Homework 16

Joe Baker, Brett Schreiber, Brian Knotten

February 18, 2018

25

Show that if a sparse language is NP-complete, then P = NP.

Proof:
Let $L \in$ NP be a sparse, complete language.

Since L is complete in NP, then there exists a reduction from 3-SAT to L in polynomial time. Let $R(x)$ be the part of that reduction which transforms the 3-CNF boolean formula from 3-SAT to an input to L.

Since L is sparse but 3-SAT is not, then there will be some $x_1, x_2 \in$ 3-SAT where $R(x_1) = R(x_2)$. Let $l = |L| + 1$. Then given any boolean formulas $f_1, f_2, ..., f_l$, consider $R(f_1), R(f_2), ..., R(f_l)$. There are two cases:

- If $\forall i, j$ then $R(f_i) \neq R(f_j)$, then $\exists k \in [1, l]$ such that $R(f_k)$ is unsatisfiable. This is true because under the reduction f is satisfiable iff $R(f)$ is in L, and by the pigeonhole principle, only $l-1$ members of $R(f_1), R(f_2), ..., R(f_l)$ can be in L, so one must be unsatisfiable.
- If $\exists i, j$ such that $R(f_i) = R(f_j)$, then f_i and f_j are either both satisfiable or both unsatisfiable.

We will now construct an algorithm for a TM M which can solve 3-SAT in polynomial time. The algorithm will construct a tree T of different variable assignments. Each level i of T will represent boolean formulas where variable i's assignment is considered. Each node in level $i-1$ has 2 branches, one where variable i is true and one where it is false. At any level, if we have a boolean formula that is satisfiable with the variable assignments at that node then the original formula is satisfiable. Also, if no boolean formulas at a level are satisfiable, then the original formula is unsatisfiable.

With n variables as input, T will have a maximum height of n and 2^n leaves. Constructing and searching this tree as it stands would take exponential time so M must prune branches at each level with more than l nodes. Let M also use the following steps to prune T as it is constructed:

1. Construct l boolean formulas $g_2 = f_1 \vee f_2, g_3 = f_1 \vee f_3, ..., g_{l+1} = f_1 \vee f_{l+1}$
2. Run R against $g_1, ..., g_l$. Since there are l different results of the transformation, we have one of our two previous properties. Consider the following cases:
   (a) $\forall i, j$ then $g_i \neq g_j$. In this case, f_1 must be unsatisfiable. If it were satisfiable, then all g must be satisfiable which we know cannot happen. Thus we prune all the f_1 node since its unsatisfiability will not help us to determine if the overall formula is unsatisfiable.
   (b) $\exists i, j$ such that $g_i = g_j$. Then we can prune either f_i or f_j with the same result. This is because of the two following cases:
      i. If f_1 is satisfiable, then the original formula is satisfiable so we don't need both f_i and f_j to determine that.
      ii. If f_1 is unsatisfiable, then f_i is satisfiable iff f_j is, so either node can be removed without changing whether our original formula is satisfiable.

Following the previous rules for pruning T, will result in a tree with a maximum depth of n and a maximum width of $n * l$ at each level. Thus the tree wil have a maximum of $n^2 * l$ nodes which is polynomial to construct and search. Finally ,M decides 3-SAT in polynomial time.

Since 3-SAT is decidable in polynomial time, 3-SAT $\in$ P. Since 3-SAT is also NP-complete, all languages in NP are in P. So we have the final results that L being NP-complete and sparse implies that P $=$ NP.

26

a

Contruct a TM N with the following behavior:
On input x:

1. Run M on x.
2. If $M(x) = 1$, accept.
3. If $M(x) = 0$, reject.
4. If $M(x) =?$, repeat step 1.

$L(N) \in$ ZPP because it uses a probablistic TM M, but has zero-sided error. If M outputs 1 or 0, then N finishes computation. But since M has a chance of outputting ?, the runtime of N is probablistic, because N only halts when M does not output ?. $P(N \text{ loops once}) = \frac{1}{2}$, $P(N \text{ loops twice}) = \frac{1}{4}$, $P(N \text{ loops } n \text{ times}) = \frac{1}{2^n}$.

b