Aalto University
School of Science

CS-E4530 Computational Complexity Theory

Lecture 9: The Class coNP, Structure of NP, and Relativisations

Aalto University
School of Science
Department of Computer Science

Spring 2017

Agenda

- The class of $\mathbf{coNP}$
- The relationship of $\mathbf{NP}$ and $\mathbf{coNP}$
- The class $\mathbf{NP} \cap \mathbf{coNP}$
- Internal structure of the class $\mathbf{NP}$
- Isomorphisms of $\mathbf{NP}$-complete languages
- Density of $\mathbf{NP}$-complete languages
- Oracle Turing machines and relativised complexity classes

(C. Papadimitriou: *Computational Complexity*, Chapters 10 & 14)

1. The Class of Complement Problems coNP

- **NP** is the class of problems with succinct certificates.
- $\mathbf{coNP} = \{L \mid \overline{L} \in \mathbf{NP}\}$ is the class of problems with succinct refutations.
  As an example, consider the following problem of VALIDITY:

Definition

VALIDITY:
INSTANCE: A Boolean expression ϕ.
QUESTION: Is ϕ valid?

- VALIDITY is in **coNP**: for an expression ϕ which is not valid, a falsifying truth assignment is a succinct refutation.
- HAMILTON PATH COMPLEMENT and SAT COMPLEMENT are also in **coNP**.
- $\mathbf{P} \subseteq \mathbf{coNP}$

coNP-completeness

Definition

A language L is $\mathbf{coNP}$-*complete* iff $L \in \mathbf{coNP}$ and $L' \leq_{\mathrm{L}} L$ holds for every language $L' \in \mathbf{coNP}$.

Proposition

VALIDITY is $\mathbf{coNP}$*-complete.*

Proof

Let $L \in \mathbf{coNP}$; we'll show that L is reducible to VALIDITY. Since $\overline{L} \in \mathbf{NP}$, there is a reduction R from $\overline{L}$ to SAT such that for every string x, $x \in \overline{L}$ iff $R(x) \in$ SAT. But then for a reduction $R'(x) = \neg R(x)$, $x \in L$ iff $R(x) \notin$ SAT iff $R'(x) = \neg R(x) \in$ VALIDITY.

- HAMILTON PATH COMPLEMENT is also $\mathbf{coNP}$-complete.

2. The Relationship of NP and coNP

Proposition

If $L \subseteq \Sigma^*$ is $\mathbf{NP}$-complete, then its complement $\overline{L} = \Sigma^* - L$ is $\mathbf{coNP}$-complete.

Further observations:

- It is open whether $\mathbf{NP} = \mathbf{coNP}$.
- If $\mathbf{P} = \mathbf{NP}$, then $\mathbf{NP} = \mathbf{coNP}$ (and $\mathbf{P} = \mathbf{coNP}$).
- It is possible that $\mathbf{P} \neq \mathbf{NP}$ but $\mathbf{NP} = \mathbf{coNP}$ (however, it is strongly believed that $\mathbf{NP} \neq \mathbf{coNP}$).
- The problems in $\mathbf{coNP}$ that are $\mathbf{coNP}$-complete are the least likely problems to be in $\mathbf{P}$ and also in $\mathbf{NP}$ (see below).

Can NP and coNP coincide?

Proposition

If a $\mathbf{coNP}$*-complete problem is in* $\mathbf{NP}$*, then* $\mathbf{NP} = \mathbf{coNP}$*.*

Proof

Suppose that L is a $\mathbf{coNP}$-complete problem that is in $\mathbf{NP}$.

($\supseteq$) Consider $L' \in \mathbf{coNP}$. Then there is a reduction R from L' to L. Then $L' \in \mathbf{NP}$, because L' can be decided by a polynomial time NTM which on input x computes first $R(x)$ and then runs the NTM for L.

($\subseteq$) Consider $L' \in \mathbf{NP}$. Then $\overline{L'} \in \mathbf{coNP}$ and there is a reduction R from $\overline{L'}$ to L. Then similarly $\overline{L'} \in \mathbf{NP}$ and hence $L' \in \mathbf{coNP}$.

The primality problem

Definition

PRIMES:
INSTANCE: An integer N in binary representation.
QUESTION: Is N a prime number?

- PRIMES $\in \mathbf{coNP}$ as any divisor acts as a succinct refutation.
- Note that an $O(\sqrt{N})$ algorithm for PRIMES testing all relevant divisor candidates is only pseudopolynomial.
- PRIMES $\in \mathbf{NP}$ (as shown below) and hence PRIMES $\in \mathbf{coNP} \cap \mathbf{NP}$.
- Long-standing open problem resolved in 2002:
  M. Agrawal, N. Kayal, N. Saxena: *PRIMES is in* $\mathbf{P}$.

3. The Class $\mathbf{NP} \cap \mathbf{coNP}$

- Problems in $\mathbf{NP} \cap \mathbf{coNP}$ have both succinct certificates and succinct refutations.
- $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$ as $\mathbf{P} \subseteq \mathbf{NP}$ and $\mathbf{P} \subseteq \mathbf{coNP}$.
- If two problems in $\mathbf{NP}$ are *dual*, i.e. each is *reducible to the complement* of the other, then both are in $\mathbf{NP} \cap \mathbf{coNP}$.

Example

MAX FLOW(D): Does network N have a flow of at least K from s to t?
MIN CUT(D): Given a network, is there a set of edges of capacity of at most B such that deleting these disconnects s from t?

Now by the max flow–min cut theorem, N *has* a flow of value at least K iff it *does not have* a cut of capacity $K - 1$ or less.

PRIMES has succinct certificates

A succinct certificate for primality can be obtained using the following theorem (the *Lucas primality test*).

Theorem

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

Example

103 is a prime as (i) $5^{102} = 1 \bmod 103$ and (ii) $5^{\frac{102}{17}} = 72 \bmod 103$, $5^{\frac{102}{3}} = 56 \bmod 103$, and $5^{\frac{102}{2}} = 102 \bmod 103$.

Corollary (V. Pratt 1975)

PRIMES is in $\mathbf{NP} \cap \mathbf{coNP}$.

- The theorem provides a succinct certificate for the primality of p:

$$C(p) = (r; q_1, C(q_1), \ldots, q_k, C(q_k))$$

  where $C(q_i)$ is a *recursive* primality certificate for each prime divisor q_i of $p - 1$.
- The recursion stops for prime divisors $q_i = 2$ for which $C(q_i) = (1)$.
- For example, C(103) = (5; 2,(1),3,(2; 2,(1)),17,(3; 2,(1)))

Verifying the certificate $C(p)$

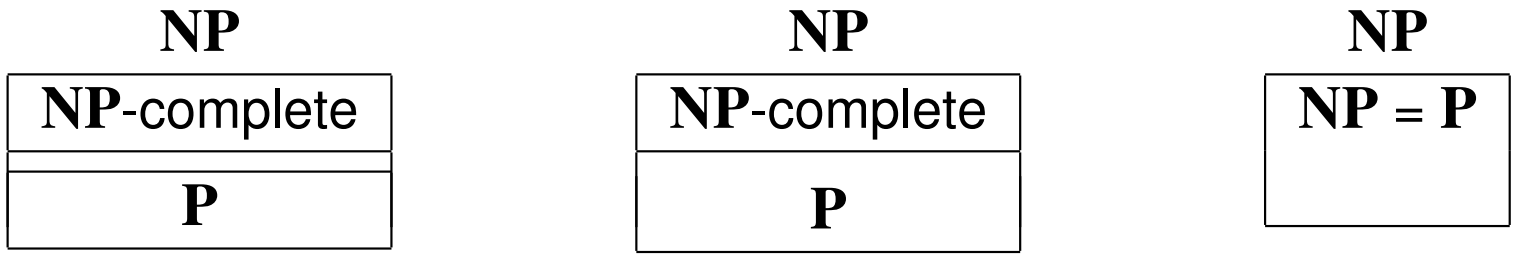
The following observations can be made:

- The certificate $C(p)$ is polynomial in the length of p (in $\log p$) and it can be checked by division and exponentiation.
- Ordinary multiplication and division are doable in polynomial time in the length of the input (in binary representation).
- Exponentiation $r^{p-1} \bmod p$ can be done in polynomial time by repeated squaring $r^1, r^2, r^4, \ldots r^{2^l} \pmod p$ where $l = \lfloor \log_2(p-1) \rfloor$ and then with at most l additional multiplications.

☞ The certificate $C(p)$ can be checked in polynomial time. □

4. Internal Structure of the Class NP

- **NP**-completeness provides a powerful tool for classifying challenging computational problems.
- However, there are problems not known to be in **P** or **NP**-complete, such as GRAPH ISOMORPHISM.
- Are there problems in **NP** that are neither in **P** nor **NP**-complete, i.e., which is the case:



- The middle alternative is not possible.

Theorem (R. Ladner 1975)

If $\mathbf{P} \neq \mathbf{NP}$*, then there is a language in* **NP** *which is neither in* **P** *nor* **NP***-complete.*

5. Isomorphisms of NP-Complete Languages

Definition

Languages K, L are *polynomial-time isomorphic* if there is a function $h : \Sigma^* \mapsto \Sigma^*$ such that

(1) h is a bijection;

(2) for each $x \in \Sigma^*$, $x \in K$ iff $h(x) \in L$; and

(3) h and h^{-1} are polynomial-time computable.

- There are always polynomial-time mappings satisfying condition (2) between any two **NP**-complete languages, viz. the logarithmic space reductions.
- However, a reduction is not necessarily an isomorphism, as it is not necessarily a bijection.

Theorem (L. Berman & J. Hartmanis 1977)

If there is a reduction from K to L and a reduction from L to K, such that the reductions are length-increasing, one-to-one and logarithmic-space invertible, then K and L are polynomial-time isomorphic.

Logspace reductions between NP-complete problems can in all known cases be easily made length-increasing, one-to-one and logspace invertible by the technique of *padding functions* (discussed below), which led Berman and Hartmanis to conjecture the following:

Conjecture If $\mathbf{P} \neq \mathbf{NP}$, then all $\mathbf{NP}$-complete languages are polynomial-time isomorphic.

Definition

A function $pad : (\Sigma^*)^2 \mapsto \Sigma^*$ is a *padding function* for L if

1. pad is computable in logarithmic space;
2. for each $x, y \in \Sigma^*$, $pad(x,y) \in L$ iff $x \in L$;
3. for each $x, y \in \Sigma^*$, $|pad(x,y)| > |x| + |y|$; and
4. there is a logarithmic-space algorithm which given $pad(x,y)$ recovers y.

Example (a padding function for SAT)

Given x (a conjunction of m clauses with n variables $v_1, \ldots, v_n$) and a binary string y, $pad(x,y)$ is all clauses of x together with $m + |y|$ more clauses with $|y| + 1$ more variables where the m new clauses are copies of the clause (v_{n+1}) and where the $m+i$'th new clause is either $(\neg v_{n+i+1})$ or (v_{n+i+1}) depending on whether the i'th symbol in y is 0 or 1.

- If R is a reduction from K to L and pad is a padding function for L, then the mapping $h(x) = pad(R(x), x)$ is a length-increasing one-to-one reduction from K to L. Furthermore, there is a logarithmic-space algorithm to compute the inverse $h^{-1}(y) = x$ from $y = pad(R(x), x)$.

Corollary

The following $\mathbf{NP}$*-complete languages are polynomial-time isomorphic: SAT, HAMILTON PATH, CLIQUE, MAX CUT, TRIPARTITE MATCHING, KNAPSACK, . . .*

6. Density of NP-Complete Languages

Density of a language L:

$$\mathrm{dens}_L(n) = |\{x \in L : |x| \leq n\}|$$

Proposition

If $K, L \subseteq \Sigma^*$ are polynomial-time isomorphic, then dens_K and dens_L are polynomially related ($\mathrm{dens}_L(n) \leq \mathrm{dens}_K(p(n))$).

- A *sparse* language: polynomially bounded density function
- A *dense* language: superpolynomial density function

Density—cont'd

- Any *tally language* L, i.e. a language over a unary alphabet ($L \subseteq \{0\}^*$) is sparse. In fact, in this case $\mathrm{dens}_L(n) \leq n$.
- All known $\mathbf{NP}$-complete language, on the other hand, are dense.
- If the Berman-Hartmanis conjecture were true, this would be a general property of $\mathbf{NP}$-complete languages.
- The following theorems show that the conjecture cannot be refuted based on simple density considerations.

Theorem (S. Fortune 1979)

If some tally language is $\mathbf{NP}$-complete, then $\mathbf{P} = \mathbf{NP}$.

Theorem (S. Mahaney 1982)

If some sparse language is $\mathbf{NP}$-complete, then $\mathbf{P} = \mathbf{NP}$.

7. Oracle Turing Machines and Relativised Complexity Classes

- The idea: study computational complexity in a setting where a part of the computation comes “for free”.
- Can be used for exploring conjectures (like $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$) in “alternative universes”.
- Can suggest limits on the power of certain types of proof techniques.
- Also provides a useful tool for generalising basic complexity classes and building complexity hierarchies. (To be discussed at next lecture.)

- An *oracle Turing machine* $M^?$:
  - New elements:
    query string, query state $q?$, answer states q_{YES}, q_{NO}.
  - From the query state $q?$ the machine moves to q_{YES} or to q_{NO}, depending on whether $y \in A$ holds or not, where y is the content of the query string and A the oracle set.
  - Note that *a query* is performed *in one step*!
  - Computation of $M^?$ with oracle A on input x: $M^A(x)$.
- For any time or space complexity class $\mathcal{C}$ and oracle A there is a corresponding *relativised* complexity class $\mathcal{C}^A$.

Theorem (T. Baker, J. Gill, R. Solovay 1975)

There is an oracle A for which $\mathbf{P}^A = \mathbf{NP}^A$.

Proof. Let A be $\mathbf{PSPACE}$-complete. Then

$$\mathbf{PSPACE} \subseteq \mathbf{P}^A \subseteq \mathbf{NP}^A \subseteq \mathbf{NPSPACE} \subseteq \mathbf{PSPACE}.$$

Oracles and Relativisations —cont'd

Theorem (T. Baker, J. Gill, R. Solovay 1975)

There is an oracle B for which $\mathbf{P}^B \neq \mathbf{NP}^B$.

Proof idea. B is constructed by "diagonalisation" over all polynomial-time Turing machines. In essence, information is "hidden" in the oracle language B in such a manner that a nondeterministic machine can find it by guessing, but every deterministic polynomial-time machine is guaranteed to miss at least one piece of it.

Lessons to be learned w.r.t. $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$:

- Both $\mathbf{P} = \mathbf{NP}$ and $\mathbf{P} \neq \mathbf{NP}$ are plausible conjectures, as they are both true in some "relativised universe".
- Ordinary simulation-based proof techniques are oblivious to the presence or absence of oracles. Hence they are not sufficient for solving $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.

Learning Objectives

- The definition of $\mathbf{coNP}$ and examples of languages from this class, e.g., VALIDITY.
- The characterisation of $\mathbf{coNP}$ based on refutations.
- The class $\mathbf{NP} \cap \mathbf{coNP}$ and examples, e.g. PRIMALITY.
- The concepts of polynomial-time isomorphism and density of a language.
- The concepts of oracle Turing machine and relativised complexity classes.