Collaborator: yc2454 (Yalu Cai)

To prove this, we need to show an NP-hard problem is P. We choose . If a unary language is NP-complete, then there exists a polynomial time reduction from to this unary language, and that a 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 problem is in length, then the reduction took time, where means “polynomial in n”. This also means the unary reduction of the formula is at most in length.

We use downward self-reduction to help us solve an problem in polynomial time. A formula represented by the possible assignments of its free variables is satisfiable if and only if one of or is satisfiable. Each of those two can then continue the logic to create a recursive tree of depth and branching factor . If we define to be the unary reduction of formula , we make the observation that if and only if both and 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 times with 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 times and we see at most unary encodings that come from satisfiable formulas, this is a polynomial time algorithm to solve SAT.