1. Short answers:

(a) Define the following terms and concepts:

i. **Union, intersection, set concatenation, Kleene-star, set subtraction, complement**

**Answer:**
- Union: \( S \cup T = \{ x \mid x \in S \text{ or } x \in T \} \)
- Intersection: \( S \cap T = \{ x \mid x \in S \text{ and } x \in T \} \)
- Concatenation: \( S \circ T = \{ xy \mid x \in S, y \in T \} \)
- Kleene-star: \( S^* = \{ w_1 w_2 \cdots w_k \mid k \geq 0, w_i \in S \forall i = 1, 2, \ldots, k \} \)
- Subtraction: \( S - T = \{ x \mid x \in S, x \not\in T \} \)
- Complement: \( \overline{S} = \{ x \in \Omega \mid x \not\in S \} = \Omega - S \), where \( \Omega \) is the universe of all elements under consideration.

ii. A set \( S \) is closed under an operation \( f \)

**Answer:** \( S \) is closed under \( f \) if applying \( f \) to members of \( S \) always returns a member of \( S \).

iii. **Regular language**

**Answer:** A regular language is defined by a DFA.

iv. **Kleene’s theorem**

**Answer:** A language is regular if and only if it has a regular expression.

v. **Context-free language**

**Answer:** A CFL is defined by a context-free grammar (CFG).

vi. **Chomsky normal form**

**Answer:** A CFG is in Chomsky normal form if each of its rules has one of 3 forms:

\[ A \rightarrow BC, \quad A \rightarrow x, \quad \text{or} \quad S \rightarrow \varepsilon, \]

where \( A, B, C \) are variables, \( B \) and \( C \) are not the start variable, \( x \) is a terminal, and \( S \) is the start variable.

vii. **Church-Turing Thesis**

**Answer:** The informal notion of algorithm corresponds exactly to a Turing machine that always halts (i.e., a decider).

viii. **Turing-decidable language**

**Answer:** A language \( A \) that is decided by a Turing machine; i.e., there is a Turing machine \( M \) such that
- \( M \) halts and accepts on any input \( w \in A \), and
- \( M \) halts and rejects on input \( w \not\in A \).

**Loopy cannot happen.**

ix. **Turing-recognizable language**

**Answer:** A language \( A \) that is recognized by a Turing machine; i.e., there is a Turing machine \( M \) such that
- \( M \) halts and accepts on any input \( w \in A \), and
- \( M \) rejects or loops on any input \( w \not\in A \).
x. Co-Turing-recognizable language

**Answer:** A language whose complement is Turing-recognizable.

xi. Countable and uncountable sets

**Answer:**
- A set $S$ is countable if it is finite or we can define a correspondence between the positive integers and $S$.
- In other words, can create (possibly infinite) list of all elements in $S$ and each specific element will eventually appear in list.
- An uncountable set is a set that is not countable.
- A common approach to prove a set is uncountable is by using a diagonalization argument.

xii. Language $A$ is mapping reducible to language $B$, $A \leq_m B$

**Answer:**
- Suppose $A$ is a language defined over alphabet $\Sigma_1$, and $B$ is a language defined over alphabet $\Sigma_2$.
- Then $A \leq_m B$ means there is a computable function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that $w \in A$ iff $f(w) \in B$.

xiii. Function $f(n)$ is $O(g(n))$

**Answer:** There exist constants $c$ and $n_0$ such that $|f(n)| \leq c \cdot g(n)$ for all $n \geq n_0$.

xiv. Classes P and NP

**Answer:**
- P is the class of languages that can be decided by a deterministic Turing machine in polynomial time.
- NP is the class of languages that can be verified in (deterministic) polynomial time.
- Equivalently, NP is the class of languages that can be decided by a nondeterministic Turing machine in polynomial time.

xv. Language $A$ is polynomial-time mapping reducible to language $B$, $A \leq_P B$

**Answer:**
- Suppose $A$ is a language defined over alphabet $\Sigma_1$, and $B$ is a language defined over alphabet $\Sigma_2$.
- Then $A \leq_P B$ means $\exists$ polynomial-time computable function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that $w \in A$ iff $f(w) \in B$. 
xvi. NP-complete

**Answer:** Language $B$ is NP-Complete if $B \in \text{NP}$, and $B$ is NP-Hard ($\forall A \in \text{NP}, \text{we have } A \leq_p B$).

![NP-complete](image)

Typical approach for proving language $C$ is NP-Complete:
- first show $C \in \text{NP}$
- then show a known NP-Complete language $B$ satisfies $B \leq_p C$.

xvii. NP-hard

**Answer:** Lang $B$ is NP-hard if $A \leq_p B$ for every lang $A \in \text{NP}$.

[b] Give the transition functions $\delta$ (i.e., give domain and range) of a DFA, NFA, PDA, Turing machine and nondeterministic Turing machine.

**Answer:**
- DFA, $\delta: Q \times \Sigma \rightarrow Q$, where $Q$ is the set of states and $\Sigma$ is the alphabet.

- NFA, $\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$, where $\Sigma \cup \{\epsilon\}$ and $\mathcal{P}(Q)$ is the power set of $Q$.

- PDA, $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L,R\})$, where $\Gamma$ is the stack alphabet and $\Gamma \epsilon = \Gamma \cup \{\epsilon\}$.

- Turing machine, $\delta: Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L,R\})$, where $\Gamma$ is the tape alphabet, $L$ means move tape head one cell left, and $R$ means move tape head one cell right.

- Nondeterministic Turing machine, $\delta: Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L,R\})$.

Multiple choices when in state $q_i$ and read $c$ from tape:

$$\delta(q_i, c) = \{ (q_j, a, L), (q_k, c, R), (q_\ell, a, L), (q_\ell, d, R) \}$$
(c) Explain the “P vs. NP” problem.

Answer:
- $P$ is class of languages that can be solved in deterministic poly time.
- $NP$ is class of languages that can be verified in deterministic poly time (equivalently, solved by poly-time NTM).
- We know that $P \subseteq NP$.
- Each poly-time DTM is also a poly-time NTM.
- But it is currently unknown if $P = NP$ or $P \neq NP$.

NP

or

P

P = NP

Detailed Proof:
- Suppose there exists a TM $H$ that decides $A_{TM}$.
- Consider language $L = \{ \langle M \rangle \mid M \text{ is a TM that does not accept } \langle M \rangle \}$.
- Now construct a TM $D$ for $L$ using TM $H$ as a subroutine:
  $D = \text{"On input } \langle M \rangle \text{, where } M \text{ is a TM:} \quad$
  1. Run $H$ on input $\langle M, \langle M \rangle \rangle$.
  2. If $H$ accepts, reject. If $H$ rejects, accept."
- If we run TM $D$ on input $\langle D \rangle$, then $D$ accepts $\langle D \rangle$ if and only if $D$ doesn’t accept $\langle D \rangle$.
- Since this is impossible, TM $H$ must not exist.

2. Recall that $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts string } w \}$.

(a) Prove that $A_{TM}$ is undecidable. You may not cite any theorems or corollaries in your proof.

Overview of Proof:
- Suppose $A_{TM}$ is decided by some TM $H$, taking input $\langle M, w \rangle \in \Omega = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \text{ a string} \}$.
  $\langle M, w \rangle \rightarrow H \quad \text{accept, if } \langle M, w \rangle \in A_{TM}$
  $\langle M, w \rangle \rightarrow H \quad \text{reject, if } \langle M, w \rangle \notin A_{TM}$
- Define another TM (decider) $D$ using $H$ as a subroutine.
  $\langle M \rangle \rightarrow \langle M, \langle M \rangle \rangle \rightarrow H \quad \text{accept \rightarrow accept \rightarrow accept}$
  $\text{reject \rightarrow reject}$
- What happens when we run $D$ with input $\langle D \rangle$?
  - $D$ accepts $\langle D \rangle$ iff $D$ doesn’t accept $\langle D \rangle$, which is impossible.

(b) Show that $A_{TM}$ is Turing-recognizable.

Answer: Universal TM (UTM) $U$ recognizes $A_{TM}$:
$U =$ “On input $\langle M, w \rangle \in \Omega$, where $M$ is a TM and $w$ is a string:
  1. Run $M$ on $w$.
  2. If $M$ accepts $w$, accept; if $M$ rejects $w$, reject.”
$U$ recognizes $A_{TM}$ but does not decide $A_{TM}$
- When we run $M$ on $w$, there is the possibility that $M$ neither accepts nor rejects $w$ but rather loops on $w$. 
3. Each of the languages below in parts (a), (b), (c), (d) is of one of the following types:
   - Type REG. It is regular.
   - Type CFL. It is context-free, but not regular.
   - Type DEC. It is Turing-decidable, but not context-free.

   For each of the following languages, specify which type it is. Also, follow these instructions:
   - If a language $L$ is of Type REG, give a regular expression and a DFA (5-tuple) for $L$.
   - If a language $L$ is of Type CFL, give a context-free grammar (4-tuple) and a PDA (6-tuple) for $L$. Also, prove that $L$ is not regular.
   - If a language $L$ is of Type DEC, give a description of a Turing machine that decides $L$. Also, prove that $L$ is not context-free.

(a) $A = \{ w \in \Sigma^* \mid w = w^R \}$, where $\Sigma = \{0, 1\}$.

**Answer:** $A$ is of type CFL.

A CFG $G = (V, \Sigma, R, S)$ for $A$ has
- $V = \{S\}$,
- $\Sigma = \{0, 1\}$,
- starting variable $S$,
- rules $R = \{ S \rightarrow 00S00 \mid 01S10 \mid 10S01 \mid 11S11 \mid \varepsilon \}$.

---

Prove $A = \{ w \in \Sigma^* \mid w = w^R, \text{length of } w \text{ is divisible by } 4 \}$ nonregular.

- For a contradiction, suppose that $A$ is regular.
- Pumping Lemma (Theorem 1.1): If $L$ is regular language, then $\exists$ number $p$
  where, if $s \in L$ with $|s| \geq p$, then can split $s = xyz$ satisfying properties
  (1) $xy^iz \in L$ for each $i \geq 0$, (2) $|y| > 0$, (3) $|xy| \leq p$
- Let $p \geq 1$ be the pumping length of the pumping lemma.
- Consider string $s = 0^p1^{2p}0^p \in A$, and note that $|s| = 4p > p$, so conclusions of pumping lemma must hold.
- Since all of the first $p$ symbols of $s$ are 0s, (3) implies that $x$ and $y$ must only consist of 0s.
  Also, $z$ must consist of rest of 0s at the beginning, followed by $1^{2p}0^p$.
- Hence, we can write $x = 0^j$, $y = 0^k$, $z = 0^m 1^{2p}0^p$, where $j + k + m = p$ since $s = 0^p1^{2p}0^p = xy = 0^j 0^k 0^m 1^{2p}0^p$.
- Moreover, (2) implies that $k > 0$.
- Finally, (1) states that $x^yyzz$ must belong to $A$. However, 
  $$x^yyzz = 0^j 0^k 0^k 0^m 1^{2p}0^p = 0^j 0^k 1^{2p}0^p$$
  since $j + k + m = p$.
- But, $k > 0$ implies reverse($x^yyzz$) $\neq x^yyzz$, which means $x^yyzz \notin A$, which contradicts (1).
- Therefore, $A$ is a nonregular language.
(b) $B = \{ b^n a^n b^n \mid n \geq 0 \}$.

**Answer:** $B$ is of type DEC.

Below is a description of a Turing machine that decides $B$.

$M =$ "On input string $w \in \{a, b\}^*$:
1. Scan input to check if it's in $b^*a*b^*$; reject if not.
2. Return tape head to left-hand end of tape.
3. Repeat following until no more $b$'s left on tape.
   4. Replace the leftmost $b$ with $x$.
   5. Scan right until $a$ occurs. If no $a$'s, reject.
   6. Replace the leftmost $a$ with $x$.
   7. Scan right until $b$ occurs. If no $b$'s, reject.
   8. Replace the leftmost $b$ (after the $a$'s) with $x$.
   9. Return tape head to left end of tape; go to stage 3.
10. If tape contains any $a$'s, reject. Else, accept."

We now prove that $B$ is not context-free by contradiction.

(c) $C = \{ w \in \Sigma^* \mid n_a(w) \mod 4 = 1 \}$, where $\Sigma = \{a, b\}$ and $n_a(w)$ is the number of $a$'s in string $w$. For example, $n_a(babaabb) = 3$. Also, $3 \mod 4 = 3$, and $9 \mod 4 = 1$.

**Answer:** $C$ is of type REG.

A regular expression for $C$ is

$$(b^*ab*ab^*ab^*)^*b^*ab^*$$

$C = \{ w \in \Sigma^* \mid n_a(w) \mod 4 = 1 \}$

DFA 5-tuple $(Q, \Sigma, \delta, q_1, F)$

- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a, b\}$
- $q_1$ is start state
- $F = \{q_2\}$
- transition fcn $\delta : Q \times \Sigma \to Q$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_3$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$q_4$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>$q_1$</td>
</tr>
</tbody>
</table>
(d) \( D = \{ b^n a^n b^k c^k \mid n \geq 0, k \geq 0 \} \).

[Hint: Recall that the class of CFLs is closed under concatenation.]

**Answer:** \( D \) is of type CFL.

A CFG \( G = (V, \Sigma, R, S) \) for \( D \) has
- \( V = \{ S, X, Y \} \)
- \( \Sigma = \{ a, b, c \} \)
- starting variable \( S \)
- Rules \( R \):

\[
\begin{align*}
S & \rightarrow XY \\
X & \rightarrow bXa \mid \varepsilon \\
Y & \rightarrow bYc \mid \varepsilon
\end{align*}
\]

Prove \( D = \{ b^n a^n b^k c^k \mid n \geq 0, k \geq 0 \} \) not regular.

- Suppose that \( D \) is regular. Let \( p \geq 1 \) be pumping length of pumping lemma (Theorem 1.1).
- Consider string \( s = b^p \ a^p \ b^p \ c^p \in D \), and note that \( |s| = 4p > p \), so conclusions of pumping lemma must hold.
- Thus, can split \( s = xyz \) satisfying
  1. \( xy^iz \in D \) for all \( i \geq 0 \),
  2. \( |y| > 0 \),
  3. \( |xy| \leq p \).
- Since all of the first \( p \) symbols of \( s \) are \( b \)’s, (3) implies that \( x \) and \( y \) must consist of only \( b \)’s.
- Also, \( z \) is rest of \( b \)’s at beginning, followed by \( a^p \ b^p \ c^p \).
- Hence, we can write \( x = b^j \), \( y = b^k \), \( z = b^{m-p} \ a^p \ b^p \ c^p \), where \( j + k + m = p \) since \( s = b^p \ a^p \ b^p \ c^p = xy = b^j \ b^k \ b^{m-p} \ a^p \ b^p \ c^p \).
- Moreover, (2) implies that \( k > 0 \).
- Finally, (1) states that \( xyyz \) must belong to \( D \), but
  \[
  xyyz = b^j \ b^k \ b^m \ a^p \ b^p \ c^p = b^{p+j+k} \ a^p \ b^p \ c^p
  \]
  since \( j + k + m = p \). Also \( k > 0 \), so \( xyyz \notin D \), which contradicts (1). Therefore, \( D \) is a nonregular language.

---

**PDA for** \( D = \{ b^n a^n b^k c^k \mid n \geq 0, k \geq 0 \} \):

<table>
<thead>
<tr>
<th>Stack</th>
<th>Input</th>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( $ )</th>
<th>( \varepsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 )</td>
<td>( b )</td>
<td>( $ )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_5 )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_6 )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Important: \( q_3 \) to \( q_4 \) pops and pushes \( \$ \) to make sure stack is empty.

---

4. Each of the languages below in parts (a), (b), (c), (d) is of one of the following types:

- Type DEC. It is Turing-decidable.
- Type TMR. It is Turing-recognizable, but not decidable.
- Type NTR. It is not Turing-recognizable.

For each of the following languages, specify which type it is. Also, follow these instructions:

- If a language \( L \) is of Type DEC, give a description of a Turing machine that decides \( L \).
- If a language \( L \) is of Type TMR, give a description of a Turing machine that recognizes \( L \). **Also, prove that \( L \) is not decidable.**
- If a language \( L \) is of Type NTR, give a proof that it is not Turing-recognizable.
In each part below, if you need to prove that the given language $L$ is decidable, undecidable, or not Turing-recognizable, you must give an explicit proof of this; i.e., do not just cite a theorem that establishes this without a proof. However, if in your proof you need to show another language $L'$ has a particular property for which there is a theorem that establishes this without a proof, then you may simply cite the theorem without proof. However, if in your proof you need to show another language $L'$ has a particular property for which there is a theorem that establishes this, you may simply cite the theorem without proof.

(a) $\overline{A_{TM}}$, where $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts string } w \}$.  

**Answer:** $\overline{A_{TM}}$ is of type NTR, which is just Theorem 4.M.  

**Proof:**  
- If $\overline{A_{TM}}$ were Turing-recognizable, then $A_{TM}$ would be both Turing-recognizable (see slide 4-25) and co-Turing-recognizable.  
- But then Theorem 4.L would imply that $A_{TM}$ is decidable, which we know is not true by Theorem 4.1.  
- Hence, $\overline{A_{TM}}$ is not Turing-recognizable.

(b) $EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are TMs with } L(M_1) = L(M_2) \}$.  

**Answer:** $EQ_{TM}$ is of type NTR (see Theorem 5.K).  

Prove by showing $\overline{A_{TM}} \leq_m EQ_{TM}$ and applying Corollary 5.I.  

- Define reducing function $f(\langle M, w \rangle) = \langle M_1, M_2 \rangle$, where  
  - $M_1$ = “reject on all inputs.”  
  - $M_2$ = “On input $x$:  
    1. Ignore input $x$, and run $M$ on $w$.  
    2. If $M$ accepts $w$, accept; if $M$ rejects $w$, reject.”  
- $L(M_1) = \emptyset$.  
- If $M$ accepts $w$ (i.e., $\langle M, w \rangle \not\in \overline{A_{TM}}$), then $L(M_2) = \Sigma^*$.  
- If $M$ doesn’t accept $w$ (i.e., $\langle M, w \rangle \in \overline{A_{TM}}$), then $L(M_2) = \emptyset$.  
- Thus, $\langle M, w \rangle \in \overline{A_{TM}} \iff f(\langle M, w \rangle) = \langle M_1, M_2 \rangle \in EQ_{TM}$, so $\overline{A_{TM}} \leq_m EQ_{TM}$.  
- But $\overline{A_{TM}}$ is not TM-recognizable (Corollary 4.M), so $EQ_{TM}$ is not TM-recognizable by Corollary 5.I.

(c) $HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that halts on input } w \}$.  

[Hint: modify universal TM to show $HALT_{TM}$ is TM-recognizable.]  

**Answer:** $HALT_{TM}$ is of type TMR (see Theorem 5.A).  

**Decision problem:** Given TM $M$ and string $w$, does $M$ halt on input $w$?  

**Universe:** $\Omega_H = \{ \langle M, w \rangle \mid \text{TM } M, \text{string } w \}$.  

**Consider following Turing machine $T$:**  

$T = \text{"On input } \langle M, w \rangle \in \Omega_H, \text{where } M \text{ is TM and } w \text{ is string: }$  

1. Run $M$ on $w$.  
2. If $M$ halts (i.e., accepts or rejects) on $w$, accept.”  

- TM $T$ recognizes $HALT_{TM}$  
  - accepts each $\langle M, w \rangle \in HALT_{TM}$  
  - loops on each $\langle M, w \rangle \not\in HALT_{TM}$

We now prove that $HALT_{TM}$ is undecidable, which is Theorem 5.A.  

- We will show that $A_{TM}$ reduces to $HALT_{TM}$, where  
  - $A_{TM} \subseteq \Omega_A \equiv \{ \langle M, w \rangle \mid \text{TM } M, \text{string } w \}$  
  - $HALT_{TM} \subseteq \Omega_H \equiv \{ \langle M, w \rangle \mid \text{TM } M, \text{string } w \}$.  
- Suppose $\exists$ TM $R$ that decides $HALT_{TM}$.  
- Then could use $R$ to build a TM $S$ to decide $A_{TM}$ by modifying UTM to first use $R$ to check if it’s safe to run $M$ on $w$.  

$S = \text{"On input } \langle M, w \rangle \in \Omega_A, \text{where } M \text{ is TM and } w \text{ is string: }$  

1. Run $R$ on input $\langle M, w \rangle$.  
2. If $R$ rejects, reject.  
3. If $R$ accepts, simulate $M$ on input $w$ until it halts.  
4. If $M$ accepts, accept; otherwise, reject.”  

- Since TM $R$ is a decider, TM $S$ always halts and decides $A_{TM}$.  
- However, $A_{TM}$ is undecidable (Theorem 4.I), so that must mean that $HALT_{TM}$ is also undecidable.
(d) $EQ_{DFA} = \{ \langle M_1, M_2 \rangle | M_1, M_2 \text{ are DFAs with } L(M_1) = L(M_2) \}$. 

**Answer:** $EQ_{DFA}$ is of type DEC (see Theorem 4.E).

- **Decision problem:** For DFAs $M_1, M_2$, is $L(M_1) = L(M_2)$?
- **Universe:** $\Omega = \{ \langle M_1, M_2 \rangle | M_1, M_2 \text{ are DFAs} \}$.
- **The following TM $T$ decides $EQ_{DFA}$:**

  $T = \text{"On input } \langle A, B \rangle \in \Omega, \text{ where } A \text{ and } B \text{ are DFAs:"
  1. Check if $\langle A, B \rangle$ properly encodes 2 DFAs. If not, reject.
  2. Construct DFA $C$ such that $L(C) = [L(A) \cap \overline{L(B)}] \cup [L(A) \cap L(B)]$
      using algorithms for DFA union, intersection and complementation.
  3. Run TM that decides $E_{DFA}$ (Theorem 4.D) on $\langle C \rangle$.
  4. If $\langle C \rangle \in E_{DFA}$, accept; if $\langle C \rangle \not\in E_{DFA}$, reject."

---

**Answer:** The answer is NO.

- For each $k \geq 1$, let $L_k = \{ a^k b^k \}$, so $L_k$ is a language consisting of just a single string $a^k b^k$.
- Since $L_k$ is finite, it must be a regular language by Theorem 1.F.
- But $L = \cup_{k=1}^{\infty} L_k = \{ a^k b^k | k \geq 1 \}$, which we know is not regular (see end of Chapter 1).

---

5. Let $L_1, L_2, L_3, \ldots$ be an infinite sequence of regular languages, each of which is defined over a common input alphabet $\Sigma$.

- Let $L = \cup_{k=1}^{\infty} L_k$ be the infinite union of $L_1, L_2, L_3, \ldots$.
- Is it always the case that $L$ is a regular language?
- If your answer is YES, give a proof.
- If your answer is NO, give a counterexample.
- Explain your answer.
- Hint: Consider, for each $k \geq 1$, the language $L_k = \{ a^k b^k \}$.

---

6. Let $L_1$, $L_2$, and $L_3$ be languages defined over the alphabet $\Sigma = \{ a, b \}$, where

- $L_1$ consists of all possible strings over $\Sigma$ except the strings $w_1, w_2, \ldots, w_{100}$; i.e.,
  - start with all possible strings over the alphabet
  - take out 100 particular strings
  - the remaining strings form the language $L_1$;
- $L_2$ is recognized by an NFA; and
- $L_3$ is recognized by a PDA.

Prove that $(L_1 \cap L_2) \cap L_3$ is a context-free language.

[Hint: First show that $L_1$ and $L_2$ are regular. Also, consider $\overline{L_1}$]
Answer:
• \( L_1 = \{w_1, w_2, \ldots, w_{100}\} \), so \( |L_1| = 100 \). Thus, \( L_1 \) is a regular language since it is finite by Theorem 1.F.
• Then Theorem 1.H implies that the complement of \( L_1 \) must be regular, but the complement of \( L_1 \) is \( L_1 \). Thus, \( L_1 \) is regular.

• Language \( L_2 \) has an NFA, so it also has a DFA by Theorem 1.C. Therefore, \( L_2 \) is regular.
• Since \( L_1 \) and \( L_2 \) are regular, \( L_1 \cap L_2 \) must be regular by Theorem 1.G. Theorem 2.B then implies that \( L_1 \cap L_2 \) is CFL.
• Since \( L_3 \) has a PDA, \( L_3 \) is CFL by Theorem 2.C.
• Hence, since \( L_1 \cap L_2 \) and \( L_3 \) are both CFLs, their concatenation is CFL by Theorem 2.F.

7. Write Y or N in the entries of the table below to indicate which classes of languages are closed under which operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Regular languages</th>
<th>CFLs</th>
<th>Decidable languages</th>
<th>Turing-recognizable languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>∪</td>
<td>Y (Thm 1.A)</td>
<td>Y (Thm 2.E)</td>
<td>Y (HW 7, prob 2a)</td>
<td>Y (HW 7, prob 2b)</td>
</tr>
<tr>
<td>∩</td>
<td>Y (Thm 1.G)</td>
<td>N (HW 6, prob 2a)</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Compl.</td>
<td>Y (Thm 1.H)</td>
<td>N (HW 6, prob 2b)</td>
<td>Y (swap acc/rej)</td>
<td>N (e.g., ( A_{TM} ))</td>
</tr>
</tbody>
</table>

Answer:

8. Consider the following CFG \( G \) in Chomsky normal form:

\[
S \rightarrow a \mid YZ \\
Z \rightarrow ZY \mid a \\
Y \rightarrow b \mid ZZ \mid YY
\]

Use CYK (dynamic programming) algorithm to fill in following table to determine if \( G \) generates string \( babba \). Does \( G \) generate \( babba \)?

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>Y</td>
</tr>
<tr>
<td>2</td>
<td>S, Z</td>
<td>Z</td>
<td>Z</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>Y</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Y</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>S, Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

\( G \) does not generate \( babba \) because \( S \) is not in \((1, 5)\) entry
9. Recall that

\[ \text{CLIQUE} = \{ \langle G, k \rangle \mid G \text{ is undirected graph with } k\text{-clique} \} \]

\[ \subseteq \{ \langle G, k \rangle \mid G \text{ is undirected graph, integer } k \} \equiv \Omega_C, \]

\[ 3\text{SAT} = \{ \langle \phi \rangle \mid \phi \text{ is satisfiable 3cnf-function} \} \]

\[ \subseteq \{ \langle \phi \rangle \mid \phi \text{ is 3cnf-function} \} \equiv \Omega_3. \]

- Show that CLIQUE is NP-Complete by showing that CLIQUE ∈ NP and 3SAT ≤_P CLIQUE.
- Be sure to prove your reduction works and that it takes polynomial time.
- Also, be sure to provide proofs of these results, and don’t just cite a theorem.

### Answer:

Prove CLIQUE ∈ NP

- Certificate c is the k-clique.
- Here is a verifier for CLIQUE:
  \[ V = \text{"On input } \langle \langle G, k \rangle, c \rangle \text{:
      1. Test whether } c \text{ is a set of } k \text{ different nodes in } G.
      2. Test whether } G \text{ contains all edges connecting nodes in } c.
      3. If both tests pass, accept; otherwise, reject."} \]

- If graph G has m nodes, then (when G is encoded as list of nodes followed by list of edges)
  - Stage 1 takes \( O(k)O(m) = O(km) \) time.
  - Stage 2 takes \( O(k^2)O(m^2) = O(k^2m^2) \) time.

Prove 3SAT ≤_m CLIQUE

#### Proof Idea:

Convert instance \( \phi \) of 3SAT problem with \( k \) clauses into instance \( \langle G, k \rangle \) of clique problem.

- Reducing fcn \( f : \Omega_3 \to \Omega_C \)
  - \( \langle \phi \rangle \in 3\text{SAT} \iff f(\langle \phi \rangle) = \langle G, k \rangle \in \text{CLIQUE} \)
  - Suppose \( \phi \) is a 3cnf-function with \( k \) clauses, e.g.,
    \[ \phi = (x_1 \lor x_1 \lor x_2) \land (x_1 \lor x_2 \lor x_2) \land (x_1 \lor x_2 \lor x_2) \]
  - Convert \( \phi \) into a graph \( G \) as follows:
    - Nodes in \( G \) are organized into \( k \) triples \( t_1, t_2, \ldots, t_k \).
    - Triple \( t_i \) corresponds to the \( i \)th clause in \( \phi \).
    - Each node in a triple corresponds to a literal within the clause.
    - Add edges between each pair of nodes, except
      - within same triple
      - between contradictory literals, e.g., \( x_1 \) and \( \overline{x_1} \)
  - Prove \( \langle \phi \rangle \in 3\text{SAT} \iff \langle G, k \rangle \in \text{CLIQUE} \).
\[ \text{3SAT} \leq_m \text{CLIQUE} \]

- 3cnf-formula with \( k = 3 \) clauses and \( m = 2 \) variables
  \[ \phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor x_2) \land (\overline{x_1} \lor x_2 \lor x_2) \]
  is satisfiable by assignment \( x_1 = 0, x_2 = 1 \).
- Corresponding graph has \( k \)-clique:

\[ \quad \]

**Claim:** \( \langle \phi \rangle \in 3\text{SAT} \) iff \( \langle G, k \rangle \in \text{CLIQUE} \).

**Proof.** Use that \( G \) has edges between every pair of nodes except for
- pairs in same triple
- contradictory literals.

Also, \( \phi \) satisfiable iff each clause has \( \geq 1 \) true literal.

**Claim:** The mapping \( \phi \to \langle G, k \rangle \) is polynomial-time computable.

**Proof.**
- Given 3cnf-function \( \phi \) with \( k \) clauses \( m \) variables.
- Constructing graph \( G \)
  - \( G \) has \( 3k \) nodes
  - Adding edges entails considering each pair of nodes in \( G \):
    \[ \binom{3k}{2} = \frac{3k(3k-1)}{2} = O(k^2) \]
  - Time to construct \( G \) is polynomial in size of 3cnf-function \( \phi \).

10. Recall that

\[ \text{3SAT} = \{ \langle \phi \rangle \mid \phi \text{ is satisfiable 3cnf-function } \} \]
\[ \subseteq \{ \langle \phi \rangle \mid \phi \text{ is 3cnf-function } \} \equiv \Omega_3, \]
\[ \text{ILP} = \{ \langle A, b \rangle \mid \text{matrix } A \text{ and vector } b \text{ satisfy } Ay \leq b \}
  \quad \text{for some integer vector } y \}
\[ \subseteq \{ \langle A, b \rangle \mid \text{matrix } A \text{ and vector } b \} \equiv \Omega_I \]

- Show that \( \text{ILP} \) is NP-Complete by showing that \( \text{ILP} \in \text{NP} \) and \( \text{3SAT} \leq_p \text{ILP} \).
- Be sure to prove your reduction works and takes polynomial time.
- Also, be sure to provide proofs of these results, and don't just cite a theorem.
- \( Ay \leq b \) denotes

\[ a_{11} y_1 + a_{12} y_2 + \cdots + a_{1n} y_n \leq b_1 \]
\[ a_{21} y_1 + a_{22} y_2 + \cdots + a_{2n} y_n \leq b_2 \]
\[ \vdots \]
\[ a_{m1} y_1 + a_{m2} y_2 + \cdots + a_{mn} y_n \leq b_m \]

**ILP \in \text{NP}**

**Proof.**
- The certificate \( c \) is an integer vector satisfying \( Ac \leq b \).
- Here is a verifier for \( \text{ILP} \):
  \[ V = \text{"On input } \langle \langle A, b \rangle, c \rangle \text{:"} \]
  1. Test whether \( c \) is a vector of all integers.
  2. Test whether \( Ac \leq b \).
  3. If both tests pass, accept; otherwise, reject."

- If \( Ay \leq b \) has \( m \) inequalities and \( n \) variables, then
  - Stage 1 takes \( O(n) \) time
  - Stage 2 takes \( O(mn) \) time
  - So verifier \( V \) runs in \( O(mn) \),
    which is polynomial in size of problem instance.

Now prove \( \text{ILP} \) is NP-Hard by showing \( \text{3SAT} \leq_p \text{ILP} \).
**3SAT \leq_{m} ILP**

- Consider 3cnf-formula with $m = 4$ variables and $k = 3$ clauses:
  \[ \phi = (x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_2} \lor x_4) \land (x_2 \lor x_4 \lor \overline{x_3}) \]

- Define integer linear program with
  - $2m = 8$ variables $y_1, y'_1, y_2, y'_2, y_3, y'_3, y_4, y'_4$
  - $y_i$ corresponds to $x_i$
  - $y'_i$ corresponds to $\overline{x_i}$

  - 3 sets of inequalities for each pair $y_i, y'_i$:
    - $0 \leq y_1 \leq 1, \quad 0 \leq y'_1 \leq 1, \quad y_1 + y'_1 = 1$
    - $0 \leq y_2 \leq 1, \quad 0 \leq y'_2 \leq 1, \quad y_2 + y'_2 = 1$
    - $0 \leq y_3 \leq 1, \quad 0 \leq y'_3 \leq 1, \quad y_3 + y'_3 = 1$
    - $0 \leq y_4 \leq 1, \quad 0 \leq y'_4 \leq 1, \quad y_4 + y'_4 = 1$

    all hold with $y_i, y'_i$ integer iff one of $y_i, y'_i$ is 1, and other 0.

- $0 \leq y_i \leq 1 \iff -y_i \leq 0 \text{ and } y_i \leq 1$

- $y_i + y'_i = 1 \iff y_i + y'_i \leq 1 \text{ and } y_i + y'_i \geq 1$

**3SAT \leq_{m} ILP**

- Given 3cnf-formula:
  \[ \phi = (x_1 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_2} \lor x_4) \land (x_2 \lor x_4 \lor \overline{x_3}) \]

- Constructed ILP:
  - $0 \leq y_1 \leq 1, \quad 0 \leq y'_1 \leq 1, \quad y_1 + y'_1 = 1$
  - $0 \leq y_2 \leq 1, \quad 0 \leq y'_2 \leq 1, \quad y_2 + y'_2 = 1$
  - $0 \leq y_3 \leq 1, \quad 0 \leq y'_3 \leq 1, \quad y_3 + y'_3 = 1$
  - $0 \leq y_4 \leq 1, \quad 0 \leq y'_4 \leq 1, \quad y_4 + y'_4 = 1$

  \[ y_1 + y_2 + y_3 \geq 1 \]

  \[ y'_1 + y'_2 + y'_4 \geq 1 \]

  \[ y'_2 + y'_4 + y'_3 \geq 1 \]

- Note that:
  \[ \phi \text{ satisfiable } \iff \text{ constructed ILP has solution} \]

  (with values of variables $\in \{0, 1\}$)

**Reducing 3SAT to ILP Takes Polynomial Time**

- Given 3cnf-formula $\phi$ with
  - $m$ variables: $x_1, x_2, \ldots, x_m$
  - $k$ clauses

- Constructed ILP has
  - $2m$ variables: $y_1, y'_1, y_2, y'_2, \ldots, y_m, y'_m$
  - $6m + k$ inequalities:
    - 3 sets of inequalities for each pair $y_i, y'_i$:
      \[ 0 \leq y_i \leq 1, \quad 0 \leq y'_i \leq 1, \quad y_i + y'_i = 1, \]
      so total of $6m$ inequalities of this type.
    - For each clause in $\phi$, ILP has corresponding inequality, e.g.,
      \[ (x_1 \lor x_2 \lor \overline{x_3}) \iff y_1 + y_2 + y'_3 \geq 1, \]
      so total of $k$ inequalities of this type.

- Thus, size of ILP is polynomial in $m$ and $k$. 

\[ \phi \text{ satisfiable } \iff \text{ constructed ILP has solution} \]

(with values of variables $\in \{0, 1\}$)