Complexity Theory
Lecture 9

Anuj Dawar

University of Cambridge Computer Laboratory
Easter Term 2010

http://www.cl.cam.ac.uk/teaching/0910/Complexity/

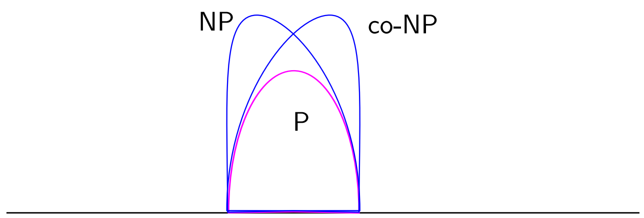
co-NP

As co-NP is the collection of complements of languages in NP, and P is closed under complementation, co-NP can also be characterised as the collection of languages of the form:

$$L = \{x \mid \forall y\, |y| < p(|x|) \rightarrow R'(x,y)\}$$

NP – the collection of languages with succinct certificates of membership.

co-NP – the collection of languages with succinct certificates of disqualification.



Any of the situations is consistent with our present state of knowledge:

- $\mathsf{P} = \mathsf{NP} = \text{co-}\mathsf{NP}$
- $\mathsf{P} = \mathsf{NP} \cap \text{co-}\mathsf{NP} \neq \mathsf{NP} \neq \text{co-}\mathsf{NP}$
- $\mathsf{P} \neq \mathsf{NP} \cap \text{co-}\mathsf{NP} = \mathsf{NP} = \text{co-}\mathsf{NP}$
- $\mathsf{P} \neq \mathsf{NP} \cap \text{co-}\mathsf{NP} \neq \mathsf{NP} \neq \text{co-}\mathsf{NP}$

co-NP-complete

VAL – the collection of Boolean expressions that are *valid* is *co-NP-complete*.

Any language L that is the complement of an NP-complete language is *co-NP-complete*.

Any reduction of a language L_1 to L_2 is also a reduction of $\bar{L_1}$–the complement of L_1–to $\bar{L_2}$–the complement of L_2.

There is an easy reduction from the complement of SAT to VAL, namely the map that takes an expression to its negation.

$$\mathsf{VAL} \in \mathsf{P} \Rightarrow \mathsf{P} = \mathsf{NP} = \text{co-}\mathsf{NP}$$

$$\mathsf{VAL} \in \mathsf{NP} \Rightarrow \mathsf{NP} = \text{co-}\mathsf{NP}$$

Prime Numbers

Consider the decision problem PRIME:

Given a number x, is it prime?

This problem is in co-NP.

$$\forall y(y < x \rightarrow (y = 1 \vee \neg(\mathrm{div}(y, x))))$$

Note again, the algorithm that checks for all numbers up to $\sqrt{n}$ whether any of them divides n, is not polynomial, as $\sqrt{n}$ is not polynomial in the size of the input string, which is $\log n$.

Primality

Another way of putting this is that Composite is in NP.

Pratt (1976) showed that PRIME is in NP, by exhibiting succinct certificates of primality based on:

A number $p > 2$ is *prime* if, and only if, there is a number r, $1 < r < p$, such that $r^{p-1} = 1 \bmod p$ and $r^{\frac{p-1}{q}} \neq 1 \bmod p$ for all *prime divisors* q of $p - 1$.

Primality

In 2002, Agrawal, Kayal and Saxena showed that PRIME is in P.

If a is co-prime to p,

$$(x - a)^p \equiv (x^p - a) \pmod{p}$$

if, and only if, p is a prime.

Checking this equivalence would take to long. Instead, the equivalence is checked *modulo* a polynomial $x^r - 1$, for "suitable" r.

The existence of suitable small r relies on deep results in number theory.

Factors

Consider the language Factor

$$\{(x, k) \mid x \text{ has a factor } y \text{ with } 1 < y < k\}$$

Factor $\in$ NP $\cap$ co-NP

Certificate of membership—a factor of x less than k.

Certificate of disqualification—the prime factorisation of x.

Optimisation

The Travelling Salesman Problem was originally conceived of as an optimisation problem

to find a minimum cost tour.

We forced it into the mould of a decision problem – TSP – in order to fit it into our theory of NP-completeness.

Similar arguments can be made about the problems CLIQUE and IND.

This is still reasonable, as we are establishing the *difficulty* of the problems.

A polynomial time solution to the optimisation version would give a polynomial time solution to the decision problem.

Also, a polynomial time solution to the decision problem would allow a polynomial time algorithm for *finding the optimal value*, using binary search, if necessary.

Function Problems

Still, there is something interesting to be said for *function problems* arising from NP problems.

Suppose

$$L = \{x \mid \exists y R(x, y)\}$$

where R is a polynomially-balanced, polynomial time decidable relation.

A *witness function* for L is any function f such that:

- if $x \in L$, then $f(x) = y$ for some y such that $R(x, y)$;
- $f(x) =$ "no" otherwise.

The class FNP is the collection of all witness functions for languages in NP.

FNP and FP

A function which, for any given Boolean expression ϕ, gives a satisfying truth assignment if ϕ is satisfiable, and returns "no" otherwise, is a witness function for SAT.

If any witness function for SAT is computable in polynomial time, then $\mathsf{P} = \mathsf{NP}$.

If $\mathsf{P} = \mathsf{NP}$, then for every language in NP, some witness function is computable in polynomial time, by a binary search algorithm.

$\mathsf{P} = \mathsf{NP}$ if, and only if, $\mathsf{FNP} = \mathsf{FP}$

Under a suitable definition of reduction, the witness functions for SAT are FNP-complete.

Factorisation

The *factorisation* function maps a number n to its prime factorisation:

$$2^{k_1} 3^{k_2} \cdots p_m^{k_m}.$$

This function is in FNP.

The corresponding decision problem (for which it is a witness function) is trivial - it is the set of all numbers.

Still, it is not known whether this function can be computed in polynomial time.