1. The Class \textbf{P}

2. The Class \textbf{NP}
   - How to Show that a Language is in \textbf{NP}

3. \textbf{NP}-Complete Languages

4. Proof that \textsc{circuit sat} is \textbf{NP}-Complete
   - Reduction from \textsc{accept} to \textsc{circuit sat}

5. Showing a New Language \textbf{NP}-Complete
   - Proof that \textsc{sat} and \textsc{3-sat} are \textbf{NP}-complete
The Class \( \mathbf{P} \)

**Definition**

A language \( L \subseteq \Gamma^* \) is in \( \mathbf{P} \) if there exists a DTM \( M \) that halts on every input in polynomial time such that for every \( x \in \Gamma^* \),

\[
x \in L \iff M(x) = 1
\]

where \( M(x) = 1(0) \) if \( M \) accepts (rejects) \( x \) in poly time in \( |x| \).

**How would you show that a language is in \( \mathbf{P} \)?**

- Let ET be the set of graphs \( G = (V, E) \) graphs with a **Eulerian tour**, a tour that visits each edge once. Show that \( ET \in \mathbf{P} \).
- Let MTX be the set of matrices \( A \) such that a linear system \( y = Ax \) has a solution. Show that \( MTX \in \mathbf{P} \).
Definition

A language \( L \subset \Gamma^* \) is in \( \textbf{NP} \) if there exists a polynomial \( p : \mathbb{N} \rightarrow \mathbb{N} \) and a polynomial time DTM \( M \) such that for every \( x \in \Gamma^* \),

\[
x \in L \iff \exists u \in \Gamma^{p(|x|)} \text{ s.t. } M(x, u) = 1
\]

If \( x \in L \), \( u \) is called a certificate for \( x \).

How would you show that a language is in \( \textbf{NP} \)?

- **CIRCUIT SAT**

  \textit{Instance}: A circuit description with \( n \) input variables \( \{x_1, x_2, \ldots, x_n\} \) for some integer \( n \) and a designated output gate.

  \textit{Answer}: “Yes” if there is an assignment of values to the variables such that the output of the circuit has value 1.
A circuit can be described by a straight-line program (SLP), a set of 4-tuples \((k, \text{OP}, i, j)\) (steps). \text{OP} is the operation at the \(k\)th step, \(i\) and \(j\) are the indices of steps providing inputs. Because the circuit is acyclic, \(i, j < k\). Without loss of generality, assume that the steps of an SLP are sorted. Let \(K\) be the number of lines.

The following describes a Full Adder on inputs, \(a\), \(b\), and \(c1\), and outputs, \(c2\) and \(s\). Its SLP = \(\{(c2, \lor, g, e), (g, \land, d, c1), (s, \oplus, d, c1), (e, \land, a, b), (d, \oplus, a, b)\}\). Without \((s, \oplus, d, c1)\), it is a “Yes” instance of CIRCUIT SAT.
Proof that **Circuit SAT** is in **NP**

**Pseudo Code for Circuit SAT with $K$ Gates on $N$ Inputs**

Let $v_k$ hold value of $k$th step and let inputs be $(u_1, u_2, \ldots, u_N)$.

\[
\text{for } 1 \leq k \leq K \text{ do} \\
\quad \text{Evaluate } v_k \\
\text{end for} \\
\text{Accept if } v_K = 1.
\]

Running time is polynomial in the length of the input.

The program is correct: $v_k$ depends on $v_i$ and $v_j$ for $i, j < k$.

**Example**

In the following evaluate $d$, $e$, $g$, $c2$ in that order.

\[
\begin{align*}
(c2, \lor, g, e) \\
(g, \land, d, c1) \\
(e, \land, a, b) \\
(d, \oplus, a, b)
\end{align*}
\]
The Class **NP**

**Definition**
A language $L \subseteq \Gamma^*$ is in **NP** if there exists a polynomial $p : \mathbb{N} \mapsto \mathbb{N}$ and a polynomial time DTM $M$ such that for every $x \in \Gamma^*$,

$$x \in L \iff \exists u \in \Gamma^{p(|x|)} \text{ s.t. } M(x, u) = 1$$

$u$ is called a **certificate** for $x$ when $x \in L$ and $u \in \Gamma^{p(|x|)}$ satisfy $M(x, u) = 1$.

**Example**

- **ACCEPT** = \{ $<M, x, 1^t>$ | $M$ is a DTM, $x \in \Gamma^*$, $\exists y \in \Gamma^*$ such that $M$ accepts $<x, y>$ in $t$ steps. \}

- **ACCEPT** is in **NP**.
**Theorem**

**ACCEPT is NP-complete.**

**Proof.**

Explain why **ACCEPT** is in **NP**.

Reduce an arbitrary language $L_1$ in **NP** to **ACCEPT**. By definition, $L_1$ in **NP** if there is an NTM $M_1$ that accepts every $x \in L_1$ in $p(n)$ steps where $n = |x|$ and $p(n)$ is a polynomial. An instance of $L_1$ is $(M_1, x, p(\cdot))$. Translate each such instance to an instance $< M_1, x, 1^{p(|x|)} >$ of **ACCEPT**. The latter instance is accepted if and only if the former is accepted. The reduction takes time polynomial in the length of $(M_1, x, p(\cdot))$ to transform it into $< M_1, x, 1^{p(|x|)} >$. 

[ ]
How to Show that a New Language is \textbf{NP}-Complete

- Show that the language $L_2$ is in \textbf{NP}.
- Give \textbf{P}-time reduction from \textbf{NP}-complete language $L_1$ to $L_2$.

\section*{Lemma}

\textit{The composition of two \textbf{P}-time reductions is a \textbf{P}-time reduction.}

\section*{Proof.}

Let $f : \Sigma_1^* \mapsto \Sigma_2^*$ and $g : \Sigma_2^* \mapsto \Sigma_3^*$ be two \textbf{P}-time reductions. Then $|f(x)|$ is a polynomial in $|x|$. Because $|g(f(x))|$ is a polynomial in $|f(x)|$, it is also a polynomial in $|x|$. Also, $x \in L_1$ if and only if $f(x) \in L_2$ if and only if $g(f(x)) \in L_3$. The statement of the lemma follows.
How to Show that a New Language is \textbf{NP}-Complete

\textbf{Theorem}

If there is a \textbf{P}-time reduction from an \textbf{NP}-complete language $L_1 \subseteq \Sigma^*_1$ to $L_2 \subseteq \Sigma^*_2$ and $L_2$ is in \textbf{NP}, $L_2$ is \textbf{NP}-complete.

\textbf{Proof.}

Let $x \in \Sigma^*_1$ and let $f : \Sigma^*_1 \mapsto \Sigma^*_2$ be a reduction from $L_1$ to $L_2$. Because $L_1$ is \textbf{NP}-complete, for every $L \in \textbf{NP}$ there is a \textbf{P}-time reduction from $L$ to $L_1$. If there is a \textbf{P}-time reduction from $L_1$ to $L_2$, then, by the lemma, there is a \textbf{P}-time reduction from $L$ to $L_2$. This establishes the second condition required of an \textbf{NP}-complete language. The first is that it be in \textbf{NP}, which is assumed.
Proof that CIRCUIT SAT is NP-Complete

**ACCEPT**

\[
\text{ACCEPT} = \{ \langle M, x, 1^t \rangle | M \text{ is a DTM, } x \in \Gamma^*, \exists y \in \Gamma^* \text{ such that } M \text{ accepts } \langle x, y \rangle \text{ in } t \text{ steps.} \}\]

**Approach**

- Reduce ACCEPT to CIRCUIT SAT and show latter in \textbf{NP}.
- Basic idea
  - DTM \( M \) executing \( p(|x|) \) steps on an input \( x \) is a DFSM.
  - Show that DFSM executing \( T \) steps has a circuit simulation.
- Simulate time-bounded DTM computation with efficient circuit.
Reduction from \textsc{Accept} to \textsc{Circuit Sat}

- $<\lfloor M_1 \rfloor, x, 1^t >$ is an instance of \textsc{Accept}.

- A verifier $V_{\text{accept}}$ is a poly-time DTM that is given an instance $y = <\lfloor M_1 \rfloor, x, 1^t >$ and a certificate $u$ on its tape. $V_{\text{accept}}$ accepts $y$ in time $q(|y|)$ for some polynomial $q(\cdot)$ if and only if $y \in \textsc{Accept}$.

- We show that $V_{\text{accept}}$ can be simulated by an instance of \textsc{Circuit Sat} that is constructed in poly-time. That is, we exhibit a poly-time reduction from an instance of \textsc{Accept} to one of \textsc{Circuit Sat}.

- The idea is to use \textit{configurations} of a TM (a verifier, e.g.). A configuration is string $(\sigma_1, \ldots, \sigma_{j-1}, (q, \sigma_j), \sigma_{j+1}, \ldots, \sigma_m)$ where $\sigma_i$ is the symbol in the $i$th tape cell, $q$ is a TM state. The notation signifies that TM head is over the $j$th tape cell.
Reduction from **ACCEPT** to **CIRCUIT SAT**

- A TM step generates a new configuration. Symbol under head may be changed and head moved right, left or not at all.
- A simple circuit simulates a cell, as indicated below. Here $T$ is the number of steps executed on an instance.
Reduction from ACCEPT to CIRCUIT SAT

- Reduction from instance of ACCEPT to one of CIRCUIT SAT can be done in polynomial time.
- Instance of CIRCUIT SAT is satisfiable (appropriate certificate given for ACCEPT) if and only if instance of ACCEPT is in language.
**SAT is NP-complete**

**SAT**

*Instance:* Literals $X = \{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$ and clauses $C = (c_1, c_2, \ldots, c_m)$ where each $c_i \subseteq X$.

*Answer:* “Yes” if for some assignment of Booleans to variables in $\{x_1, x_2, \ldots, x_n\}$, at least one literal in each clause has value 1.

**Theorem**

**SAT is NP-complete.**

**Proof.**

Earlier we showed that a straight-line program (SLP) (description of instance of CIRCUIT SAT) could be translated into instance of SAT poly-time on a DTM. Because CIRCUIT SAT is NP-complete and SAT is in NP, SAT is NP-complete.
3-SAT is **NP-complete**

3-SAT

**Instance:** Literals $X = \{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$ and clauses $C = (c_1, c_2, \ldots, c_m)$ where each $c_i \subseteq X$ **contains three literals.**

**Answer:** “Yes” if for some assignment of Booleans to variables in $\{x_1, x_2, \ldots, x_n\}$, at least one literal in each clause has value 1.

Theorem

3-SAT is **NP-complete.**

Proof.

The proof that SAT is **NP-complete** applies here.
Reducing SAT to 3-SAT

If we had shown SAT to be \( \text{NP} \)-complete with no restriction on the number of literals per clause, a reduction to 3-SAT exists, as illustrated below.

- Let a clause have \( k \geq 4 \) literals, \((l_1 \lor l_2 \lor \ldots \lor l_k)\). Replace by 
  \((l_1 \lor l_2 \lor n_2)(\bar{n}_2 \lor l_3 \lor n_3)(\bar{n}_3 \lor l_4 \lor n_4)\ldots(\bar{n}_{k-1} \lor l_k)\) where 
  \(n_2, n_3, \ldots, n_{k-1}\) are new Boolean variables.

- If the original clause is satisfiable with \( l_j = 1 \), there are assignments to the new variables so that all of the new clauses are satisfied. (What are they?)

- If original clause is not satisfied, for no assignment to new variables are all new clauses satisfied. (Why?)

- The reduction from one clause to \( k - 2 \) new clauses can be done in polynomial time. (Why?)
NAESAT is **NP-complete**

**NAESAT**

_**Instance:**_ An instance of 3-SAT.

_**Answer:**_ “Yes” if each clause is satisfiable when not all literals have the same value.

To show that **NAESAT** is **NP-complete**, we show how to reduce from **CIRCUIT SAT** to 3-SAT via a reduction that ensures that when all clauses are satisfied, there is at least one satisfied and one unsatisfied literal in each clause.
Proof that \textit{NAE-SAT} is \textbf{NP}-complete

Use the reduction from \textsc{circuit sat} to \textsc{3-sat}. Instead of representing each SLP step, such as $z = \text{AND}(x, y)$, $z = \text{OR}(x, y)$, and $z = \text{NOT}(x)$, in CNF, combine terms.

For example, the CNF of $z = \text{AND}(x, y)$ is 

$$(x \lor y \lor \bar{z}) \land (x \lor \bar{y} \lor \bar{z}) \land (\bar{x} \lor y \lor \bar{z}) \land (\bar{x} \lor \bar{y} \lor z).$$

Combining the first and second (first and third) terms gives $(x \lor \bar{z}) ((y \lor \bar{z}))$. Thus, $z = \text{AND}(x, y)$ is True if and only if $f_{\text{and}}(x, y, z) = (x \lor \bar{z}) \land (y \lor \bar{z}) \land (\bar{x} \lor \bar{y} \lor z)$ is satisfied.

Simplifying the CNF for $z = \text{OR}(x, y)$ gives $f_{\text{or}}(x, y, z) = (\bar{x} \lor z) \land (\bar{y} \lor z) \land (x \lor y \lor \bar{z})$. The CNF for $z = \text{NOT}(x)$ is $f_{\text{not}}(x, z) = (x \lor z) \land (\bar{x} \lor \bar{y})$. 
Proof that \textbf{NAESAT} is \textbf{NP}-complete

In polynomial time produce clauses $C$ by reducing an SLP having \textbf{AND}, \textbf{OR} and \textbf{NOT} using $f_{\text{and}}(x, y, z)$, $f_{\text{or}}(x, y, z)$, and $f_{\text{or}}(x, z)$. When each evaluates to True, the literals in all clauses are not all equal except for $(x \lor \bar{z})$ and $(y \lor \bar{z})$ in $f_{\text{and}}(x, y, z)$ and $(\bar{x} \lor z)$ $(\bar{y} \lor z)$ in $f_{\text{or}}(x, y, z)$. To each such clause add literal $w$ not found in other clauses, giving clauses $C'$. If clauses $C$ are satisfiable, clauses $C'$ are satisfiable with not all literals equal. To show this, let $w$ be False.

If the clauses $C'$ are satisfiable, they are satisfiable when variables are complemented. Thus, for some assignment $w$ is False. This assignment satisfies $C$. Thus, $C$ is satisfiable if and only if $C'$ is satisfiable.