# Homework 16

Austin Frownfelter Matthew Bialecki

February 19, 2018

## 1 Problem 25

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

Let S be an NP-complete sparse language. Since S is NP-complete, there exists a polytime reduction $f$ from LSAT to S.

Define LSAT to be a variant of SAT where $(\phi, x) \in$ LSAT iff $\phi \in$ SAT $\wedge$ $x \in S$.

Since f is polytime, $|f(x)| \leq |x|^k$ for some k.

The input of LSAT $(\phi, x)$ is converted by $f(x)$ to be the input of S. Since f is polytime, the output of f $\leq p(n)$ where $n = |x|$ and $p(n)$ is a polynomial upper bound. Assign q(x) to the the size of the intersection of the set S and the set of strings $\{0,1\}^{\leq p(n)}$. q(x) is polynomial by sparseness of f.

Imagine an algorithm for LSAT as a tree. The root's children are decided by a variable assignment of the remaining $\phi$. If the number of children (elements in the next row of the tree) is greater than q(n), then prune the branches until the row is q(n) wide using the following policy:

if any branches yield the same string, keep only 1 of them

if there are more than q(x) unique strings, choose q(x) to keep

Repeat this process until all variables have been assigned. Accept if there exists a child in the final row of the tree that satisfies the formula.

Since f is polytime, deciding the children takes polytime. Since there are n levels (n variables), and the width of the tree (number of paths) is at most q(n), the runtime is $nq(n)$ which is polynomial.

Therefore, if there is an NP-complete sparse language, NP-complete languages can be reduced to it to run them in polynomial time, implying P=NP.

(Note: the following websites were used to understand this problem

1. https://math.stackexchange.com/questions/235162/if-an-unary-language-exists-in-npc-then-p-np

2. http://www.cs.umd.edu/ jkatz/complexity/f05/lecture6.pdf)

## 2 Problem 26

Show that ZPP = RP $\cap$ coRP

ZPP = RP $\cap$ CoRP

ZPP $\supseteq$ RP $\cap$ CoRP
Let language L be in RP and CoRP:
There exists a TM A that accepts $x$ with probability $\geq 0.6$ if $x \in L$
There exists a TM B that rejects $x$ with probability $\geq 0.6$ if $x \notin L$

TM C on input x:

for k iterations:

Run A on x. If accept, accept

Run B on x. If reject, reject

reject

The probability C accepts if $x \in L = 1 - P[A(x) \text{ is wrong}]^k$, rejects if $x \notin L$ $= 1 - P[B(x) \text{ is wrong}]^k$. As $k \to \infty$, the probability C is wrong exponentially shrinks to 0. Thus, the probability of C being correct is 1 if $k = \infty$.

The expected runtime for C is polynomial. Therefore, ZPP contains RP and CoRP.

ZPP $\subseteq$ RP $\cap$ CoRP
ZPP $\subseteq$ RP.
Run C on input x for at least double its expected runtime. If it hasn't halted in this time, reject.

By Markov's inequality, the probability C yields a result in this time is at least $\frac{1}{2}$. If it hasn't decided by then, reject, since the probability it is wrong when $x \in L$ is less than $\frac{1}{2}$, matching the definition of RP.

ZPP $\subseteq$ CoRP.
Run C on input x for at least double its expected runtime. If it hasn't halted in this time, accept.

By Markov's inequality, the probability C yields a result in this time is at least $\frac{1}{2}$. If it hasn't decided by then, accept, since the probability it is wrong when $x \notin L$ is less than $\frac{1}{2}$, matching the definition of CoRP.

Therefore, ZPP $\subseteq$ RP $\cap$ CoRP.