A Corollary
The construction in the above proof shows the following.

**Corollary 31** If $L \in \text{TIME}(T(n))$, then a circuit with $O(T^2(n))$ gates can decide if $x \in L$ for $|x| = n$.

**MONOTONE CIRCUIT VALUE**

- A monotone boolean circuit’s output cannot change from true to false when one input changes from false to true.
- Monotone boolean circuits are hence less expressive than general circuits as they can compute only monotone boolean functions.
  - Monotone circuits do not contain $\neg$ gates.
- **MONOTONE CIRCUIT VALUE** is **CIRCUIT VALUE** applied to monotone circuits.

**MONOTONE CIRCUIT VALUE is P-Complete**

Despite their limitations, **MONOTONE CIRCUIT VALUE** is as hard as **CIRCUIT VALUE**.

**Corollary 32** **MONOTONE CIRCUIT VALUE** is **P-complete**.

- Given any general circuit, we can “move the $\neg$’s downwards” using de Morgan’s laws. (Think!)

**Cook’s Theorem: the First NP-Complete Problem**

**Theorem 33** *(Cook (1971))* **SAT** is **NP-complete**.

- **SAT** $\in$ **NP** (p. 80).
- **CIRCUIT SAT** reduces to **SAT** (p. 214).
- Now we only need to show that all languages in **NP** can be reduced to **CIRCUIT SAT**.
The Proof (continued)

- Let single-string NTM $M$ decide $L \in \text{NP}$ in time $n^k$.
- Assume $M$ has exactly two nondeterministic choices at each step: choices 0 and 1.
- For each input $x$, we construct circuit $R(x)$ such that $x \in L$ if and only if $R(x)$ is satisfiable.
- A sequence of nondeterministic choices is a bit string $B = (c_0, c_1, \ldots, c_{|x|} + 1) \in \{0, 1\}^{|x|}$.
- Once $B$ is fixed, the computation is deterministic.

---

The Computation Tableau for NTMs and $R(x)$

---

The Proof (continued)

- Each choice of $B$ results in a deterministic polynomial-time computation, hence a table like the one on p. 239.
- Each circuit $C$ at time $i$ has an extra binary input $c$ corresponding to the nondeterministic choice: $C(T_i, t_i, 1; T_i, t_i, 1, j + 1; c) = T_{ij}.$

---

The Proof (concluded)

- The overall circuit $R(x)$ (on p. 246) is satisfiable if there is a truth assignment $B$ such that the computation table accepts.
- This happens if and only if $M$ accepts $x$, i.e., $x \in L$. 

---
Parsimonious Reductions

- The reduction $R$ in Cook’s theorem (p. 243) is such that
  - Each satisfying truth assignment for circuit $R(x)$ corresponds to an accepting computation path for $M(x)$.
- The number of satisfying truth assignments for $R(x)$ equals that of $M(x)$’s accepting computation paths.
- This kind of reduction is called **parsimonious**.
- We will loosen the requirement for parsimonious reduction: It runs in deterministic polynomial time.

An Alternative Characterization of NP

**Proposition 34 (Edmonds (1965))** Let $L \subseteq \Sigma^*$ be a language. Then $L \in \text{NP}$ if and only if there is a polynomially decidable and polynomially balanced relation $R$ such that

$$L = \{ x : \exists y (x, y) \in R \}.$$

- Suppose such an $R$ exists.
- $L$ can be decided by this NTM:
  - On input $x$, the NTM guesses a $y$ of length $\leq |x|^k$ and tests if $(x, y) \in R$ in polynomial time.
  - It returns “yes” if the test is positive,

Two Notions

- Let $R \subseteq \Sigma^* \times \Sigma^*$ be a binary relation on strings.
- $R$ is called **polynomially decidable** if
  $$\{ x, y : (x, y) \in R \}$$
  is in $\text{P}$.
- $R$ is said to be **polynomially balanced** if $(x, y) \in R$ implies $|y| \leq |x|^k$ for some $k \geq 1$.

The Proof (concluded)

- Now suppose $L \in \text{NP}$.
- NTM $N$ decides $L$ in time $|x|^k$.
- Define $R$ as follows: $(x, y) \in R$ if and only if $y$ is the encoding of an accepting computation of $N$ on input $x$.
- Clearly $R$ is polynomially balanced because $N$ is polynomially bounded.
- $R$ is polynomially decidable because it can be efficiently verified by checking with $N$’s transition function.
- Finally $L = \{ x : (x, y) \in R \text{ for some } y \}$ because $N$ decides $L$. 
Comments

- Any "yes" instance $x$ of an NP problem has at least one succinct certificate or polynomial witness $y$.
- "No" instances have none.
- Certificates are short and easy to verify.
  - An alleged satisfying truth assignment for SAT; an alleged Hamiltonian path for HAMILTONIAN PATH.
- Certificates may be hard to generate (otherwise, NP equals P), but verification must be easy.
- NP is the class of easy-to-verify (in P) problems.

3 SAT

- $k$-SAT, where $k \in \mathbb{Z}^+$, is the special case of SAT.
- The formula is in CNF and all clauses have exactly $k$ literals (repetition of literals is allowed).
- For example,
  $$(x_1 \vee x_2 \vee \neg x_3) \land (x_1 \vee x_1 \vee \neg x_2) \land (x_1 \vee \neg x_2 \vee \neg x_3).$$

You Have an NP-Complete Problem (for Your Thesis)

- From Propositions 28 (p. 223) and Proposition 29 (p. 225), it is the least likely to be in P.
- Your options are:
  - Approximations.
  - Special cases.
  - Average performance.
  - Randomized algorithms.
  - Exponential-time algorithms that work well in practice.
  - "Heuristics" (and pray).

3 SAT Is NP-Complete

- Recall Cook's Theorem (p. 243) and the reduction of CIRCUIT SAT to SAT (p. 214).
- The resulting CNF has at most 3 literals for each clause.
  - This shows that 3SAT where each clause has at most 3 literals is NP-complete.
- Finally, duplicate one literal once or twice to make it a 3SAT formula.
- Note: The overall reduction remains parsimonious.
Another Variant of 3SAT

Proposition 35 3SAT is NP-complete for expressions in which each variable is restricted to appear at most three times, and each literal at most twice. (3SAT here requires only that each clause has at most 3 literals.)

- Consider a general 3SAT expression in which $x$ appears $k$ times.
- Replace the first occurrence of $x$ by $x_1$, the second by $x_2$, and so on, where $x_1, x_2, \ldots, x_k$ are $k$ new variables.

2SAT and Graphs

- Let $\phi$ be an instance of 2SAT: Each clause has 2 literals.
- Define graph $G(\phi)$ as follows:
  - The nodes are the variables and their negations.
  - Add edges $(\neg\alpha, \beta)$ and $(\neg\beta, \alpha)$ to $G(\phi)$ if $\alpha \lor \beta$ is a clause in $\phi$.
    - For example, if $x \lor \neg y \in \phi$, add $(\neg x, \neg y)$ and $(y, x)$.
    - Two edges are added for each clause.
  - Think of the edges as $\neg\alpha \Rightarrow \beta$ and $\neg\beta \Rightarrow \alpha$.
  - $b$ is reachable from $a$ iff $\neg a$ is reachable from $\neg b$.
  - Paths in $G(\phi)$ are valid implications.

The Proof (concluded)

- Add $(\neg x_1 \lor x_2) \land (\neg x_2 \lor x_3) \land \cdots \land (\neg x_k \lor x_1)$ to the expression.
  - This is logically equivalent to $x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_k \Rightarrow x_1$.
  - Each clause may have fewer than 3 literals.
- The resulting equivalent expression satisfies the condition for $x$. 

Illustration: Directed Graph for $(x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor x_2) \land (x_2 \lor x_3)$
Properties of $G(\phi)$

**Theorem 36** $\phi$ is unsatisfiable if and only if there is a variable $x$ such that there are paths from $x$ to $\neg x$ and from $\neg x$ to $x$ in $G(\phi)$.

We skip the proof in the text.

---

**Generalized 2SAT: MAX2SAT**

- Consider a 2SAT expression.
- Let $K \in \mathbb{N}$.
- MAX2SAT is the problem of whether there is a truth assignment that satisfies at least $K$ of the clauses.
- MAX2SAT becomes 2SAT when $K$ equals the number of clauses.
- MAX2SAT is an optimization problem.
- MAX2SAT $\in$ NP: Guess a truth assignment and verify the count.

---

**2SAT is in NL $\subseteq$ P**

- NL is a subset of P (p. 185).
- By Corollary 26 on p. 202, coNL equals NL.
- We need to show only that recognizing unsatisfiable expressions is in NL.
- In nondeterministic logarithmic space, we can test the conditions of Theorem 36 by guessing a variable $x$ and testing if $\neg x$ is reachable from $x$ and if $\neg x$ can reach $x$.
  - See the algorithm for REACHABILITY (p. 92).

---

**MAX2SAT is NP-Complete**

- Consider the following 10 clauses:

  $$(x) \land (y) \land (z) \land (w)$$

  $$(\neg x \lor \neg y) \land (\neg y \lor \neg z) \land (\neg z \lor \neg x)$$

  $$(x \lor \neg w) \land (y \lor \neg w) \land (z \lor \neg w)$$

- Let the 2SAT formula $r(x, y, z, w)$ represent the conjunction of these clauses.
- How many clauses can we satisfy?
- The clauses are symmetric with respect to $x$, $y$, and $z$.

---

\(^{a}\)Garey, Johnson, and Stockmeyer (1976).
The Proof (continued)

All of \( x, y, z \) are true: By setting \( w \) to true, we can satisfy 
\[ 4 + 0 + 3 = 7 \] clauses.

Two of \( x, y, z \) are true: By setting \( w \) to true, we can satisfy 
\[ 3 + 2 + 2 = 7 \] clauses.

One of \( x, y, z \) is true: By setting \( w \) to false, we can satisfy 
\[ 1 + 3 + 3 = 7 \] clauses.

None of \( x, y, z \) is true: By setting \( w \) to false, we can satisfy 
\[ 0 + 3 + 3 = 6 \] clauses, whereas by setting \( w \) to true, we can satisfy only 
\[ 1 + 3 + 0 = 4 \] clauses.

The Proof (continued)

- Any truth assignment that satisfies \( x \lor y \lor z \) can be extended to satisfy 7 of the 10 clauses and no more.
- Any other truth assignment can be extended to satisfy only 6 of them.
- The reduction from 3SAT \( \phi \) to \( \text{MAX2SAT} \) \( R(\phi) \):
  - For each clause \( C_i = (\alpha \lor \beta \lor \gamma) \) of \( \phi \), add group
    \( r(\alpha, \beta, \gamma, w_i) \) to \( R(\phi) \).
  - If \( \phi \) has \( m \) clauses, then \( R(\phi) \) has \( 10m \) clauses.
- Set \( K = 7m \).

The Proof (concluded)

- We now show that \( K \) clauses of \( R(\phi) \) can be satisfied if
  and only if \( \phi \) is satisfiable.
- Suppose \( 7m \) clauses of \( R(\phi) \) can be satisfied.
  - 7 clauses must be satisfied in each group because
    each group can have at most 7 clauses satisfied.
  - Hence all clauses of \( \phi \) must be satisfied.
- Suppose all clauses of \( \phi \) are satisfied.
  - Each group can set its \( w_i \) appropriately to have 7
    clauses satisfied.

NAESAT

- The NAESAT (for “not-all-equal” SAT) is like 3SAT.
- But we require additionally that there be a satisfying
  truth assignment under which no clauses have the three
  literals equal in truth value.
  - Each clause must have one literal assigned true and
    one literal assigned false.
NAESAT is NP-Complete\textsuperscript{a}

- Recall the reduction of \textsc{circuit sat} to \textsc{sat} on p. 214.
- It produced a CNF $\phi$ in which each clause has at most 3 literals.
- Add the same variable $z$ to all clauses with fewer than 3 literals to make it a $3\text{Sat}$ formula.
- Goal: The new formula $\phi(z)$ is \textsc{nae}-satisfiable if and only if the original circuit is satisfiable.

\textsuperscript{a}Karp (1972).

The Proof (concluded)

- Suppose there is a truth assignment that satisfies the circuit.
  - Then there is a truth assignment $T$ that satisfies every clause of $\phi$.
  - Extend $T$ by adding $T(z) = \text{false}$ to obtain $T'$.
  - $T'$ satisfies $\phi(z)$.
  - So in no clauses are all three literals false under $T'$.
  - Under $T'$, in no clauses are all three literals true.
  - Review the detailed construction on p. 215 and p. 216.

The Proof (continued)

- Suppose $T$ \textsc{nae}-satisfies $\phi(z)$.
  - $T$ also \textsc{nae}-satisfies $\phi(z)$.
  - Under $T$ or $T'$, variable $z$ takes the value false.
  - This truth assignment must still satisfy all clauses of $\phi$.
  - So it satisfies the original circuit.

Undirected Graphs

- An \textbf{undirected graph} $G = (V, E)$ has a finite set of nodes, $V$, and a set of undirected edges, $E$.
- It is like a directed graph except that the edges have no directions and there are no self-loops.
- We use $[i, j]$ to denote the fact that there is an edge between node $i$ and node $j$. 
Independent Sets

- Let $G = (V, E)$ be an undirected graph.
- $I \subseteq V$.
- $I$ is independent if whenever $i, j \in I$, there is no edge between $i$ and $j$.
- The independent set problem: Given an undirected graph and a goal $K$, is there an independent set of size $K$?
  - Many applications.

Reduction from 3SAT to INDEPENDENT SET

- Let $\phi$ be an instance of 3SAT with $m$ clauses.
- We will construct graph $G$ (with constraints as said) with $K = m$ such that $\phi$ is satisfiable if and only if $G$ has an independent set of size $K$.
- There is a triangle for each clause with the literals as the nodes.
- Add additional edges between $x$ and $\neg x$ for every variable $x$.

INDEPENDENT SET is NP-Complete

- This problem is in NP: Guess a set of nodes and verify that it is independent and meets the count.
- If a graph contains a triangle, any independent set can contain at most one node of the triangle.
- We consider graphs whose nodes can be partitioned in $m$ disjoint triangles.
  - If the special case is hard, the original problem must be at least as hard.

A Sample Construction

\[
(x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3).
\]
The Proof (continued)

- Suppose $G$ has an independent set $I$ of size $K = m$.
  - An independent set can contain at most $m$ nodes, one from each triangle.
  - An independent set of size $m$ exists if and only if it contains exactly one node from each triangle.
  - Truth assignment $T$ assigns true to those literals in $I$.
  - $T$ is consistent because contradictory literals are connected by an edge, hence not both in $I$.
  - $T$ satisfies $\phi$ because it has a node from every triangle, thus satisfying every clause.

Other INDEPENDENT SET-Related NP-Complete Problems

**Corollary 37** 4-degree independent set is NP-complete.

**Theorem 38** INDEPENDENT SET is NP-complete for planar graphs.

The Proof (concluded)

- Suppose a satisfying truth assignment $T$ exists for $\phi$.
  - Collect one node from each triangle whose literal is true under $T$.
  - The choice is arbitrary if there is more than one true literal.
  - This set of $m$ nodes must be independent by construction.
  * Literals $x$ and $\neg x$ cannot be both assigned true.

**CLIQUE and NODE COVER**

- We are given an undirected graph $G$ and a goal $K$.
- **CLIQUE** asks if there is a set of $K$ nodes that form a clique, which have all possible edges between them.
- **NODE COVER** asks if there is a set $C$ with $K$ or fewer nodes such that each edge of $G$ has at least one of its endpoints in $C$. 
**CLIQUE Is NP-Complete**

**Corollary 39** \textit{clique is NP-complete.}

- Let $G$ be the \textit{complement} of $G$, where $[x, y] \in G$ if and only if $[x, y] \notin G$.
- $I$ is a clique in $G \iff I$ is an independent set in $G$.

![Diagram](clique.png)

**MIN CUT and MAX CUT**

- A \textit{cut} in an undirected graph $G = (V, E)$ is a partition of the nodes into two nonempty sets $S$ and $V - S$.
- The size of a cut $(S, V - S)$ is the number of edges between $S$ and $V - S$.
- \textit{MIN CUT} is in \textit{P} by the maxflow algorithm.
- \textit{MAX CUT} asks if there is a cut of size at least $K$.
  - $K$ is part of the input.

![Diagram](min_cut.png)

**NODE COVER Is NP-Complete**

**Corollary 40** \textit{node cover is NP-complete.}

- $I$ is an independent set of $G = (V, E)$ if and only if $V - I$ is a node cover of $G$.

![Diagram](node_cover.png)