Theorem 1. *3CSP is $\mathcal{NP}$-complete.*

Proof. It is easy to see that 3CSP is in $\mathcal{NP}$ (convince yourself of that by constructing an appropriate verifier!), and we focus on showing $\mathcal{NP}$-hardness.

Given a Boolean circuit $C\colon \{0,1\}^m \to \{0,1\}$ of description size n, we construct in polynomial time a list $\mathcal{L}$ of 3-constraints over $\bar{m} \leq n$ variables such that

$$\exists x \in \{0,1\}^m : C(x) = 1 \iff \exists x \in \{0,1\}^{\bar{m}} : \forall c \in \mathcal{L}, \quad c(x) = 1 \,.$$

Let s be the number of non-input nodes in C. The variables for $\mathcal{L}$ are $x_1, ..., x_m, z_1, ..., z_s$ (so $\bar{m} = m + s \leq n$), where we think of each z_i as associated with a node in C.

High-level idea:

- The list will be satisfied by $x_1, ..., x_m, z_1, ..., z_s$ if and only if $C(x_1, ..., x_m) = 1$ and each z_i is the value of node i in $C(x_1, ..., x_m)$.
- To enforce this, we add constraints as follows:
  - **Output constraint:** One constraint is that $z_s = 1$, which we intuitively think of as "the output gate of $C(x)$ is 1".
  - **Consistency constraints:** For each node i in C, we add a constraint on z_i and on the two variables $z_j, z_{j'}$ representing nodes feeding into node i. The constraint is that z_i is correctly computed from $z_j, z_{j'}$, according to the gate-type of node i. For example, for an AND node, we add a constraint $z_i = z_j \wedge z_{j'}$.
- The key point is that each constraint only involves at most three variables: the variable representing node i, and the variable/s representing the node/s feeding into node i (there are at most two).

Given C of description size n, we construct a list $\mathcal{L}$ as above:

- There are $\bar{m} = m + s$ variables for $\mathcal{L}$, denoted $x_1, ..., x_m, z_1, ..., z_s$.
- We add a constraint "$z_s = 1$".
- We iterate over nodes $i = 1, ..., s$ in the description of C. For each node i, let f_i be the function it computes, let $child(i)_1$ be the variable representing the first node feeding into i (i.e., $child(i) = x_j$ for some $j \in [m]$ or $child(i) = z_j$ for some $j \in [s]$), and let $child(i)_2$ be defined accordingly.[1] We add a constraint on the variables $z_i, child(i)_1, child(i)_2$ asserting that $z_i = f_i(child(i)_1, child(i)_2)$.[2]

[1] If i is a NOT node then $child(i)_1 = child(i)_2$.

[2] To formally type-match with the definition of a 3-constraint, convince yourself that the constraint $z_i = f_i(z_j, z_{j'})$ can be expressed as a function $g\colon \{0,1\}^3 \to \{0,1\}$.

Note that the procedure above is implementable in time $\text{poly}(n)$, since there are at most n nodes, and for each $i \in [n]$ we can parse the description of C to find the nodes feeding into node i (and the function f_i) in time $\text{poly}(n)$.

Turning to correctness, we need to prove that $\mathcal{L} \in 3CSP$ if and only if $C \in CircuitSAT$. (As usual, both directions are crucial.)

- If $C \in CircuitSAT$ then there is x such that $C(x) = 1$. Then, there is an assignment to $x_1, ..., x_n, z_1, ..., z_s$ such that all constraints are satisfied, namely assigning x to the x-variables and assigning the gate values of $C(x)$ to the z-variables.

  To see that this assignment satisfies all constraints, note that $z_s = 1$, because z_s gets the value of the output node of $C(x) = 1$; and every consistency constraint is satisfied, because we assigned to each z_i the value of node i, which equals f_i applied to the values of the two nodes feeding into i.[3] Hence, $\mathcal{L} \in 3CSP$.

- If $C \in CircuitSAT$ then for all x we have $C(x) = 0$. Thus, for every assignment $x_1, ..., x_n, z_1, ..., z_s$ to $\mathcal{L}$, either all of the z's correctly represent the node values in $C(x)$, in which case $z_s = 0$ and the output constraint is unsatisfied; or there are $z_i, child(i)_1, child(i)_2$ such that $z_i \neq f_i(child(i)_1, child_i(2))$,[4] in which case the corresponding consistency constraint is unsatisfied. Hence, $\mathcal{L} \notin 3CSP$.

Thus, $C \in CircuitSAT$ if and only if $\mathcal{L} \in 3CSP$. ■

[3] If you are unconvinced, you can prove this by induction on the nodes, using topological order.

[4] Otherwise, if all $z_i = f_i(child(i)_1, child(i)_2)$, then all the z_i's represent the gate values in $C(x)$, and in particular $z_s = C(x) = 0$.