Review For The Midterm
Rules

1. **Begin** Midterm ON Gradescope: Tuesday April 7, 6:00PM-9:00PM. (DSS students get an extension)
Rules

1. **Begin** Midterm ON Gradescope: Tuesday April 7, 6:00PM-9:00PM. (DSS students get an extension)

2. **Resources** Midterm is open-everything. The web, my notes, my HW solutions, all fine to use. Cannot ask someone for help. Honor System.

3. **Caveat** You must hand in your own work and you must understand what you hand in.

4. **Warning** Mindlessly copying does not work.
Rules

1. **Begin** Midterm ON Gradescope: Tuesday April 7, 6:00PM-9:00PM. (DSS students get an extension)

2. **Resources** Midterm is open-everything. The web, my notes, my HW solutions, all fine to use. Cannot ask someone for help. Honor System.

3. **Caveat** You must hand in your own work and you must understand what you hand in.

4. **Warning** Mindlessly copying does not work.

5. **Neat** LaTeX is best. Good handwriting okay. Draw Aut, or use LateX tool posted.
1. **Begin** Midterm ON Gradescope: Tuesday April 7, 6:00PM-9:00PM. (DSS students get an extension)

2. **Resources** Midterm is open-everything. The web, my notes, my HW solutions, all fine to use. Cannot ask someone for help. Honor System.

3. **Caveat** You must hand in your own work and you must understand what you hand in.

4. **Warning** Mindlessly copying does not work.

5. **Neat** LaTex is best. Good handwriting okay. Draw Aut, or use LateX tool posted.

6. **Our Intent** This is exam I intended to give out originally. The extra time is meant for you to format and put in LaTeX.
Rules

1. **Begin** Midterm ON Gradescope: Tuesday April 7, 6:00PM-9:00PM. (DSS students get an extension)

2. **Resources** Midterm is open-everything. The web, my notes, my HW solutions, all fine to use. Cannot ask someone for help. Honor System.

3. **Caveat** You must hand in your own work and you must understand what you hand in.

4. **Warning** Mindlessly copying does not work.

5. **Neat** LaTeX is best. Good handwriting okay. Draw Aut, or use LateX tool posted.

6. **Our Intent** This is exam I intended to give out originally. The extra time is meant for you to format and put in LaTeX.

7. **Scope of the Exam**
   - **Short Answer** HWs and lectures.
   - **Long Answer** This Presentation.
What We Have Covered: Regular Languages

1. Examples of Reg Langs
What We Have Covered: Regular Languages

1. **Examples of Reg Langs**
   
   Numbers that are $\equiv i \pmod{j}$
What We Have Covered: Regular Languages

1. Examples of Reg Langs

   Numbers that are \( \equiv i \pmod{j} \)
   \[ \{ w : \#_a(w) \equiv i_1 \pmod{j_1} \wedge \#_b(w) \equiv i_2 \pmod{j_2} \} \]
1. **Examples of Reg Langs**

Numbers that are $\equiv i \pmod{j}$

$\{w: \#_a(w) \equiv i_1 \pmod{j_1} \land \#_b(w) \equiv i_2 \pmod{j_2}\}$

For a fixed string $w$, $w\{a, b\}^*$, $\{a, b\}^*w$
What We Have Covered: Regular Languages

1. Examples of Reg Langs

Numbers that are $\equiv i \pmod{j}$

$\{w : \#_a(w) \equiv i_1 \pmod{j_1} \land \#_b(w) \equiv i_2 \pmod{j_2}\}$

For a fixed string $w$, $w\{a, b\}^*$, $\{a, b\}^*w$

$\{a, b\}^*a\{a, b\}^n$ (DFA requires $\sim 2^n$, NFA $\sim n$. Cool!)
What We Have Covered: Regular Languages

1. Examples of Reg Langs

Numbers that are \( \equiv i \pmod{j} \)
\( \{ w : \#_a(w) \equiv i_1 \pmod{j_1} \land \#_b(w) \equiv i_2 \pmod{j_2} \} \)

For a fixed string \( w \), \( w \{ a, b \}^* \), \( \{ a, b \}^* w \)
\( \{ a, b \}^* a \{ a, b \}^n \) (DFA requires \( \sim 2^n \), NFA \( \sim n \). Cool!)
\( \{ a^i : i \neq n \} \) (DFA requires \( \sim n \), NFA \( \sim 2\sqrt{n} + \log \text{stuff} \). Cool!)
1. **Examples of Reg Langs**

Numbers that are \( \equiv i \pmod{j} \)

\[ \{ w : \#_a(w) \equiv i_1 \pmod{j_1} \land \#_b(w) \equiv i_2 \pmod{j_2} \} \]

For a fixed string \( w \), \( w\{a, b\}^* \), \( \{a, b\}^*w \)

\[ \{a, b\}^*a\{a, b\}^n \text{ (DFA requires } \sim 2^n \text{, NFA } \sim n. \text{ Cool!)} \]

\[ \{a^i : i \neq n\} \text{ (DFA requires } \sim n \text{, NFA } \sim 2\sqrt{n} + \log\text{stuff Cool!)} \]

Others
What We Have Covered: Regular Languages

1. **Examples of Reg Langs**
   
   Numbers that are $\equiv i \pmod{j}$
   
   \[ \{ w : \#_a(w) \equiv i_1 \pmod{j_1} \land \#_b(w) \equiv i_2 \pmod{j_2} \} \]
   
   For a fixed string $w$, $w\{a, b\}^*, \{a, b\}^*w$
   
   \[ \{a, b\}^*a\{a, b\}^n \text{ (DFA requires } \sim 2^n, \text{ NFA } \sim n. \text{ Cool!)} \]
   
   \[ \{a^i : i \neq n\} \text{ (DFA requires } \sim n, \text{ NFA } \sim 2\sqrt{n} + \log\text{stuff Cool!)} \]
   
   Others

2. DFA, NFA, REGEX. Equivalence of all of these.


What We Have Covered: Context Free Languages

1. Examples of CFL’s

2. Chomsky Normal Form. Needed to make size comparisons.


4. Non-CFL’s:
   - If $L \subseteq a^*$ and not regular, than not CFL.
   - If need to keep track of TWO things then NOT CFL. E.g., $\{a^n b^n c^n : n \in \mathbb{N}\}$
What We Have Covered: Context Free Languages

1. Examples of CFL’s
   \[ \{ a^{k_1 n} b^{k_2 n} : n \in \mathbb{N} \} \]
What We Have Covered: Context Free Languages

1. **Examples of CFL’s**
   
   \[ \{ a^{k_1 n} b^{k_2 n} : n \in \mathbb{N} \} \]
   
   \[ \{ w : \#_a(w) = \#_b(w) \} \]
What We Have Covered: Context Free Languages

1. Examples of CFL’s
   \{a^{k_1n}b^{k_2n} : n \in \mathbb{N}\}
   \{w : \#_a(w) = \#_b(w)\}
   \{w : k_1\#_a(w) = k_2\#_b(w)\}
What We Have Covered: Context Free Languages

1. Examples of CFL’s
   \[ \{ a^{k_1n} b^{k_2n} : n \in \mathbb{N} \} \]
   \[ \{ w : \#_a(w) = \#_b(w) \} \]
   \[ \{ w : k_1\#_a(w) = k_2\#_b(w) \} \]
   \[ \{ a^n \} \text{ (Interesting: Small CFL, Large NFA)} \]
What We Have Covered: Context Free Languages

1. **Examples of CFL’s**
   - $\{a^{k_1 n} b^{k_2 n} : n \in \mathbb{N}\}$
   - $\{w : \#_a(w) = \#_b(w)\}$
   - $\{w : k_1 \#_a(w) = k_2 \#_b(w)\}$
   - $\{a^n\}$ (Interesting: Small CFL, Large NFA)

2. **Chomsky Normal Form.** Needed to make size comparisons.

3. **Closure Properties.**

4. **Non-CFL’s:**
   - If $L \subseteq a^*$ and not regular, than not CFL.
   - If need to keep track of TWO things then NOT CFL.
   - E.g., $\{a^n b^n c^n : n \in \mathbb{N}\}$
Equivalence of DFA, NFA, REGEX

1. \(L_{DFA} \rightarrow L_{REGEX}\): Dynamic Programming. \(\alpha\) is in number of states.

2. \(L_{REGEX} \rightarrow L_{NFA}\): induction on formation of a REGEX.

3. \(L_{NFA} \rightarrow L_{DFA}\): powerset construction. States blowup exponentially.
Equivalence of DFA, NFA, REGEX

1. $L_{DFA} \rightarrow L_{REGEX}$:
Equivalence of DFA, NFA, REGEX

1. $L$ DFA $\rightarrow$ $L$ REGEX: $R(i,j,k)$ Dynamic Programming. $|\alpha|$ is exp in number of states.
1. \( L \text{ DFA} \rightarrow L \text{ REGEX} \): \( R(i, j, k) \) Dynamic Programming. \(|\alpha|\) is exp in number of states.

2. \( L \text{ REGEX} \rightarrow L \text{ NFA} \):
Equivalence of DFA, NFA, REGEX

1. $L \text{DFA} \rightarrow L \text{REGEX}$: $R(i, j, k)$ Dynamic Programming. $|\alpha|$ is exp in number of states.
2. $L \text{REGEX} \rightarrow L \text{NFA}$: induction on formation of a REGEX.
Equivalence of DFA, NFA, REGEX

1. $L \text{DFA} \rightarrow L \text{REGEX}$: $R(i, j, k)$ Dynamic Programming. $|\alpha|$ is exp in number of states.

2. $L \text{REGEX} \rightarrow L \text{NFA}$: induction on formation of a REGEX.

3. $L \text{NFA} \rightarrow L \text{DFA}$:
1. \( L \text{ DFA} \rightarrow L \text{ REGEX}: R(i, j, k) \) Dynamic Programming. \(|\alpha|\) is exp in number of states.

2. \( L \text{ REGEX} \rightarrow L \text{ NFA} \): induction on formation of a REGEX.

3. \( L \text{ NFA} \rightarrow L \text{ DFA} \): powerset construction. States blowup exponentially.
Closure Properties

1. **Union** What to use?
Closure Properties

1. **Union** What to use?  
   - DFA: Cross Product Construction, or  
   - REGEX: by definition, or  
   - NFA: e-transitions.

2. **Intersection** What to use?
Closure Properties

1. **Union** What to use?
   - DFA: Cross Product Construction, or
   - REGEX: by definition, or
   - NFA: e-transitions.

2. **Intersection** What to use?
   - DFA: Cross Product Construction.
   - NFA: Cross Product Construction.

3. **Complimentation** What to use?
Closure Properties

1. **Union** What to use?
   - DFA: Cross Product Construction, or
   - REGEX: by definition, or
   - NFA: e-transitions.

2. **Intersection** What to use?
   - DFA: Cross Product Construction.
   - NFA: Cross Product Construction.

3. **Complimentation** What to use?
   - DFA: Swap final and non-final states.

4. **Concatenation** What to use?
Closure Properties

1. **Union** What to use?
   DFA: Cross Product Construction, or
   REGEX: by definition, or
   NFA: e-transitions.

2. **Intersection** What to use?
   DFA: Cross Product Construction.
   NFA: Cross Product Construction.

3. **Complementation** What to use?
   DFA: Swap final and non-final states.

4. **Concatenation** What to use?
   NFA: e-transition from one machine to the other.
   REGEX: By Definition.

5. **Star** What to use?
Closure Properties

1. **Union** What to use?
   - DFA: Cross Product Construction, or
   - REGEX: by definition, or
   - NFA: e-transitions.

2. **Intersection** What to use?
   - DFA: Cross Product Construction.
   - NFA: Cross Product Construction.

3. **Complimentation** What to use?
   - DFA: Swap final and non-final states.

4. **Concatenation** What to use?
   - NFA: e-transition from one machine to the other.
   - REGEX: By Definition.

5. **Star** What to use?
   - DFA: On Midterm.
   - REGEX: By Definition.
**SUBSEQ Problems**

**Definition** If $w = \sigma_1 \cdots \sigma_n$ is a string then any string of the form

$$\sigma_{i_1} \cdots \sigma_{i_k}$$

where $i_1 < \cdots < i_k$ is a *subsequence of* $w$.

$\text{SUBSEQ}(w)$ is the set of all subsequences of the string $w$.

**Examples** If $w = aaba$ then the subsequences are

$\text{SUBSEQ}(aaba) = \{e, a, b, aa, ab, ba, aaa, aab, aba, aaba\}$.

**Definition** If $L \subseteq \{a, b\}^*$ then

$$\text{SUBSEQ}(L) = \bigcup_{w \in L} \text{SUBSEQ}(w).$$
Definition If \( w = \sigma_1 \cdots \sigma_n \) is a string then any string of the form \( \sigma_{i_1} \cdots \sigma_{i_k} \)

where \( i_1 < \cdots < i_k \) is a subsequence of \( w \).

\( \text{SUBSEQ}(w) \) is the set of all subsequences of the string \( w \).

Examples If \( w = aaba \) then the subsequences are

\[ \text{SUBSEQ}(aaba) = \{ e, a, b, aa, ab, ba, aaa, aab, aba, aaba \} \].

Definition If \( L \subseteq \{ a, b \}^* \) then

\[ \text{SUBSEQ}(L) = \bigcup_{w \in L} \text{SUBSEQ}(w) \].

T or F and prove:

1. If \( L \) is regular then \( \text{SUBSEQ}(L) \) is regular.
2. If \( L \) is context free then \( \text{SUBSEQ}(L) \) is context free.
If $L$ is regular then $SUBSEQ(L)$ is regular.
If $L$ is regular than $\text{SUBSEQ}(L)$ is regular. YES.
If $L$ is regular then $\text{SUBSEQ}(L)$ is regular. YES.
Let $M$ be a DFA for $L$.
We form an NFA for $\text{SUBSEQ}(L)$.
For every $\delta(p, \sigma) = q$ in $M$ we add $\delta(p, e) = q$. 
If $L$ is CFL than $\text{SUBSEQ}(L)$ is CFL.
If $L$ is CFL then $\text{SUBSEQ}(L)$ is CFL. YES.
If $L$ is CFL than $\text{SUBSEQ}(L)$ is CFL. YES.

Let $M$ be a CFL for $L$ in Chomsky Normal Form.

We form a CFL $\text{SUBSEQ}(L)$.

For every rule $A \rightarrow \sigma$ we add $A \rightarrow \varepsilon$. 
Context Free Languages

Definition
A Context Free Grammar (CFL) is \((V, \Sigma, P, S)\)

- \(V\) is set of nonterminals
- \(\Sigma\) is the alphabet, also called terminals
- \(P \subseteq V \times (V \cup \Sigma)^*\) are the productions or rules
- \(S \in V\) is the start symbol.

\(L(G)\) is the set of strings generated by CFL \(G\).

A Context Free Language (CFL) is a lang that is \(L(G)\) for some CFL \(G\).

A CFL is in Chomsky Normal Form (CNF) if all of the productions are either of the form
\[A \rightarrow BC\]
\[A \rightarrow \sigma\] where \(\sigma \in \Sigma\)
\[A \rightarrow \varepsilon\] (I didn't include it in class, but I am now.)

Note: If \(G\) is a CFL, then there exists a CNF CFL that generates it.
**Context Free Languages**

**Definition**
A **Context Free Grammar (CFL)** is $(V, \Sigma, P, S)$
- $V$ is set of nonterminals
- $\Sigma$ is the alphabet, also called terminals
- $P \subseteq V \times (V \cup \Sigma)^*$ are the productions or rules
- $S \in V$ is the start symbol.

$L(G)$ is the set of strings generated by CFL $G$.
A **Context Free Lang (CFL)** is a lang that is $L(G)$ for some CFL $G$. 
**Context Free Languages**

**Definition**

A **Context Free Grammar (CFL)** is 

\[(V, \Sigma, P, S)\]

- \(V\) is set of **nonterminals**
- \(\Sigma\) is the **alphabet**, also called **terminals**
- \(P \subseteq V \times (V \cup \Sigma)^*\) are the **productions** or **rules**
- \(S \in V\) is the start symbol.

\(L(G)\) is the set of strings generated by CFL \(G\).

A **Context Free Lang (CFL)** is a lang that is \(L(G)\) for some CFL \(G\).

A CFL is in **Chomsky Normal Form (CNF)** if all of the productions are either of the form

- \(A \rightarrow BC\)
- \(A \rightarrow \sigma\) where \(\sigma \in \Sigma\)
- \(A \rightarrow e\) (I didn’t include it in class, but I am now.)

**Note:** If \(G\) is a CFL then there exists a CNF CFL that generates it.
Examples of CFL’s that are NOT Regular

\{ a^n b^n : n \in \mathbb{N} \}
S \rightarrow aSb | e

To prove it works requires a proof by induction

Not to worry, I will ASSUME you could do such a proof and hence
WILL NOT make you.
Examples of CFL’s that are NOT Regular

\{ a^n b^n : n \in \mathbb{N} \} \\
S \rightarrow aSb|e

\{ w : \#_a(w) = \#_b(w) \} \\
S \rightarrow aSbS \\
S \rightarrow bSaS \\
S \rightarrow SS \\
S \rightarrow e \\
To prove it works requires a proof by induction
Examples of CFL’s that are NOT Regular

\{ a^n b^n : n \in \mathbb{N} \}
S \rightarrow aSb|e

\{ w : \#_a(w) = \#_b(w) \}
S \rightarrow aSbS
S \rightarrow bSaS
S \rightarrow SS
S \rightarrow e

To prove it works requires a proof by induction
Not to worry, I will ASSUME you could do such a proof and hence
WILL NOT make you.
Examples of Langs with Small CFL’s, Large NFA’s

\[ L = \{ a^n \} \]

- NFA requires \( \geq n - 2 \) states. Let’s prove it

There is a CNF CFL with \( \leq 2 \log_2 n \) rules. For \( n = 2 \) very easy. If not then have to write \( n \) as a sum of powers of 2. Example on next slide.
Examples of Langs with Small CFL’s, Large NFA’s

\[ L = \{ a^n \} \]

- NFA requires \( \geq n - 2 \) states. Let's prove it.
  If \( M \) is an NFA with \( \leq n - 2 \) states then find a path from the start state to the final state. Let \( a^m \) be the shortest string that take you from the start state to the final state. Since the number of states is \( \leq n - 2 \), \( m \leq n - 2 \). So we have \( a^m \) accepted when it should not be. Contradiction.

- There is a CNF CFL with \( \leq 2 \log_2 n \) rules.
  For \( n = 2^n \) VERY EASY. If not then have to write \( n \) as a sum of powers of 2. Example on next slide.
CNF CFG for \{a^{10}\}

\[10 = 2^3 + 2^1\]
CNF CFG for \( \{a^{10}\} \)

\[
10 = 2^3 + 2^1
\]

\[
S \rightarrow XY
\]
CNF CFG for \( \{ a^{10} \} \)

\[ 10 = 2^3 + 2^1 \]

\[ S \rightarrow XY \] We make \( X \Rightarrow a^8 \) and \( Y \Rightarrow a^2 \).
CNF CFG for \{a^{10}\}

10 = 2^3 + 2^1
S \rightarrow XY \text{ We make } X \Rightarrow a^8 \text{ and } Y \Rightarrow a^2.
X \rightarrow X_1X_1
X_1 \rightarrow X_2X_2
X_2 \rightarrow X_3X_3
X_3 \rightarrow a
Y \rightarrow Y_1Y_1
Y_1 \rightarrow a
10 = 2^3 + 2^1
S → XY We make X ⇒ a^8 and Y ⇒ a^2.
X → X_1X_1
X_1 → X_2X_2
X_2 → X_3X_3
X_3 → a
Y → Y_1Y_1
Y_1 → a
Can shorten a bit: We need Y ⇒ aa, so can just use X_2.
CNF CFG for \( \{ a^{10} \} \)

\[
10 = 2^3 + 2^1
\]

\[
S \rightarrow XY \quad \text{We make } X \Rightarrow a^8 \text{ and } Y \Rightarrow a^2.
\]

\[
X \rightarrow X_1X_1
\]

\[
X_1 \rightarrow X_2X_2
\]

\[
X_2 \rightarrow X_3X_3
\]

\[
X_3 \rightarrow a
\]

\[
Y \rightarrow Y_1Y_1
\]

\[
Y_1 \rightarrow a
\]

Can shorten a bit: We need \( Y \Rightarrow aa \), so can just use \( X_2 \).

\[
S \rightarrow XX_2
\]

\[
X \rightarrow X_1X_1
\]

\[
X_1 \rightarrow X_2X_2
\]

\[
X_2 \rightarrow X_3X_3
\]

\[
X_3 \rightarrow a
\]