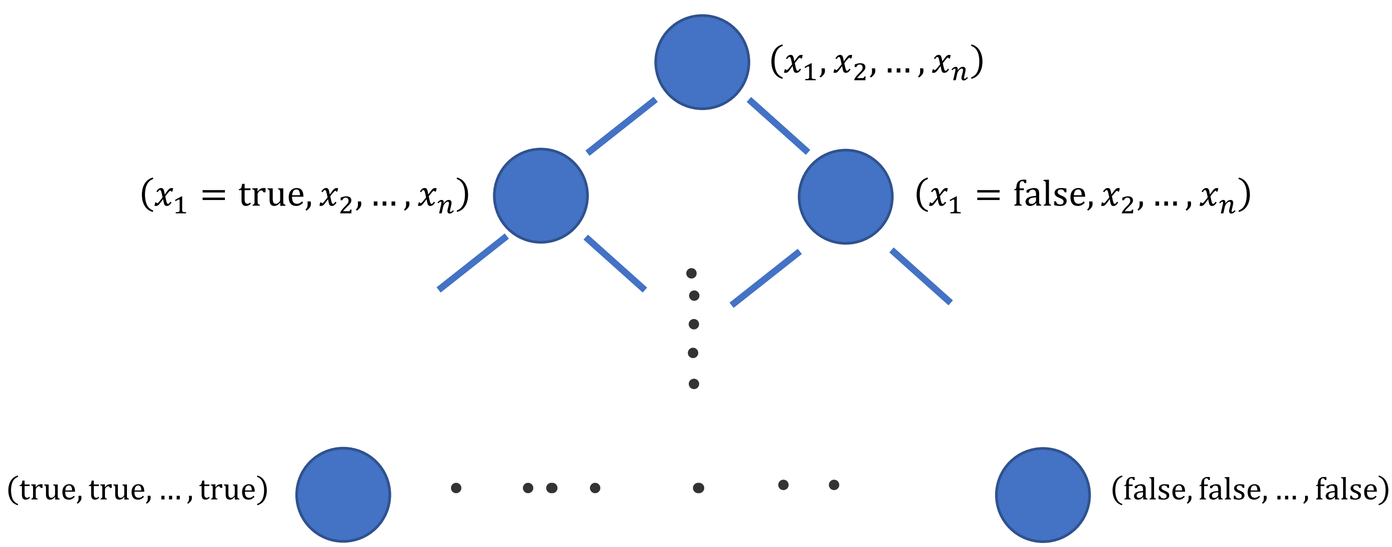
Collaborator: yc2454 (Yalu Cai)

To prove this, we need to show an NP-hard problem is P. We choose SAT. If a unary language is NP-complete, then there exists a polynomial time reduction from SAT to this unary language, and that a SAT problem is satisfiable if and only if the answer to the corresponding unary language problem is a certain answer. Polynomial time reduction means if the SAT problem is n in length, then the reduction took $p(n)$ time, where $p(n)$ means "polynomial in n". This also means the unary reduction of the SAT formula is at most $p(n)$ in length.

We use downward self-reduction to help us solve an SAT problem in polynomial time. A formula represented by the possible assignments of its free variables $(x_1, x_2, \dots, x_n)$ is satisfiable if and only if one of $(x_1 = \text{false}, x_2, \dots, x_n)$ or $(x_1 = \text{true}, x_2, \dots, x_n)$ is satisfiable. Each of those two can then continue the logic to create a recursive tree of depth n and branching factor 2. If we define $U(\phi)$ to be the unary reduction of formula ϕ, we make the observation that $U(\phi) = U(\psi)$ if and only if both ϕ and ψ are satisfiable or both unsatisfiable.



Grow the tree from the root in a depth-first manner and record the unary conversion of each node. Whenever we encounter a unary encoding more than m times with m being the number of variables in the original formula, then we prune any subtrees we encounter later whose root's unary encoding is that. This is because if we find a leaf that evaluates to true, it's along the left-most path that leads to a true leaf (because of depth first). Since we see each unary encoding that indicates an unsatisfiable formula at most m times and we see at most m unary encodings that come from satisfiable formulas, this is a polynomial time algorithm to solve SAT.