Complexity Theory

Lecture 8: coNP

Tom Gur

http://www.cl.cam.ac.uk/teaching/2324/Complexity

What's next

What's next

1) coNP

What's next

1) coNP
2) Cryptography

What's next

1) coNP
2) Cryptography
3) Space Complexity

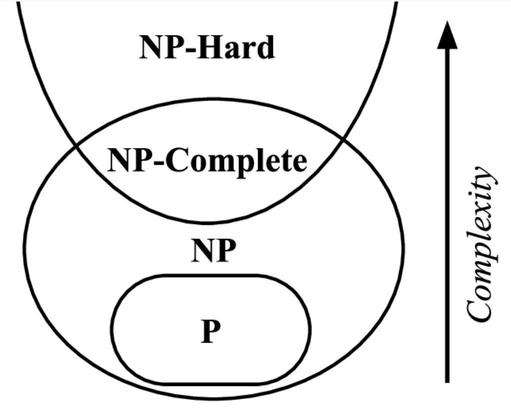
What's next

1) coNP
2) Cryptography
3) Space Complexity
4) Space and Time Hierarchy

What's next

1) coNP
2) Cryptography
3) Space Complexity
4) Space and Time Hierarchy
5) Quantum Complexity

The story so far, in a picture



Bypassing NP-Completeness

Confronted by an NP*-complete problem, what can we do?*

Bypassing NP-Completeness

Confronted by an NP-complete problem, what can we do?

- It's a single instance, does asymptotic complexity matter? (Chess?)

Bypassing NP-Completeness

Confronted by an NP-complete problem, what can we do?

- It's a single instance, does asymptotic complexity matter? (Chess?)
- What about abusing the representation?

Bypassing NP-Completeness

Confronted by an NP-*complete problem, what can we do?*

- It's a single instance, does asymptotic complexity matter? (Chess?)
- What about abusing the representation?
- Are the inputs structured?

Bypassing NP-Completeness

Confronted by an NP-complete problem, what can we do?

- It's a single instance, does asymptotic complexity matter? (Chess?)
- What about abusing the representation?
- Are the inputs structured?
- Can we use randomness? Quantum?

Bypassing NP-Completeness

Confronted by an NP-complete problem, what can we do?

- It's a single instance, does asymptotic complexity matter? (Chess?)
- What about abusing the representation?
- Are the inputs structured?
- Can we use randomness? Quantum?
- Is it enough to only deal with average-case instances?

Bypassing NP-Completeness

Confronted by an NP-complete problem, what can we do?

- It's a single instance, does asymptotic complexity matter? (Chess?)
- What about abusing the representation?
- Are the inputs structured?
- Can we use randomness? Quantum?
- Is it enough to only deal with average-case instances?
- Will an approximate solution suffice? (TODAY: Ordered TSP)

Bypassing NP-Completeness

Confronted by an NP-complete problem, what can we do?

- It's a single instance, does asymptotic complexity matter? (Chess?)
- What about abusing the representation?
- Are the inputs structured?
- Can we use randomness? Quantum?
- Is it enough to only deal with average-case instances?
- Will an approximate solution suffice? (TODAY: Ordered TSP)
- Can we delegate the computation?

Bypassing NP-Completeness

Confronted by an NP-complete problem, what can we do?

- It's a single instance, does asymptotic complexity matter? (Chess?)
- What about abusing the representation?
- Are the inputs structured?
- Can we use randomness? Quantum?
- Is it enough to only deal with average-case instances?
- Will an approximate solution suffice? (TODAY: Ordered TSP)
- Can we delegate the computation?
- Are there useful heuristics that can constrain a search? SAT-solvers?

Beyond NP!

Unsatisfiability

We define UNSAT to be the set of all Boolean functions for which there are no satisfying assignments. (algorithm?)

Unsatisfiability

We define UNSAT to be the set of all Boolean functions for which there are no satisfying assignments. (algorithm?)

By an exhaustive search algorithm similar to the one for SAT, UNSAT is in $\mathsf{TIME}(n^2 2^n)$.

Unsatisfiability

We define UNSAT to be the set of all Boolean functions for which there are no satisfying assignments. (algorithm?)

By an exhaustive search algorithm similar to the one for SAT, UNSAT is in $\mathsf{TIME}(n^2 2^n)$.

Is $\mathsf{UNSAT} \in \mathsf{NP}$?

Unsatisfiability

We define UNSAT to be the set of all Boolean functions for which there are no satisfying assignments. (algorithm?)

By an exhaustive search algorithm similar to the one for SAT, UNSAT is in $\mathsf{TIME}(n^2 2^n)$.

Is UNSAT $\in$ NP?

Note that UNSAT is the complement of SAT!

Complementation

Question 1: If a language $L \in \mathsf{P}$, then is $\overline{L} \in \mathsf{P}$ as well?

Complementation

Question 1: If a language $L \in \mathsf{P}$, then is $\overline{L} \in \mathsf{P}$ as well?

Yes. Run the TM and switch ACCEPT with REJECT.

Complementation

Question 1: If a language $L \in \mathsf{P}$, then is $\overline{L} \in \mathsf{P}$ as well?

Yes. Run the TM and switch ACCEPT with REJECT.

Question 2: If a language $L \in \mathsf{NP}$, then is $\overline{L} \in \mathsf{NP}$ as well?

Complementation

Question 1: If a language $L \in \mathrm{P}$, then is $\overline{L} \in \mathrm{P}$ as well?

Yes. Run the TM and switch ACCEPT with REJECT.

Question 2: If a language $L \in \mathrm{NP}$, then is $\overline{L} \in \mathrm{NP}$ as well?

Not necessarily: the quantifiers change – "there exists" becomes "for all".

Complementation

Question 1: If a language $L \in \mathrm{P}$, then is $\overline{L} \in \mathrm{P}$ as well?

Yes. Run the TM and switch ACCEPT with REJECT.

Question 2: If a language $L \in \mathrm{NP}$, then is $\overline{L} \in \mathrm{NP}$ as well?

Not necessarily: the quantifiers change – "there exists" becomes "for all".

This leads to the following natural definition:

co-NP – the languages whose complements are in NP.

Succinct Certificates

The complexity class NP can be characterised as the collection of languages of the form:

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

Where R is a relation on strings satisfying two key conditions

Succinct Certificates

The complexity class NP can be characterised as the collection of languages of the form:

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

Where R is a relation on strings satisfying two key conditions

1. R is decidable in polynomial time.

Succinct Certificates

The complexity class NP can be characterised as the collection of languages of the form:

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

Where R is a relation on strings satisfying two key conditions

1. R is decidable in polynomial time.
2. R is *polynomially balanced*. That is, there is a polynomial p such that if $R(x, y)$ and the length of x is n, then the length of y is no more than $p(n)$.

co-NP

As co-NP is the collection of complements of languages in NP, hence can also be characterised as the collection of languages of the form:

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

Note that $\neg R$ is poly-time decidable
(as P is closed under complementation, and R is as before).

co-NP

As co-NP is the collection of complements of languages in NP, hence can also be characterised as the collection of languages of the form:

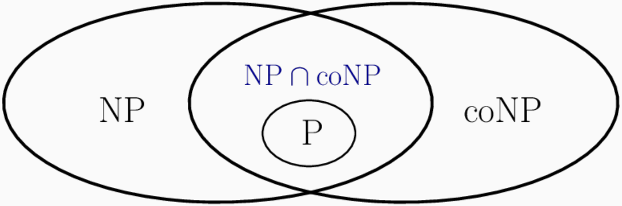
$$L = \{x \mid \forall y \neg R(x, y)\}$$

Note that $\neg R$ is poly-time decidable
(as P is closed under complementation, and R is as before).

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

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

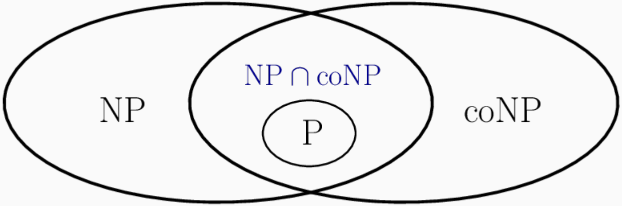
Extending our picture



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

- P = NP = co-NP

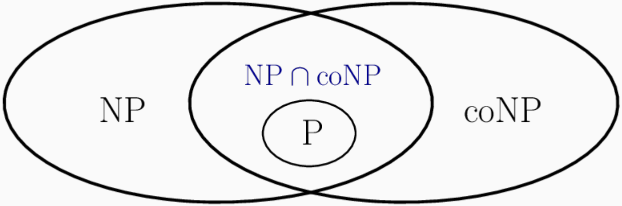
Extending our picture



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

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

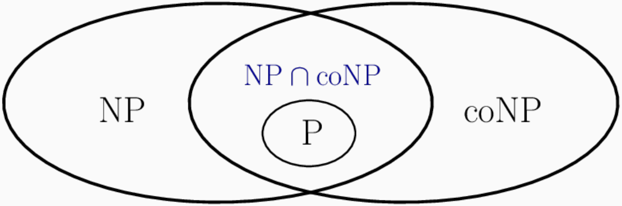
Extending our picture



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

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

Extending our picture



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

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

Interlude: On “belief” in mathematics and CS

coNP-completeness

UNSAT – the collection of Boolean formulas that are not satisfiable is *co-NP-complete*.

coNP-completeness

UNSAT – the collection of Boolean formulas that are not satisfiable is *co-NP-complete*.

Any language L that is the complement of an NP-complete language is *co-NP-complete*. (why?)

coNP-completeness

UNSAT – the collection of Boolean formulas that are not satisfiable is *co-NP-complete*.

Any language L that is the complement of an NP-complete language is *co-NP-complete*. (why?)

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

Prime Numbers

Consider the decision problem PRIME:

Given a number x, is it prime?

Prime Numbers

Consider the decision problem PRIME:

Given a number x, is it prime?

Prime Numbers

Consider the decision problem PRIME:

Given a number x, is it prime?

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 $\log n$.

Prime Numbers

Consider the decision problem PRIME:

Given a number x, is it prime?

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 $\log n$.

Prime Numbers

Consider the decision problem PRIME:

Given a number x, is it prime?

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 $\log n$.

This problem is in co-NP. (why?)

Primality

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

Primality

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

Is PRIME is in NP?

Primality

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

Is PRIME is in NP?

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

Primality

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

Is PRIME 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

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

Is PRIME 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

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

Is PRIME 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 $p - 1$.*

$\mathsf{NP} \cap \text{co-}\mathsf{NP} \setminus \mathsf{P}$ is often where quantum might have a great potential!

Primality

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

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.

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.

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

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

Factors

Consider the language Factor

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

What is the relation to the search version?

Factors

Consider the language Factor

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

What is the relation to the search version?

In what complexity classes can we place Factor?

Factors

Consider the language Factor

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

What is the relation to the search version?

In what complexity classes can we place Factor?

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

Factors

Consider the language Factor

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

What is the relation to the search version?

In what complexity classes can we place Factor?

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

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

Factors

Consider the language Factor

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

What is the relation to the search version?

In what complexity classes can we place Factor?

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

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

Certificate of disqualification—the prime factorisation of x.

Graph Isomorphism

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, is there a *bijection*

$$\iota : V_1 \to V_2$$

such that for every $u, v \in V_1$,

$$(u, v) \in E_1 \quad \text{if, and only if,} \quad (\iota(u), \iota(v)) \in E_2.$$

Graph Isomorphism

Graph Isomorphism is

Graph Isomorphism

Graph Isomorphism is

- in NP

Graph Isomorphism

Graph Isomorphism is

- in NP
- not known to be in P

Graph Isomorphism

Graph Isomorphism is

- in NP
- not known to be in P
- not known to be in co-NP

Graph Isomorphism

Graph Isomorphism is

- in NP
- not known to be in P
- not known to be in co-NP
- not known (or *expected*) to be NP-complete

Graph Isomorphism

Graph Isomorphism is

- in NP
- not known to be in P
- not known to be in co-NP
- not known (or *expected*) to be NP-complete
- shown to be in *quasi-polynomial time*, i.e. in

$$\mathrm{TIME}(n^{(\log n)^k})$$

  for a constant k.

Bonus: Randomness and BPP