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:** \( S - T = \{ x \mid x \in S, x \notin T \} \)
- **Complement:** \( \overline{S} = \{ x \in \Omega \mid x \notin S \} = \Omega - S \)

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 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 \( w \notin A \).

Looping 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 \( w \notin A \).
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 \( S \) and the positive integers.
- In other words, we can create a list of all the elements in \( S \) and each specific element will eventually appear in the 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 \( w \in A \) if and only if \( f(w) \in B \).

\[
\Sigma_1^* \xrightarrow{f} \Sigma_2^*
\]

\[ w \in A \iff f(w) \in B \]

Yes instance for problem \( A \) \iff Yes instance for problem \( B \)

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 \( w \in A \) if and only if \( f(w) \in B \).
xvi. NP-complete

**Answer:** Language $B$ is NP-Complete if $B \in \text{NP}$, and for every language $A \in \text{NP}$, we have $A \leq_{\text{p}} B$.

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

xvii. NP-hard

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

(b) Give the transition functions $\delta$ 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, $\delta : Q \times \Sigma_{\varepsilon} \rightarrow \mathcal{P}(Q)$, where $\Sigma_{\varepsilon} = \Sigma \cup \{\varepsilon\}$ and $\mathcal{P}(Q)$ is the power set of $Q$.
- PDA, $\delta : Q \times \Sigma_{\varepsilon} \times \Gamma_{\varepsilon} \rightarrow \mathcal{P}(Q \times \Gamma_{\varepsilon})$, where $\Gamma$ is the stack alphabet and $\Gamma_{\varepsilon} = \Gamma \cup \{\varepsilon\}$.
- Turing machine, $\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$.
- 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:

$\delta(q_i, c) = \{(q_j, a, L), (q_k, c, R), (q_{\ell}, a, L), (q_{\ell}, d, R)\}$

---

**Stack**

$a \rightarrow c$

Before

$\begin{array}{c|c|c}
  & b & c \\
  d & d & \\
  \$ & \$ & \\
\end{array}$

After

**Tape**

read $\rightarrow$ write, move

Before $a \ b \ a \ \underline{\ ; \ ;}$

After $a \ b \ b \ a \ \underline{\ ; \ ;}$
(c) Explain the “P vs. NP” problem.

**Answer:**
- P is the class of languages that can be solved in polynomial time.
- NP is the class of languages that can be verified in (deterministic) polynomial time.
- We know that $P \subseteq NP$, but it is currently unknown if $P = NP$ or $P \neq NP$.

2. Recall that $A_{TM} = \{ \langle M, w \rangle \mid M$ 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 $\langle M, w \rangle$.

\[
\langle M, w \rangle \rightarrow \begin{cases} 
accept, & \text{if } \langle M, w \rangle \in A_{TM} \\
reject, & \text{if } \langle M, w \rangle \notin A_{TM}
\end{cases}
\]

- Define another TM $D$ using $H$ as a subroutine.

\[
\begin{align*}
D & \rightarrow \langle M, \langle M \rangle \rangle \\
\langle M \rangle & \rightarrow \langle M, \langle M \rangle \rangle \\
\langle M, \langle M \rangle \rangle & \rightarrow H \\
H & \rightarrow accept \\
H & \rightarrow reject
\end{align*}
\]

- 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:** The universal TM $U$ recognizes $A_{TM}$, where $U$ is defined as follows:

\[U = "\text{On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ is a string:}"
\]

1. Run $M$ on $w$.
2. If $M$ accepts $w$, accept; if $M$ rejects $w$, reject.

Note that $U$ only recognizes $A_{TM}$ and does not decide $A_{TM}$ since when we run $M$ on $w$, there is the possibility that $M$ neither accepts nor rejects $w$ but rather loops on $w$. 

---

**CS 341 Practice Final**
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 for $L$.
- If a language $L$ is of Type CFL, give a context-free grammar and a PDA 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) $A = \{ w \in \Sigma^* \mid w = \text{reverse}(w) \text{ and the length of } w \text{ is divisible by } 4 \}$, where $\Sigma = \{0, 1\}$.

Answer: $A$ is of type CFL. A CFG for $A$ has rules $S \rightarrow 0S00 \mid 01S10 \mid 10S01 \mid 11S11 \mid \varepsilon$.

A PDA for $A$ is as follows:

We now prove that $A$ is not regular by contradiction.

- Suppose that $A$ is regular. Let $p \geq 1$ be the pumping length of the pumping lemma (Theorem 1.1).
- Consider string $s = 0^p1^2p0^p \in A$, and note that $|s| = 4p > p$, so conclusions of pumping lemma must hold.
- Thus, can split $s = xyz$ satisfying (1) $xy^iz \in A$ for all $i \geq 0$, (2) $|y| > 0$, and (3) $|xy| \leq p$.

(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 $b^n a^n b^n$; 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.
Suppose that $B$ is context-free.
• Let $p$ be pumping length of CFL pumping lemma (Theorem 2.D).
• Consider string $s = b^p a^p b^p \in B$. Note that $|s| = 3p > p$, so the pumping lemma will hold.
• Thus, can split $s = b^p a^p b^p = uvxyz$ satisfying $uv^ixy^iz \in B$ for all $i \geq 0$, $|vy| \geq 1$, and $|vxy| \leq p$.
• We now consider all of the possible choices for $v$ and $y$:
  • Suppose strings $v$ and $y$ are uniform (e.g., $v = b^j$ for some $j \geq 0$, and $y = a^k$ for some $k \geq 0$). Then $|vy| \geq 1$ implies that $v \neq \epsilon$ or $y \neq \epsilon$ (or both), so $uv^2xy^2z$ won’t have the correct number of $b$’s at the beginning, $a$’s in the middle, and $b$’s at the end. Hence, $uv^2xy^2z \not\in B$.
  • Now suppose strings $v$ and $y$ are not both uniform. Then $uv^2xy^2z$ won’t have form $b \cdots ba \cdots ab \cdots b$, so $uv^2xy^2z \not\in B$.
• Every case gives contradiction, so $B$ is not a CFL.

(d) $D = \{ b^n a^n b^k c^k | n \geq 0, k \geq 0 \}$. [Hint: Recall that the class of context-free languages is closed under concatenation.]

Answer: $D$ is of type CFL. A CFG for $D$ is

$$S \to XY$$
$$X \to bXa | \epsilon$$
$$Y \to bYe | \epsilon$$

A PDA for $D$ is below:

![PDA diagram](image)

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

We now prove that $D$ is not regular by contradiction.
• Suppose that $D$ is regular. Let $p \geq 1$ be pumping length of pumping lemma (Theorem 1.I).
• 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.

(c) $C' = \{ w \in \Sigma^* | n_a(w) \mod 4 = 1 \}$, 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 $(a^*ba^*ab^*a^*b^*)^*b^*a^*$, and a DFA for $C$ is below:

![DFA diagram](image)

• Thus, can split $s = xyz$ satisfying (1) $xy^iz \in D$ for all $i \geq 0$, (2) $|y| > 0$, and (3) $|xy| \leq p$.
• Since all of the first $p$ symbols of $s$ are $b$’s, (3) implies that $x$ and $y$ must only consist of $b$’s. Also, $z$ must consist of the rest of the $b$’s at the beginning, followed by $a^p b^p c^p$.
• Hence, we can write $x = b^j$, $y = b^k$, $z = b^m a^p b^p c^p$, where $j + k + m = p$ since $s = b^p a^p b^p c^p = xyz = b^j b^k b^m a^p b^p c^p$.
• Moreover, (2) implies that $k > 0$.
• Finally, (1) states that $xyyz$ must belong to $D$, but $xyyz = b^j b^k b^m a^p b^p c^p = b^{p+k} a^p b^p c^p$ since $j + k + m = p$. Also $k > 0$, so $xyyz \not\in D$, which contradicts (1). Therefore, $D$ is a nonregular language.
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.

(a) $A_{TM}$, where $A_{TM} = \{ \langle M, w \rangle \mid M$ is a TM that accepts string $w \}$.
   **Answer:** $A_{TM}$ is of type NTR, which is just Theorem 4.M.
   **Proof:**
   - If $A_{TM}$ 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.I.
   - Hence, $A_{TM}$ is not Turing-recognizable.

(b) $EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1, M_2 $ are TMs with $L(M_1) = L(M_2) \}$.
   **[Hint: show $A_{TM} \leq_T EQ_{TM}$]**
   **Answer:** $EQ_{TM}$ is of type NTR (see Theorem 5.K). We prove this by showing $\overline{A_{TM}} \leq_T EQ_{TM}$ and applying Corollary 5.1.
   - Define the 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.”
   - $L(M_1) = \emptyset$.
   - If $M$ accepts $w$ (i.e., $\langle M, w \rangle \notin \overline{A_{TM}}$), 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 $\overline{A_{TM}} \leq_T EQ_{TM}$.
   - But $\overline{A_{TM}}$ is not TM-recognizable (Corollary 4.4.M), so $EQ_{TM}$ is not TM-recognizable by Corollary 5.1.

(c) $HALT_{TM} = \{ \langle M, w \rangle \mid M$ is a TM that halts on input $w \}$. [Hint: modify the universal TM to show $HALT_{TM}$ is Turing-recognizable.]
   **Answer:** $HALT_{TM}$ is of type TMR (see Theorem 5.A). The following Turing machine recognizes $HALT_{TM}$:
   - $T$ = “On input $\langle M, w \rangle$, where $M$ is a TM and $w$ is a string:
     1. Run $M$ on $w$.
     2. If $M$ halts on $w$, accept.”
   We now prove that $HALT_{TM}$ is undecidable, which is Theorem 5.A.
   - Suppose there exists a TM $R$ that decides $HALT_{TM}$.
   - Then could use $R$ to build a TM $S$ to decide $A_{TM}$ by modifying universal TM to first use $R$ to see if it’s safe to run $M$ on $w$. 
$S = \text{"On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ is a string:}\n$
\begin{enumerate}
\item Run $R$ on input $\langle M, w \rangle$.
\item If $R$ rejects, reject.
\item If $R$ accepts, simulate $M$ on input $w$ until it halts.
\item If $M$ accepts, accept; otherwise, reject."
\end{enumerate}

- Since TM $R$ is a decider, TM $S$ always halts and is a decider.
- Thus, deciding $A_{TM}$ is reduced to deciding $HALT_{TM}$.
- However, $A_{TM}$ is undecidable (Theorem 4.1), so that must mean that $HALT_{TM}$ is also undecidable.

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 $L = \bigcup_{k=1}^{\infty} L_k$ 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 $L_k = \{a^kb^k\}$.

(d) $EQ_{DFA} = \{ \langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are DFAs with } L(M_1) = L(M_2) \}$.  

Answer: $EQ_{DFA}$ is of type DEC (see Theorem 4.E). The following TM decides $EQ_{DFA}$:

\begin{enumerate}
\item Check if $\langle A, B \rangle$ properly encodes 2 DFAs. If not, reject.
\item Construct DFA $C$ such that $L(C) = [L(A) \cap \overline{L(B)}] \cup [\overline{L(A)} \cap L(B)]$ using algorithms for DFA union, intersection and complementation.
\item Run TM that decides $E_{DFA}$ (Theorem 4.D) on $\langle C \rangle$.
\item If $\langle C \rangle \in E_{DFA}$, accept; if $\langle C \rangle \notin E_{DFA}$, reject."

Answer: The answer is NO.
- For each $k \geq 1$, let $L_k = \{a^kb^k\}$, so $L_k$ is a language consisting of just a single string $a^kb^k$.
- Since $L_k$ is finite, it must be a regular language by Theorem 1.F.
- But $L = \bigcup_{k=1}^{\infty} L_k = \{ a^kb^k \mid k \geq 1 \}$, which we know is not regular (see end of Chapter 1).
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 $(L_1 \cap L_2)L_3$ is a context-free language.

[Hint: First show that $L_1$ and $L_2$ are regular.
Also, consider $\overline{L_1}$.]
9. Recall that

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

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

- Show that \text{CLIQUE} is \text{NP-Complete} by showing that \text{CLIQUE} \in \text{NP} and \text{3SAT} \leq_p \text{CLIQUE}.
- Be sure to prove your reduction works and that it requires polynomial time.
- Also, be sure to provide proofs of these results, and don’t just cite a theorem.

**Answer:**

Prove \text{3SAT} \leq_m \text{CLIQUE}

**Proof Idea:** Convert instance \( \phi \) of \text{3SAT} problem with \( k \) clauses into instance \( \langle G, k \rangle \) of clique problem.

- Suppose \( \phi \) is a 3cnf-function with \( k \) clauses, e.g.,

\[ \phi = (x_1 \lor \bar{x}_2 \lor x_3) \land (x_3 \lor \bar{x}_5 \lor \bar{x}_6) \land (x_3 \lor \bar{x}_6 \lor x_4) \land (x_2 \lor x_1 \lor \bar{x}_5) \]

- 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 \( \bar{x}_1 \)

- Prove \( \langle \phi \rangle \in \text{3SAT} \) iff \( \langle G, k \rangle \in \text{CLIQUE} \).
**3SAT \leq_m CLIQUE**

**Example:** 3cnf-function with \(k = 3\) clauses and \(m = 2\) variables:
\[
\phi = (x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2)
\]

Corresponding Graph:

- Clause 1
  - \(x_1\)
  - \(x_1\)
  - \(x_2\)
- Clause 2
  - \(x_1\)
  - \(x_1\)
  - \(x_2\)
- Clause 3
  - \(x_1\)
  - \(x_2\)
  - \(x_2\)

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

**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 \(\phi \rightarrow \langle G, k \rangle\) is polynomial-time computable.

**Proof.**
- Given 3cnf-function \(\phi\) with
  - \(k\) clauses
  - \(m\) variables.
- Constructed graph \(G\) has
  - \(3k\) nodes
  - \((\#\) of edges in \(G\)) \(= \frac{3k(3k-1)}{2} = O(k^2)\)
  - Size of graph \(G\) is polynomial in size of 3cnf-function \(\phi\).

**10. Recall that**

\[
ILP = \{ \langle A, b \rangle \mid \text{matrix } A \text{ and vector } b \text{ satisfy } Ay \leq b \text{ with } y \text{ an integer vector} \}.
\]

- Show that \(ILP\) is NP-Complete by showing that \(ILP \in NP\) and \(3SAT \leq_p ILP\).
- Be sure to prove your reduction works and that it requires polynomial time.
- Also, be sure to provide proofs of these results, and don’t just cite a theorem.
**ILP ∈ NP**

**Proof.**
- The certificate $c$ is an integer vector satisfying $Ac \leq b$.
- Here is a verifier for ILP:
  - $V = "On input \langle A, b, c \rangle:"$
  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.

Now prove ILP is NP-Hard by showing $3SAT \leq_p ILP$.

---

**3SAT ≤ₘ ILP**

- Recall 3cnf-formula with $m = 4$ variables and $k = 3$ clauses:
  \[ \phi = (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_4) \land (\overline{x}_2 \lor x_4 \lor \overline{x}_3) \]
  - $\phi$ satisfiable iff each clause evaluates to 1.
  - A clause evaluates to 1 iff at least one literal in the clause equals 1.
  - For each clause $(x_i \lor \overline{x}_j \lor x_k)$, create inequality $y_i + y'_j + y_k \geq 1$.
  - For our example, ILP has inequalities
    \[
    \begin{align*}
    y_1 + y_2 + y'_3 & \geq 1 \\
    y'_1 + y'_2 + y_4 & \geq 1 \\
    y'_2 + y'_4 + y'_3 & \geq 1
    \end{align*}
    \]
    which guarantee that each clause evaluates to 1.

- Consider 3cnf-formula with $m = 4$ variables and $k = 3$ clauses:
  \[ \phi = (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_4) \land (\overline{x}_2 \lor x_4 \lor \overline{x}_3) \]
  - 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 of pair $y_i, y'_i$:
      \[
      \begin{align*}
      0 \leq y_1 \leq 1, & \quad 0 \leq y'_1 \leq 1, & \quad y_1 + y'_1 = 1 \\
      0 \leq y_2 \leq 1, & \quad 0 \leq y'_2 \leq 1, & \quad y_2 + y'_2 = 1 \\
      0 \leq y_3 \leq 1, & \quad 0 \leq y'_3 \leq 1, & \quad y_3 + y'_3 = 1 \\
      0 \leq y_4 \leq 1, & \quad 0 \leq y'_4 \leq 1, & \quad y_4 + y'_4 = 1
      \end{align*}
      \]
    which guarantee that exactly one of $y_i$ and $y'_i$ is 1, and other is 0.
- $0 \leq y_i \leq 1 \iff -y_i \leq 0 \& y_i \leq 1$
- $y_i + y'_i = 1 \iff y_i + y'_i \leq 1 \& y_i + y'_i \geq 1$

- Given 3cnf-formula:
  \[ \phi = (x_1 \lor x_2 \lor \overline{x}_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_4) \land (\overline{x}_2 \lor x_4 \lor \overline{x}_3) \]
- Constructed ILP:
  \[
  \begin{align*}
  0 \leq y_1 \leq 1, & \quad 0 \leq y'_1 \leq 1, & \quad y_1 + y'_1 = 1 \\
  0 \leq y_2 \leq 1, & \quad 0 \leq y'_2 \leq 1, & \quad y_2 + y'_2 = 1 \\
  0 \leq y_3 \leq 1, & \quad 0 \leq y'_3 \leq 1, & \quad y_3 + y'_3 = 1 \\
  0 \leq y_4 \leq 1, & \quad 0 \leq y'_4 \leq 1, & \quad y_4 + y'_4 = 1 \\
  y_1 + y_2 + y'_3 \geq 1 \\
  y'_1 + y'_2 + y_3 \geq 1 \\
  y'_2 + y'_4 + y'_3 \geq 1
  \end{align*}
  \]
- 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 $k$. 