Circuit Satisfiability

![AND gate](image1) ![OR gate](image2) ![NOT gate](image3)

Boolean Combinational Circuits

\[
x_1 \rightarrow !x_1 \rightarrow x_1 \\
\]

not satisfiable for any \( x_1 \in \{0,1\} \)

\[
x_1 \rightarrow x_1 \vee x_2 \\
\]

satisfiable for \( x_1 x_2 = 01, 10, \) or \( 11 \), thus satisfiable

Circuit-Satisfiability Problem

Given a boolean combination circuit composed of AND, OR, and NOT gates, is it satisfiable?

\[
\text{CIRCUIT-SAT} = \{ \langle C \rangle \mid \text{C is a satisfiable boolean combinational circuit} \}
\]

where \( \langle C \rangle \) is a binary-string encoding of the circuit (e.g., as a graph)

Determining membership in CIRCUIT-SAT would require checking the \( 2^k \) possible binary assignments to the \( k \) inputs of a circuit.

There is strong evidence that CIRCUIT-SAT \( \not\in P \).
CIRCUIT-SAT is NP-Complete

Proof:

1. CIRCUIT-SAT ∈ NP
   Proof: Can verify an input assignment satisfies a circuit by computing the output of a finite number of gates, one of which will be the output of the circuit. This can be done in polynomial time. Thus, by definition of NP, CIRCUIT-SAT ∈ NP.

2. CIRCUIT-SAT ∈ NP-Hard
   I.e., L ≤ₚ CIRCUIT-SAT for every L ∈ NP
   Proof: Complex
   Show that any problem in NP can be computed using a boolean combination circuit (i.e., a computer).
   This circuit has a polynomial number of elements and can be constructed in polynomial time. Thus, L ≤ₚ CIRCUIT-SAT for all L ∈ NP.
   Thus, CIRCUIT-SAT ∈ NP-Hard.

CIRCUIT-SAT is NP-Complete
Proof by Cook, 1971

NP-Completeness Proofs

Lemma 36.8
If \( L \) is a language such that \( L' \leq_P L \) for some \( L' \in \text{NPC} \), then \( L \) is NP-Hard. If also \( L \in \text{NP} \), then \( L \in \text{NPC} \).

**Strategy for proving \( L \in \text{NP} \)**

1. Prove \( L \in \text{NP} \) (poly-time verifiable)
2. Select \( L' \in \text{NPC} \)
3. Describe poly-time algorithm computing a function \( f \) that maps instances of \( L' \) to instances of \( L \)
4. Prove that \( x \in L' \) iff \( f(x) \in L \) for all \( x \in \{0,1\}^* \).

**Note:** Showing \( L' \leq_P \text{spec}(L) \) implies \( L' \leq_P L \).

---

**Example: Boolean Formula Satisfiability**

**SAT:** Given a Boolean formula in Conjunctive Normal Form (C.N.F.), does there exist a satisfying assignment?

\[
\text{SAT} = \{ \ B : \ B \text{ is a boolean formula in CNF that is satisfiable by some truth assignment to its variables} \}
\]

A CNF formula is a boolean formula composed of variables and connectives AND, OR, NOT, IMPLIES, and EQUIV, possibly separated by parentheses.

Let \( B = (u_1 \lor \overline{u}_2) \land (\overline{u}_1 \lor u_2) \).

This is an *instance* of SAT for which the answer is “yes”. A satisfying truth assignment is given by \( t(u_1) = t(u_2) = T \).

On the other hand, the expression \( u_1 \land \overline{u}_1 \) is an instance of SAT for which the answer is “no”.

---

3
SAT ∈ NPC

Proof:

1. SAT ∈ NP
   Replace each variable with 0 or 1 as specified by the certificate and evaluate (poly-time).

2. Select L’ = CIRCUIT-SAT

3. Reduction from CIRCUIT-SAT to SAT.
   Straight-forward technique of computing the formula of each gate output as a combination of the input formulae may cause exponential instantiations of a variable as outputs are copied to multiple inputs. Instead, let each gate output be a variable. AND the output variable with expressions for each gate describing the equivalence between the gate’s output and input variables.

Example

Circuit C

\[
\begin{align*}
x_1 & \quad x_2 \quad x_4 \quad x_4 \\
x_3 & \quad x_5 & \quad x_6
\end{align*}
\]

Formula

\[
\phi = x_6 \land (x_4 \leftrightarrow (x_1 \land x_2)) \land (x_5 \leftrightarrow \neg x_3) \land (x_6 \leftrightarrow (x_4 \lor x_5))
\]

Constructing this formula takes polynomial time.
SAT $\in$ NPC

4. Prove $x \in L'$ iff $f(x) \in L$, where $L'$ is CIRCUIT-SAT, $L$ is SAT, and $f$ is the construction above.

$$x \in \text{CIRCUIT-SAT} \rightarrow f(x) \in \text{SAT}$$

If $C$ has a satisfying assignment, then each wire is well-defined and the output is 1.

Therefore, each conjunct of $\phi$ is 1, and $\phi$ will evaluate to 1.

A satisfying assignment to $\phi$ yields a valid circuit $C$ whose output is 1.

---

**CNF Satisfiability**

When the full power of SAT is not required to prove a language is in NPC, 3-CNF provides a more constrained alternative.

$k$-CNF (Conjunctive Normal Form) is a formula having a conjunction of clauses, where each clause is a disjunction of exactly $k$ literals (variable or its negation).

Example 3-CNF: $(x_1 \lor x_2 \lor x_3) \land (x_4 \lor \neg x_2 \lor x_4)$

**Theorem 36.10**

3-CNF-SAT $\in$ NPC
Some NP-Complete Problems

Clique Problem

**CLIQUE**: Given graph $G = (V, E)$, find largest subset $V' \subseteq V$ such that $\forall u, v \in V'$, $(u, v) \in E$.

I.e., $V'$ forms a complete subgraph of $G$ (usually want largest).

$\text{CLIQUE} = \{ \langle G, k \rangle : \exists V' \subseteq V \text{ of size } \geq k \text{ and } \forall u, v \in V', (u, v) \in E \}$

Example
The running time of the CLIQUE algorithm is $\Omega\left(k^2 \binom{|V|}{k}\right)$.

**Theorem 36.11: CLIQUE $\in$ NPC**

1. **CLIQUE $\in$ NP**
   To show CLIQUE in NP, we use set $V'$ of vertices as a certificate.
   Verifying is polynomial time, check whether for every pair $u,v$ in $V'$, the edge is in $E$ ($|V'|^2$ pairs).

2. **$L' = 3$-CNF-SAT**

3. **$3$-CNF-SAT $\leq_P$ CLIQUE**
Start with instance of 3-CNF-SAT (also called 3CNF). Let f be 3CNF with k clauses, \((C_{11} \lor C_{12} \lor C_{13}) \land (C_{21} \lor C_{22} \lor C_{23}) \land (C_{31} \lor C_{32} \lor C_{33}) \land \ldots (C_{k1} \lor C_{k2} \lor C_{k3})\).

For \(r = 1, 2, \ldots, k\), each clause has three distinct literals \(l^r_1, l^r_2, l^r_3\).

Construct a graph \(G\) such that \(f\) is satisfiable iff \(G\) has a clique of size \(k\).

For each \(C_r\) in \(f\), put triple of vertices \(v^r_1, v^r_2, v^r_3\) in \(V\).
Add edge \((v^r_i, v^r_j)\) if

1. \(v^r_i\) and \(v^r_j\) are in different triples \((r \neq s)\), and
2. their corresponding literals are consistent \((l^r_i\) is not the negation of \(l^s_j))\.

---

**Proof (cont.)**

This graph is constructed in polynomial time.

If \(f = (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)\), then the graph is

![Graph Diagram]

---

| 8 |
Proof (cont.)

4. Is this a reduction?

Suppose f has a satisfying assignment. Then each clause \( C_r \) contains at least one literal \( l_i^r \) that is assigned 1, and each such literal corresponds to a vertex \( v_i^r \).

Picking one such ”true” literal from each clause yields a set \( V' \) of k vertices.

Is \( V' \) a clique? For any two vertices \( v_i^r, v_j^s \), \( r \neq s \), the corresponding literals are mapped to 1 by the satisfying assignment and thus the literals cannot be complements. By the construction of \( G \), the edge \( (v_i^r, v_j^s) \) belongs in \( E \).

Proving the other direction, if \( G \) has a clique \( V' \) of size k, no edges in \( G \) connect vertices in the same triple, so \( V' \) contains exactly one vertex per triple. Assign a 1 to each literal \( l_i^r \) such that \( v_i^r \) in \( V' \) without fear of assigning 1 to a literal and its complement. Each clause is satisfied, and \( f \) is satisfied.

Vertex-Cover Problem

A vertex cover of an undirected graph \( G = (V, E) \) is a subset \( V' \subseteq V \) such that each edge in \( E \) is incident on at least one of the vertices in \( V' \).

\[
VC = \{ (G, k) : G = (V, E) \text{ is a graph, and } \exists V' \subseteq V \text{ such that } |V'| \leq k \text{ and } \forall (u, v) \in E, \text{ either } u \in V' \text{ or } v \in V' \text{ (or both) } \}
\]
Original Graph \( G \)

\[
\begin{array}{c}
\text{K = 1} & \text{NONE} \\
\text{K = 2} & \text{NONE} \\
\text{K = 3} & \text{NONE} \\
\text{K = 4} & \\
\end{array}
\]

**Vertex Cover**

\[
\text{VC} = \{ (G, k) \mid \text{graph } G \text{ has vertex cover of size } k \}
\]

**Theorem 36.12:** \( \text{VC} \in \text{NPC} \)

Proof Sketch:

1. **VC \in NP**
   
   Given \( V' \), check \( |V'| = k \), and for each edge \((u,v) \in E\), check that either \( u \in V' \) or \( v \in V' \).

2. **L' = CLIQUE**

3. **CLIQUE \leq_P VC**
   
   If graph \( G = (V, E) \) has clique \( V' \), then graph \( \overline{G} \) has vertex cover \( V - V' \).

   \( \overline{G} = (V, \overline{E}) \) is the complement of \( G = (V, E) \), where \( \overline{E} = \{ (u, v) \mid (u, v) \notin E \} \).
\[ E \}
Reduction: \( G \rightarrow \overline{G} \) (poly-time)

4. \( x \in \text{CLIQUE}(G) = V' \rightarrow f(x) \in \text{VC}(G) = V - V' \) \(|V'| = k\)

Every edge \((u,v) \in E\) implies \((u,v) \notin E\), thus at least one of \(u\) and \(v\) \(\notin V'\). Thus, at least one of \(u,v\) belongs to \(V - V'\), which means edge \((u,v)\) is covered by \(V - V'\). Similar argument for other direction.

---

**Set-Covering Problem**

Given a finite set \(X\) and a family \(F\) of subsets of \(X\), \(X = \bigcup_{S \in F} S\), find a minimum-size subset \(C \subseteq F\) whose members cover all of \(X\).

\[
\text{SC} = \{ \langle X, F, k \rangle \mid \text{there exists a set cover } C \subseteq F \text{ covering } X \text{ with size } \leq k \}
\]

---

**Theorem:** \( \text{SC} \in \text{NPC} \)

Proof:

1. Given \(C\), check that all elements of \(X\) are members of some set in \(C\) and that \(|C| \leq k\).
2. \( L' = VC \)

3. Given \( \langle G, k \rangle \in VC \), define \( F \) such that each element of \( F \) is a subset for a vertex \( v \) in \( G \) containing \( v \) and all vertices reachable by an edge from \( v \).
   Let \( X = V \). Then \( \langle X, F, k \rangle \in SC \).

4. If \( C \) is the vertex cover of \( \langle G, k \rangle \in VC \), then every vertex \( u \) in \( G \) is incident from an edge \((u,v)\) where either \( u \in C \) or \( v \in C \). Thus all vertices will appear in some set in \( F \), and the sets in \( F \) corresponding to the vertices in \( C \) make up the set covering of \( \langle X, F, k \rangle \in SC \).

---

**Subset Sum Problem**

\( \text{SUBSET-SUM} = \{ \langle S, t \rangle \mid \text{there exists } S' \subseteq S \subseteq N \text{ such that } \sum_{s \in S'} s = t \in N \} \),

\( N = \) set of natural numbers

**Theorem 36.13:** \( \text{SUBSET-SUM} \in NPC \)

Proof:

1. \( \text{SUBSET-SUM} \in \text{NP} \). Just add up elements of \( S' \) and compare sum to \( t \).

2. \( L' = VC \)

3. \( VC \leq_P \text{SUBSET-SUM} \)

4. \( x \in VC \iff f(x) \in SS \)
   Proof is complex.
Hamiltonian Cycle Problem

A Hamiltonian Cycle is a simple cycle in a graph going through each vertex exactly once.

\[ \text{HC} = \{ \langle G \rangle | \text{G has a Hamiltonian cycle}\} \]

\[ \text{HC} \in \text{NPC} \]

Proof:

1. Done earlier.

2. \( L' = 3\text{-CNF-SAT} \)

3. \( 3\text{-CNF-SAT} \leq_p \text{HC} \)

4. \( x \in 3\text{-CNF-SAT} \iff f(x) \in \text{HC} \)
   
   Proof is complex.

---------------------------------------------------------------

Traveling Salesman Problem

Given a complete graph with weights on the edges, find a cycle of least total weight that visits each vertex exactly once.

Decision Problem:

\[ \text{TSP} = \{ \langle G, k \rangle | \text{G is a complete graph with weights on edges that contains a cycle of total weight } \leq k \text{ visiting each vertex exactly once}\} \]
**Theorem 36.15: TSP ∈ NPC**

Variant of proof in textbook.
Proof sketch:

1. **TSP ∈ NP**
   Given a tour, check that each vertex is visited exactly once and the sum of costs ≤ k

2. **L’ = HC**

3. **HC ≤ₚ TSP**
   Given graph $G = (V, E)$, transformation $f$ outputs complete graph with vertices $V$.
   Weights of edges = 1 if $e \in E$, or $(-V - 1)$ if $e \notin E$
   Also outputs the number $-V-$. 
   $f$ is clearly implementable in polynomial time.

4. Then there exists a tour in this complete graph of size ≤ $|V|$ iff there exists a Hamiltonian Cycle in original graph.
Partition Problem

Given a finite set $A$ and a “size” $s(a) \in \mathbb{Z}^+$ for each $a \in A$, find a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in (A-A')} s(a)$$

PARTITION $= \{ \langle A, s(a) \rangle : \exists A' \subseteq A$ such that the sums of $A'$ and $(A-A')$ are equal $\}$

For example, if $A = \{a = 1, b = 2, c = 3, d = 4, e = 5, f = 7, g = 8\}$, then one possible partition is $A' = \{a, b, c, d, e\}$ and $A - A' = \{f, g\}$. The sum of both subsets is 15.

Knapsack Problem

**KNAPSACK:** Given a finite set $U$, a “size” $s(u) \in \mathbb{Z}^+$ and a “value” $v(u) \in \mathbb{Z}^+$ for each $u \in U$, a size constraint $B \in \mathbb{Z}^+$, and a value goal $K \in \mathbb{Z}^+$, is there a subset $U' \subseteq U$ such that $\sum_{u \in U'} s(u) \leq B$ and $\sum_{u \in U'} v(u) \geq K$?

This can be seen as a knapsack, which has a size limit for the objects, as in the picture below.
The goal is to pick a collection of objects that will fit in the knapsack and whose total value is at least $K$ ($K$ is input)

$\text{KNAPSACK} = \{(U, s, v, B, K) : \exists \text{ subset } U' \text{ of } U \text{ such that the sum of } s \text{ values is at most } B, \text{ and the sum of } v \text{ values is at least } K\}$

---

**KNAPSACK is NP-Complete**

**Proof:** We will show that the KNAPSACK problem is NP-complete by polynomial-time restricting it in a way that makes it equal to the PARTITION problem, or $\text{PARTITION} \leq_P \text{spec(KNAPSACK)}$.

We can restrict KNAPSACK to PARTITION by allowing only instances in which $s(u) = v(u)$ for all $u \in U$ and $B = K = 1/2 \sum_{u \in U} s(u)$.

---

**NP-Complete Problems**