1. **4SAT** is the special case of SAT where every clause of the input CNF formula Φ contains exactly four distinct literals. Prove that 4SAT is NP-hard. *[Hint: Reduce from 3SAT.]*

Solution (one new variable per clause): We describe a polynomial-time reduction from 3SAT.

Given an arbitrary 3CNF formula Φ, we construct a 4CNF formula Φ' by replacing each clause $(a \vee b \vee c)$ with two clauses $(a \vee b \vee c \vee x) \wedge (a \vee b \vee c \vee \overline{x})$, where x is a new variable unique to that clause. Thus, if Φ has n variables and m clauses, then Φ' has $n + m$ variables and $2m$ clauses. I claim that

Φ is satisfiable if and only if Φ' is satisfiable.

$\Longrightarrow$ Suppose Φ is satisfiable. Then for some assignment of variables in Φ, every clause $(a \vee b \vee c)$ contains at least one true literal. It follows that with the same assignment, every clause $(a \vee b \vee c \vee x)$ or $(a \vee b \vee c \vee \overline{x})$ in Φ' contains at least one true literal, no matter what value we give x. We conclude that Φ' is satisfiable.

$\Longleftarrow$ Suppose Φ' is satisfiable. Then for some assignment of variables in Φ', every clause $(a \vee b \vee c \vee x)$ or $(a \vee b \vee c \vee \overline{x})$ contains at least one true literal. If $x = \text{TRUE}$, then the second clause implies that $(a \vee b \vee c)$ contains a true literal; on the other hand, if $x = \text{FALSE}$, then the first clause implies that $(a \vee b \vee c)$ contains a true literal. Thus, the same assignment (ignoring the new variables x) also satisfies Φ. We conclude that Φ is satisfiable.

We can easily construct Φ' from Φ in polynomial time. ■

Solution (only four new variables): We describe a polynomial-time reduction from 3SAT.

Given an arbitrary 3CNF formula Φ, we construct a 4CNF formula Φ' by introducing four new variables w, x, y, and z, as follows. First, we add the variable z to every clause in Φ, for example changing $(a \vee b \vee \overline{c})$ into $(a \vee b \vee \overline{c} \vee z)$. Then we add eight new clauses that force z to be FALSE in every satisfying assignment:

$$(w \vee x \vee y \vee \overline{z}) \wedge (\overline{w} \vee x \vee y \vee \overline{z}) \wedge (w \vee \overline{x} \vee y \vee \overline{z}) \wedge (\overline{w} \vee \overline{x} \vee y \vee \overline{z})$$
$$\wedge (w \vee x \vee \overline{y} \vee \overline{z}) \wedge (\overline{w} \vee x \vee \overline{y} \vee \overline{z}) \wedge (w \vee \overline{x} \vee \overline{y} \vee \overline{z}) \wedge (\overline{w} \vee \overline{x} \vee \overline{y} \vee \overline{z})$$

I claim that

Φ is satisfiable
if and only if
Φ' is satisfiable.

- Suppose Φ is satisfiable. Fix a satisfying assignment for Φ, and then extend this assignment by setting $z \leftarrow$ FALSE. Because every clause $(a \vee b \vee c)$ in Φ contains at least one true literal, the corresponding clause $(a \vee b \vee c \vee z)$ in Φ' also contains at least one true literal. Moreover, the remaining clauses all contain the true literal $\overline{z}$. We conclude that Φ' is satisfiable.
- Suppose Φ' is satisfiable; fix a satisfying assignment for Φ'. The final eight clauses of Φ' imply that the variable z must be false. Since every clause $(a \vee b \vee c \vee z)$ in Φ' contains at least one true literal, but that literal is not z, the corresponding clause $(a \vee b \vee c)$ in Φ must also contain a true literal. We conclude that Φ is satisfiable.

We can construct Φ' from Φ in polynomial time by brute force. ■

2. **BalancedSat** asks whether a given CNF formula Φ has a satisfying assignment where exactly half of the variables are True. Prove that BalancedSat is NP-hard.

Solution: We describe a polynomial-time reduction from 3Sat.

Let Φ be an arbitrary CNF formula with n variables and m clauses. Duplicate Φ using all new variables and invert every literal to obtain a new 3CNF formula $\overline{\Phi}$ with $2n$ variables and $2m$ clauses. For example, if $\Phi = (w \vee x \vee \overline{y}) \wedge (\overline{w} \vee y \vee \overline{z})$, then $\overline{\Phi} = (\overline{w}' \vee \overline{x}' \vee y') \wedge (w' \vee \overline{y}' \vee z')$. Finally, let $\Phi' = \Phi \wedge \overline{\Phi}$. I claim that

> Φ has a satisfying assignment
> if and only if
> Φ' has a balanced satisfying assignment.

$\Longrightarrow$ Suppose Φ is satisfiable; fix an arbitrary satisfying assignment with t True variables and f False variables. For each variable x in Φ, set the corresponding variable x' in $\overline{\Phi}$ to the opposite value. This gives us a satisfying assignment for $\overline{\Phi}$ with $t + f$ True variables and $f + t$ False variables.

$\Longleftarrow$ Suppose $\overline{\Phi}$ has a balanced satisfying assignment. Then in particular, this assignment satisfies every clause in Φ. Thus, Φ is satisfiable.

We can construct Φ' from Φ in polynomial time by brute force. ■

3. **EXACT3SAT** asks whether a given CNF formula Φ, with exactly three distinct literals per clause, has an assignment such that each clause contains *exactly* one TRUE literal. Prove that EXACT3SAT is NP-hard.

Solution (from 3SAT): Given an arbitrary 3CNF formula Φ, we construct a new 3CNF formula Φ' by converting each clause $(a \vee b \vee c)$ in Φ into three new clauses

$$(\overline{a} \vee w \vee x) \wedge (b \vee x \vee y) \wedge (\overline{c} \vee y \vee z),$$

where u, v, w, and x are new variables. Thus, if the input formula Φ has n variables and m clauses, then the new formula Φ' has exactly $n + 4m$ variables and $3m$ clauses. I claim that

Φ is satisfiable if and only if Φ' is exactly satisfiable.

$\Longrightarrow$ Suppose Φ is satisfiable, and fix a satisfying assignment. For each clause $(a \vee b \vee c)$ in Φ, we can exactly satisfy the corresponding triple of clauses in Φ' using the following rules, where $*$ means the value doesn't matter:

$$\begin{aligned}
(a,b,c) = (*,1,*) &\implies (w,x,y,z) = (a,0,0,c) \\
(a,b,c) = (1,0,1) &\implies (w,x,y,z) = (1,0,1,0) \\
(a,b,c) = (1,0,0) &\implies (w,x,y,z) = (0,1,0,0) \\
(a,b,c) = (0,0,1) &\implies (w,x,y,z) = (0,0,1,0)
\end{aligned}$$

Thus, Φ' is exactly satisfiable.

$\Longleftarrow$ On the other hand, suppose Φ' is exactly satisfiable, and fix an exactly satisfying assignment. Consider the clauses in Φ' corresponding to an arbitrary clause $(a \vee b \vee c)$ in Φ. If $a = 0$, then $w = x = 0$, because the first clause is satisfied exactly. Similarly, if $c = 0$, then $y = z = 0$, because the third clause is satisfied exactly. Finally, if $x = y = 0$, then $b = 1$, because the second clause is satisfied exactly. Thus, the clause $(a \vee b \vee c)$ is satisfied. We conclude that Φ is satisfiable.

Given the formula Φ, we can construct Φ' in polynomial (actually $O(n+m)$) time by brute force. ■

Solution (from 3COLOR, because it has a 3 in it): Given an undirected graph $G = (V, E)$, we construct a 3CNF boolean formula $\Phi(G)$ as follows.

- For each vertex v, the formula Φ includes three variables r_v, g_v, and b_v and one *vertex clause* $(r_v \vee g_v \vee b_v)$.
- For each edge uv, the formula Φ includes variables r_{uv}, g_{uv}, and b_{uv} and three *edge clauses*

$$(r_u \vee r_v \vee \overline{r_{uv}}) \wedge (g_u \vee g_v \vee \overline{g_{uv}}) \wedge (b_u \vee b_v \vee \overline{b_{uv}}).$$

Thus, if the graph G has n vertices and m edges, then $\Phi(G)$ has exactly $3n + 3m$ variables and $n + 3m$ clauses. Now I claim that

G is 3-colorable if and only if $\Phi(G)$ is exactly satisfiable.

$\Longrightarrow$ Suppose G is 3-colorable; color the vertices of G red, green, and blue so that every edge joins vertices of different colors. For each vertex v, set $r_v = 1$ if and only if v is red, $b_v = 1$ if and only if v is blue, and $g_v = 1$ if and only if v is green. Because v is assigned exactly one color, the vertex clause $(r_v \vee g_v \vee b_v)$ is exactly satisfied. For each edge uv, set

$$r_{uv} = r_u \vee r_v, \quad g_{uv} = g_u \vee g_v, \quad b_{uv} = b_u \vee b_v.$$

Because u and v have different colors, each of the edge clauses for uv are exactly satisfied. We conclude that $\Phi(G)$ is exactly satisfiable.

$\Longleftarrow$ Now suppose $\Phi(G)$ is exactly satisfiable; fix an exactly satisfying assignment. Color each vertex v in G red if $r_v = 1$, green if $g_v = 1$, and blue if $b_v = 1$. The vertex clause $(r_v \vee g_v \vee b_v)$ is exactly satisfied, so v is assigned exactly one color. For each edge uv, the edge clause $(r_u \vee r_v \vee \overline{r}_{uv})$ is exactly satisfied, so u and v cannot both be colored red; a similar argument applies to the other two colors. We conclude that G is 3-colorable.

Given the graph G, we can construct $\Phi(G)$ in $O(n + m)$ time by brute force. ■

Harder problems to think about later:

4. **(≤3,≤3)-SAT** is the special case of SAT where each clause of the input CNF formula Φ contains ***at most*** three literals, and each variable appears in at most ***three*** clauses. Prove that (≤3,≤3)-SAT is NP-hard.

Solution: We describe a polynomial-time reduction from 3SAT. Let Φ be an arbitrary 3CNF formula, with n variables $x_1, x_2, \ldots, x_n$ and m clauses, each with exactly three distinct literals. For each index i, let m_i denote the number of clauses that contain either x_i or $\overline{x_i}$. We immediately have $\sum_i m_i = 3m$.

For each variable x_i such that $m_i \geq 3$, we modify the formula Φ as follows.

- Introduce m_i new variables $x_{i,1}, x_{i,2}, \ldots, x_{i,m_i}$.
- Add m_i new two-literal clauses

$$(\overline{x_{i,1}} \vee x_{i,2}) \wedge (\overline{x_{i,2}} \vee x_{i,3}) \wedge \cdots \wedge (\overline{x_{i,m_i}} \vee x_{i,1})$$

- For all j from 1 to m_i, replace the jth occurrence of x_i with $x_{i,j}$.

The resulting formula Φ' has at most $n + 3m$ variables and at most $2m$ clauses.

Each original variable x_i appears in at most three clauses of Φ', and each new variable $x_{i,j}$ appears in exactly three clauses of Φ' (specifically, one three-literal clause and two two-literal clauses). Thus, Φ' is a valid instance of (≤3,≤3)-SAT.

To prove that this reduction is correct, we need to prove that

Φ is satisfiable if and only if Φ' is satisfiable.

$\Longrightarrow$ Suppose Φ is satisfiable; fix a satisfying assignment. For all indices i and j, set $x_{i,j} = x_i$.

- Every clause in Φ is satisfied. Thus, every three-literal clause in Φ' is also satisfied; we've only changed the variable names, not their values.
- Each two-literal clause $(\overline{x_{i,j}} \vee x_{i.j+1})$ is satisfied, because exactly one of those two literals is TRUE.

We conclude that Φ' is satisfiable.

$\Longleftarrow$ On the other hand, suppose Φ' is satisfiable; fix a satisfying assignment. For each original variable x_i that does not appear in Φ, set $x_i = x_{i,1}$.

- The two-variable clauses we added for x_i are logically equivalent to

$$(x_{i,1} \Rightarrow x_{i,2}) \wedge (x_{i,2} \Rightarrow x_{i,3}) \wedge \cdots \wedge (x_{i,m_i} \Rightarrow x_{i,1}).$$

All of these two-literal clauses are satisfied. it follows that the variables x_i, $x_{i,1}$, $x_{i,2}$, ..., x_{i,m_i} must all have the same value.

- Each three-literal clause in Φ' is satisfied. This, every clause in Φ is also satisfied; changing the variable names does not change their values.

We conclude that $x_1, x_2, \ldots, x_n$ is a satisfying assignment for Φ.

Given Φ, it is straightforward to construct Φ' in polynomial time by brute force. ■

Source: Craig Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics* 8:85–89, 1984.

5. **(3,≤4)-SAT** is the special case of SAT where each clause of the input CNF formula Φ contains ***exactly*** three distinct literals, and each variable appears in at most ***four*** clauses. Prove that (3,≤4)-SAT is NP-hard.[1] *[Hint: Reduce from (≤3,≤3)-SAT.]*

Solution (sketch): We describe a polynomial-time reduction from (≤3,≤3)-SAT. Let Φ be a CNF formula where each clause has at most three literals and each variable appears in at most three clauses.

If any clause contains only one literal, we can assign that variable the only value that satisfies that literal and simplify the formula. (Alternatively, we can assume that Φ is the output of our previous reduction from 3SAT.) Thus, without loss of generality, we can assume that every clause in Φ contains either two or three literals.

We transform Φ into a new 3CNF formula Φ' by replacing each two-literal clause $(\overline{u} \vee v)$ with fourteen three-literal clauses

$$\begin{aligned}
&(\overline{u} \vee v \vee \overline{z}) \wedge (\overline{w_1} \vee \overline{w_1} \vee \overline{w_1}) \\
&\wedge (z \vee x_1 \vee y_1) \wedge (w_1 \vee \overline{x_1} \vee y_1) \wedge (w_1 \vee x_1 \vee \overline{y_1}) \wedge (w_1 \vee \overline{x_1} \vee \overline{y_1}) \\
&\wedge (z \vee x_2 \vee y_2) \wedge (w_2 \vee \overline{x_2} \vee y_2) \wedge (w_2 \vee x_2 \vee \overline{y_2}) \wedge (w_2 \vee \overline{x_2} \vee \overline{y_2}) \\
&\wedge (z \vee x_3 \vee y_3) \wedge (w_3 \vee \overline{x_3} \vee y_3) \wedge (w_3 \vee x_3 \vee \overline{y_3}) \wedge (w_3 \vee \overline{x_3} \vee \overline{y_3}),
\end{aligned}$$

where w_1, w_2, w_3, x_1, x_2, x_3, y_1, y_2, y_3, and z are ten new variables unique to that clause. Thus, if the original formula Φ has n variables and m clauses, then the new formula Φ' has at most $n + 10m$ variables and at most $14m$ clauses.

Every clause in Φ' has exactly three literals. Each variable in Φ still appears at most three times in Φ', as do the new variables x_i and y_i, but the new variable z and w_i each appear four times. Thus, Φ' is a valid instance of (3,≤4)-SAT.

Brute-force case analysis implies that in every satisfying assignment *for these fourteen clauses* of Φ', we must have $z = \text{FALSE}$. It follows that the original formula Φ is satisfiable if and only if the new formula Φ' is satisfiable.

Given Φ, we can construct Φ' in polynomial time by brute force. ■

Source: Craig Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics* 8:85–89, 1984.

[1] The related problem (3,≤3)-SAT, where each clause in the input formula contains exactly three literals and each variable appears in at most ***three*** clauses, is trivial. All such formulae are satisfiable!

6. Describe a polynomial-time algorithm for XorSat!

Solution: Using the identity $(x \oplus \overline{y}) = \overline{(x \oplus y)}$, we can transform each clause into one of two standard forms: either the exclusive-or of one or more variables, or the negation of the exclusive-or of one or more variables:

$$(a \oplus b \oplus \cdots \oplus z) \quad \text{or} \quad \overline{(a \oplus b \oplus \cdots \oplus z)}$$

This transformation takes $O(nm)$ time, where n is the number of variables, and m is the number of clauses.

Choose an arbitrary variable x. If x or $\overline{x}$ is a clause by itself, assign the only possible value to x, simplify the formula, and recurse. Otherwise, if x appears in more than one clause, we can reduce the number of clauses that contain x using one of the following identities (where y and z are subclauses):

$$\begin{aligned}
(x \oplus y) \wedge (x \oplus z) &= (x \oplus y) \wedge \overline{(y \oplus z)} \\
(x \oplus y) \wedge \overline{(x \oplus z)} &= (x \oplus y) \wedge (y \oplus z) \\
\overline{(x \oplus y)} \wedge \overline{(x \oplus z)} &= \overline{(x \oplus y)} \wedge \overline{(y \oplus z)}
\end{aligned}$$

By applying these rules repeatedly, we can ensure that some variable x appears in exactly one nontrivial clause. Recursively compute a satisfying assignment for all the other clauses. (If the other clauses cannot be satisfied, neither can the original formula.) The values of the other variables determine a unique value for x that satisfies its clause. The recursion bottoms out either when the formula is empty (and therefore vacuously satisfied) or contains both a variable x and its negation $\overline{x}$ as clauses (so the formula is not satisfiable).

Suppose the formula has m clauses and n variables. For each variable, we apply the simplification rules at most m times (once for each clause), and each application takes $O(n)$ time (constant time per variable). Once we have a satisfying assignment for the other variables, we can compute an appropriate value for x in $O(n)$ time. Thus, the total running time of the algorithm is $O(n^2m)$. ■

Solution (same as above but in different words): Any XCNF formula can be interpreted as a set of linear equations over the field $\mathbb{Z}_2$ of integers modulo 2 as follows. Each variable is either 0 (False) or 1 (True), and the negation $\overline{x}$ of any variable x has value $1 - x = (1 + x) \bmod 2$. A clause is satisfied if and only if it contains an odd number of true literals, or equivalently, if the sum of the numerical values of the literals is equivalent to 1 (mod 2). The entire formula is satisfiable if and only if the resulting system of mod-2 linear equations has a solution.

We can solve the system of linear equations using Gaussian elimination, exactly as we would solve a system of linear equations over the reals, only using mod-2 arithmetic everywhere. (This works precisely because the set $\{0, 1\}$ with mod-2 arithmetic is a field.) Given a system of m equations (clauses) in n unknowns (variables), Gaussian elimination requires $O(n^2m)$ time. ■