k(G) \geq n+1 \) or \( v(G) \geq n+1 \) (or both).

Most of the time:

There are more pigeons than holes
There is more than one hole
We know how many holes and pigeons there are
(otherwise the result isn't very interesting...)
2) Try writing out all the nouns mentioned in the problem statement and their quantity (if known).

Let's begin with some new definitions. First, we'll say that a **matching** in a graph $G = (V, E)$ is a set $M \subseteq E$ of edges in $G$ such that no two edges in $M$ share an endpoint. The **size** of a matching is the number of edges it contains. The **matching number** of a graph $G$, denoted $\nu(G)$, is the size of the largest matching in $G$.

Now, let's introduce a variation on a definition we've seen before. A **$k$-edge coloring** of a graph $G = (V, E)$ is a way of coloring each of the edges in $G$ one of $k$ different colors so that no two edges that share an endpoint are assigned the same color. The **chromatic index** of a graph $G$, denoted $\chi'_1(G)$, is the minimum number of colors needed in any edge coloring of $G$.

Let $G$ be an undirected graph with exactly $n^2 + 1$ edges for some natural number $n \geq 1$. Prove that either $\chi'_1(G) \geq n+1$ or $\nu(G) \geq n+1$ (or both).

Most of the time:

- There are more pigeons than holes
- There is more than one hole
- We know how many holes and pigeons there are
- (otherwise the result isn't very interesting...)

### Pigeonhole Principle Clues + Tips

- **Graph** (1)  
- **Vertices** (?)
- **Edges** $(n^2 + 1)$
- **Edges in largest matching** (at most $n^2 + 1$)
- **Colors** (at most $n^2 + 1$)

**maybe the pigeons?!**  
**maybe the holes?!**
How do you prove a statement of the form $P$ or $Q$?
Example Pigeonhole Principle Proof

Let's begin with some new definitions. First, we'll say that a matching in a graph $G = (V, E)$ is a set $M \subseteq E$ of edges in $G$ such that no two edges in $M$ share an endpoint. The size of a matching is the number of edges it contains. The matching number of a graph $G$, denoted $\nu(G)$, is the size of the largest matching in $G$.

Now, let's introduce a variation on a definition we've seen before. A $k$-edge coloring of a graph $G = (V, E)$ is a way of coloring each of the edges in $G$ one of $k$ different colors so that no two edges that share an endpoint are assigned the same color. The chromatic index of a graph $G$, denoted $\chi_1(G)$, is the minimum number of colors needed in any edge coloring of $G$.

Let $G$ be an undirected graph with exactly $n^2 + 1$ edges for some natural number $n \geq 1$. Prove that either $\chi_1(G) \geq n+1$ or $\nu(G) \geq n+1$ (or both).

**Proof:** Let $G$ be an arbitrary undirected graph with $n^2 + 1$ edges for some positive natural number $n$. We will prove that if $\chi_1(G) \leq n$, then $\nu(G) \geq n+1$.

Suppose that $\chi_1(G) \leq n$. This means that there is an $n$-edge coloring of the graph $G$. Since there are $n^2 + 1$ edges and $n$ colors, by the generalized pigeonhole principle we know that there must be at least $\lceil (n^2 + 1) / n \rceil = \lceil n + 1/n \rceil = n + 1$ edges that are all the same color in the $n$-edge coloring. Since all those edges are assigned the same color, we know that no two of them can share an endpoint. Therefore, this set of $n+1$ edges forms a matching, so $\nu(G) \geq n+1$, as required. ■
Induction
Let $P$ be some predicate. The *principle of mathematical induction* states that if

- $P(0)$ is true
- and it stays true...
- $\forall k \in \mathbb{N}. (P(k) \rightarrow P(k+1))$

then

$\forall n \in \mathbb{N}. P(n)$

...then it's always true.
Complete vs Regular Induction

- Use regular induction if you can
- Use complete if you need more than just $P(k)$ when proving $P(k+1)$

Regular induction

- Assume $P(k)$
- Prove base case
- Prove $P(k+1)$

Complete induction

- Assume $P(0), P(1)...P(k)$

Graphical representation of the induction process.
Induction Clues + Tips

1) Look for "all natural numbers n" in the problem statement. Pretty much every induction problem uses that phrase (but there are non-induction problems that do, too!).

Let’s begin with a refresher of the key terms and definitions involved. As a reminder, if \( L_1 \) and \( L_2 \) are languages over an alphabet \( \Sigma \), then the **concatenation of \( L_1 \) and \( L_2 \)**, denoted \( L_1L_2 \), is the language

\[
L_1L_2 = \{ wx \mid w \in L_1 \text{ and } x \in L_2 \}.
\]

From concatenation, we can define **language exponentiation** of a language \( L \) inductively as follows:

\[
L^0 = \{ \varepsilon \} \quad \quad \quad L^{n+1} = LL^n
\]

You may find these formal terms helpful in the course of solving this problem.

Let \( A \) and \( B \) be arbitrary languages over some alphabet \( \Sigma \). Prove, by induction, that if \( X = AX \cup B \), then \( A^nB \subseteq X \) for every \( n \in \mathbb{N} \). Please use the formal definitions of concatenation, language exponentiation, union, and subset in the course of writing up your answer.
Induction Clues + Tips

2) Look for a link between smaller and larger problems (recursion!).

Let's begin with a refresher of the key terms and definitions involved. As a reminder, if $L_1$ and $L_2$ are languages over an alphabet $\Sigma$, then the **concatenation of $L_1$ and $L_2$**, denoted $L_1L_2$, is the language

$$L_1L_2 = \{ wx \mid w \in L_1 \text{ and } x \in L_2 \}.$$  

From concatenation, we can define **language exponentiation** of a language $L$ inductively as follows:

$$L^0 = \{ \varepsilon \} \quad \quad L^{n+1} = LL^n$$

You may find these formal terms helpful in the course of solving this problem.

Let $A$ and $B$ be arbitrary languages over some alphabet $\Sigma$. Prove, by induction, that if $X = AX \cup B$, then $A^nB \subseteq X$ for every $n \in \mathbb{N}$. Please use the formal definitions of concatenation, language exponentiation, union, and subset in the course of writing up your answer.
Induction Clues + Tips

3) Think about **building up** for existential $P(n)$ and **building down** for universal $P(n)$.

Let's begin with a refresher of the key terms and definitions involved. As a reminder, if $L_1$ and $L_2$ are languages over an alphabet $\Sigma$, then the **concatenation of $L_1$ and $L_2$**, denoted $L_1L_2$, is the language

$$L_1L_2 = \{ wx \mid w \in L_1 \text{ and } x \in L_2 \}.$$  

From concatenation, we can define **language exponentiation** of a language $L$ inductively as follows:

$$L^0 = \{ \varepsilon \} \quad \quad L^{n+1} = LL^n$$

You may find these formal terms helpful in the course of solving this problem.

Let $A$ and $B$ be arbitrary languages over some alphabet $\Sigma$. Prove, by induction, that if $X = AX \cup B$, then $A^nB \subseteq X$ for every $n \in \mathbb{N}$. Please use the formal definitions of concatenation, language exponentiation, union, and subset in the course of writing up your answer.
Induction Clues + Tips

4) Write down P(n) and make sure:
   ● P(n) will allow you to prove your end goal.
   ● The definition of P(n) includes n (not all natural numbers n)

Let's begin with a refresher of the key terms and definitions involved. As a reminder, if \( L_1 \) and \( L_2 \) are languages over an alphabet \( \Sigma \), then the **concatenation of** \( L_1 \) **and** \( L_2 \), denoted \( L_1L_2 \), is the language

\[
L_1L_2 = \{ wx | w \in L_1 \text{ and } x \in L_2 \}.
\]

From concatenation, we can define **language exponentiation** of a language \( L \) inductively as follows:

\[
L^0 = \{ \varepsilon \} \quad \quad L^{n+1} = LL^n
\]

You may find these formal terms helpful in the course of solving this problem.

Let \( A \) and \( B \) be arbitrary languages over some alphabet \( \Sigma \). Prove, by induction, that if \( X = AX \cup B \), then \( A^nB \subseteq X \) for every \( n \in \mathbb{N} \). Please use the formal definitions of concatenation, language exponentiation, union, and subset in the course of writing up your answer.
Let's begin with a refresher of the key terms and definitions involved. As a reminder, if $L_1$ and $L_2$ are languages over an alphabet $\Sigma$, then the **concatenation of $L_1$ and $L_2$**, denoted $L_1L_2$, is the language

$$L_1L_2 = \{ wx \mid w \in L_1 \text{ and } x \in L_2 \}.$$  

From concatenation, we can define **language exponentiation** of a language $L$ inductively as follows:

$$L^0 = \{ \varepsilon \} \quad \quad L^{n+1} = LL^n$$

You may find these formal terms helpful in the course of solving this problem.

Let $A$ and $B$ be arbitrary languages over some alphabet $\Sigma$. Prove, by induction, that if $X = AX \cup B$, then $A^nB \subseteq X$ for every $n \in \mathbb{N}$. Please use the formal definitions of concatenation, language exponentiation, union, and subset in the course of writing up your answer.
Example Induction Proof

Proof: Let $A$ and $B$ be arbitrary languages over some alphabet $\Sigma$ where $X = AX \cup B$. Let $P(n)$ be the statement “$A^nB \subseteq X$.” We will prove by induction that $P(n)$ is true for all $n \in \mathbb{N}$, from which the theorem follows.

As our base case, we prove $P(0)$, that $A^0B \subseteq X$. Consider any $w \in A^0B$. This string must be of the form $xy$ where $x \in A^0$ and $y \in B$. Since the only string in $A^0$ is $\varepsilon$, this means that $w = \varepsilon y = y$, so $w \in B$. Then, since $w \in B$, we know that $w \in AX \cup B$, and therefore that $w \in X$. Since our choice of $w$ was arbitrary, this shows that every element of $A^0B$ is an element of $X$, so $A^0B \subseteq X$, as required.

For our inductive step, assume for some arbitrary $k \in \mathbb{N}$ that $P(k)$ holds and that $A^kB \subseteq X$. We will prove that $A^{k+1}B \subseteq X$. To do so, consider any arbitrary $w \in A^{k+1}B$. We will prove that $w \in X$.

Since $A^{k+1}B = AA^kB = A(A^kB)$, we know see that $w \in A(A^kB)$. Consequently, there exist some $x \in A$ and $y \in A^kB$ such that $w = xy$. Since $y \in A^kB$, by our inductive hypothesis we see that $y \in X$. Overall, this shows that $w = xy$ where $x \in A$ and $y \in X$, so we see that $w \in AX$. Since $w \in AX$, we see that $w \in AX \cup B$, or equivalently that $w \in X$, as required. Thus $P(k+1)$ is true, completing the induction. ■
Regular Languages
Regular Languages

DFAs

NFAs

RegExs

(a \cup b)^*aa(a \cup b)^*
Designing a DFA

states = pieces of information
transitions = when I read in a new character, how might this change what I know?
Designing an NFA

**states** = pieces of information

**transitions** = when I read in a new character, how might this change what I know?

AND

**nondeterminism** = assume you'll magically "know" when it's time to take the right transition
DFA Construction Example

Let $\Sigma = \{a, b\}$ and let $L_1 = \{ w \in \Sigma^* \mid w$ does not contain $bbb$ as a substring $\}$. Design a DFA for $L_1$.

What are the states? How should my transitions link together the states?
DFA Construction Example

Let $\Sigma = \{ a, b \}$ and let $L_1 = \{ w \in \Sigma^* \mid w$ does not contain bbb as a substring $\}$. Design a DFA for $L_1$.

What are the states? How should my transitions link together the states?

Here is one possible solution:

This automaton works by advancing forward every time it sees a b and resetting whenever it sees an a. If it finds three consecutive b’s, it enters a dead state.
Designing a RegEx

1. Write out example strings and look for patterns
2. Can I separate the strings into different categories?
   a. If yes: UNION the categories together.
3. Can I break the strings into smaller subproblems?
   a. If yes: CONCATENATE each piece together.
4. Is there some sort of repeating structure?
   a. If yes: Use the KLEENE STAR on the smallest repeating pattern.

(a ∪ b)*aa(a ∪ b)*
Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w$ has an odd number of $a$'s$\}$. Write a regular expression for $L$.  

\[
\begin{align*}
  &a \\
  &aaa \\
  &abb \\
  &bab \\
  &bba \\
  &aaa \\
  &aaabb \\
  &bbaaa \\
  &baaab \\
  & \ldots
\end{align*}
\]
RegEx Construction Example

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ has an odd number of } a's\}$. Write a regular expression for $L$.

- We need at least one $a$
- We can have any number of $b$'s, in any position
- We can add $a$'s two at a time to the original $a$
Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* | w \text{ has an odd number of } a\text{'s}\}$. Write a regular expression for $L$. We need at least one $a$. 

\[
\begin{align*}
  \text{a} \\
  \text{aaa} \\
  \text{abb} \\
  \text{bab} \\
  \text{bba} \\
  \text{aaa} \\
  \text{aaabb} \\
  \text{bbaaa} \\
  \text{bbaaa} \\
  \text{baaab} \\
  \ldots
\end{align*}
\]
Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w$ has an odd number of a's$. Write a regular expression for $L$.

Here is one possible solution:

$b^*ab^*$

We need at least one a

We can have any number of b's, in any position

...
Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* | w$ has an odd number of a's\}. Write a regular expression for $L$.

Here is one possible solution:

\[b^*ab^*(b^*ab^*ab^*)^*\]

- We need at least one a
- We can have any number of b's, in any position
- We can add a's two at a time to the original a
RegEx Construction Example

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w$ has an odd number of $a$'s$\}$. Write a regular expression for $L$.

Here is one possible solution:

$$b^*ab^*(b^*ab^*ab^*)^*$$
Myhill-Nerode Theorem
Myhill-Nerode Refresher

**Theorem:** Let $L$ be a language over $\Sigma$. If there is a set $S \subseteq \Sigma^*$ with the following properties, then $L$ is not regular:

- $S$ is infinite (that is, $S$ contains infinitely many strings).
- The strings in $S$ are **pairwise distinguishable relative to $L$**. That is,

\[
\forall x \in S. \forall y \in S. (x \neq y \rightarrow x \not\in_L y).
\]

If you pick any two strings in $S$ that aren’t equal to one another...

... then they’re distinguishable relative to $L$. 
Distinguishability Refresher

- Let $L$ be an arbitrary language over $\Sigma$.
- Two strings $x \in \Sigma^*$ and $y \in \Sigma^*$ are called distinguishable relative to $L$ if there is a string $w \in \Sigma^*$ such that exactly one of $xw$ and $yw$ is in $L$.
- We denote this by writing $x \not\equiv_L y$. 
Myhill-Nerode Clues + Tips

1) Look for "**not a regular language**" in the problem statement. Pretty much every Myhill-Nerode problem involves proving that a language is not regular.

---

Let $\Sigma = \{a, b\}$. Consider the following language $L_2$ over $\Sigma$:

$$L_2 = \{ a^n b^m \mid m, n \in \mathbb{N} \text{ and } m \leq 2n \}$$

For example, $aa \in L_2$ $aab \in L_2$ $aabb \in L_2$, and $aabb \in L_2$, but $aabbbb \notin L_2$.

Prove that $L_2$ is not a regular language.
Myhill-Nerode Clues + Tips

2) Think about what you need to remember in order to prove that a string is in the language. Use that to pick an infinite set $S$. You need to prove that every pair of strings in $S$ is distinguishable relative to $L$.

*The strings in $S$ do not need to be in $L$**

Let $\Sigma = \{a, b\}$. Consider the following language $L_2$ over $\Sigma$: $L_2 = \{ a^m b^n \mid m, n \in \mathbb{N} \text{ and } m \leq 2n \}$

For example, $aa \in L_2$, $aab \notin L_2$, $aaab \in L_2$, $aabb \in L_2$, and $aabb \in L_2$, but $aabb \notin L_2$. Prove that $L_2$ is not a regular language.
Example Myhill-Nerode Proof

Let $\Sigma = \{a, b\}$. Consider the following language $L_2$ over $\Sigma$:

$L_2 = \{a^mb^n \mid m, n \in \mathbb{N} \text{ and } m \leq 2n \}$

For example, $aa \in L_2$ $aab \in L_2$ $aabb \in L_2$, $aabb \in L_2$, and $aabb \in L_2$, but $aabb \notin L_2$.

Prove that $L_2$ is not a regular language.

**Proof:** Let $S = \{a^n \mid n \in \mathbb{N}\}$. The set $S$ is infinite because it contains one string for each natural number. Now, consider any two strings $a^n, a^m \in S$. Without loss of generality, assume that $n < m$. Now, consider the strings $a^n b^{2m}$ and $a^m b^{2m}$. The string $a^n b^{2m}$ is not in $L_2$ because $2m > 2n$, so there are too many $b$'s in the string for it to be in $L_2$. On the other hand, the string $a^m b^{2m}$ is in $L$ because the number of $b$'s is precisely twice the number of $a$'s. Therefore, we see that $a^n \not\equiv_{L_2} a^m$. Since our choices of $a^n$ and $a^m$ were arbitrary, we therefore see that any two distinct strings in $S$ are distinguishable relative to $L_2$. Therefore, since $S$ is infinite, by the Myhill-Nerode theorem we see that $L_2$ is not regular. ■
Designing a CFG

1. Write out example strings and look for patterns
2. Think recursively - look for smaller strings within larger ones
3. "For every x I see, I need y somewhere else" means that x, y need to be added at the same time
4. Non-terminals represent different states/types of strings.

\[ S \rightarrow \epsilon \mid a \mid b \mid aSa \mid bSb \]
CFG Construction Example

Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w \text{ has no a's or no b's}\}$. Write a CFG for $L$.

```
bb
bbbbb
bb
a
aaa
aa
...
```
Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w$ has no a's or no b's$\}$. Write a CFG for $L$.

We can either have any number of a's or any number of b's.
Let $\Sigma = \{a, b\}$ and $L = \{w \in \Sigma^* \mid w$ has either no a's or no b's$\}$. Write a CFG for $L$.

- We start by "choosing" a's or b's.
- After we've chosen, we can add as many of the chosen letter as we want.

Start symbol: $S$

- $S \rightarrow \varepsilon \mid aA \mid bB$
- $A \rightarrow aA \mid \varepsilon$
- $B \rightarrow bB \mid \varepsilon$
Lava Diagram
REG: Problems Solvable with Finite Memory
Are there are finitely many cases to check?

RE: Languages with Verifiers
Given any string $w \in L$, could you prove that $w \in L$?

R: Languages with Deciders
In addition to the RE requirements, given any string $w \notin L$, could you prove that $w \notin L$?

$L_1 = \{ (M) \mid M \text{ is a TM and } |L(M)| \geq 2 \}$
$L_2 = \{ (M) \mid M \text{ is a TM and } |L(M)| = 2 \}$
$L_3 = \{ a^n b^n \mid n \in \mathbb{N} \text{ and } n > 1000 \}$
$L_4 = \{ a^n b^n \mid n \in \mathbb{N} \text{ and } n \leq 1000 \}$

Hopefully, this gives you a good starting point for working through Lava Diagram questions. Good luck!
Lava Diagram

Intuition:
Start in ALL and see if you can move the language in more

REG: Problems Solvable with Finite Memory
Are there are finitely many cases to check?

RE: Languages with Verifiers
Given any string \( w \in L \), could you prove that \( w \in L \)?

\( L_1 = \{ (M) \mid M \text{ is a TM and } |\mathcal{L}(M)| \geq 2 \} \)
\( L_2 = \{ (M) \mid M \text{ is a TM and } |\mathcal{L}(M)| = 2 \} \)
\( L_3 = \{ a^n b^n \mid n \in \mathbb{N} \text{ and } n > 1000 \} \)
\( L_4 = \{ a^n b^n \mid n \in \mathbb{N} \text{ and } n \leq 1000 \} \)

R: Languages with Deciders
In addition to the RE requirements, given any string \( w \notin L \), could you prove that \( w \notin L \)?

Hopefully, this gives you a good starting point for working through Lava Diagram questions. Good luck!
Questions?