DD2445 COMPLEXITY THEORY: LECTURE 15

Last lecture

Ended in the middle of proof of $coNP \subseteq IP$
(to illustrate ideas in result $IP = PSPACE$)

VERIFIER — Probabilistic poly time (in $|x|$). Private random string r

PROVER — Computationally unbounded

$$\xrightarrow{a_1 = f(x, r)}$$

$$\xleftarrow{a_2 = g(x, a_1)}$$

$$\xrightarrow{a_3 = f(x, r, a_1, a_2)}$$

$$\xleftarrow{a_4 = g(x, a_1, a_2, a_3)}$$

$$\vdots$$

TRANSCRIPT

Verifier announces decision
$out_f \langle f, g \rangle (x) \in \{0, 1\}$

Language L in $\boxed{IP}$ if $\exists$ protocol with polynomial (in $|x|$) # rounds with

COMPLETENESS

$x \in L \Rightarrow \exists$ prover P $\Pr[out_V \langle V, P \rangle (x) = 1] \geq 2/3$

SOUNDNESS

$x \notin L \Rightarrow \forall P'$ $\Pr[out_V \langle V, P' \rangle (x) = 1] \leq 1/3$

THEOREM 9 $coNP \subseteq IP$

Construct protocol for more general problem

$$\boxed{\#SAT_D = \{ \langle \varphi, K \rangle \mid \varphi \text{ 3-CNF with exactly } K \text{ satisfying assignments} \}}$$

$K = 0$ gives $\overline{3\text{-SAT}}$ as special case

Write clause C_j as polynomial p_j

$$x_i \vee \overline{x}_j \vee x_k \rightsquigarrow 1 - (1-x_i)\, x_j\, (1-x_k)$$

Write formula $\varphi = \bigwedge_{j=1}^{m} C_j$ as polynomial

$$P_\varphi = \prod_{j=1}^{m} p_j \qquad (*)$$

Degree $\leq 3m$

Efficient representation in size $O(m)$ (arithmetic circuit)

Want to check # satisfying assignments

$$K = \sum_{b_1 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\varphi(b_1, \ldots, b_n) \qquad (**)$$

Do calculations mod prime $p > 2^n \geq K$

Observation If for $g(x_1, \ldots, x_n)$ we plug in $x_i = b_i$ for $i = 2, \ldots, n$, then get univariate polynomial. True also for

$$h(x_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(x_1, b_2, \ldots, b_n) \qquad (\dagger)$$

(for $g = P_\varphi$ or other polynomial)

We have that

$$K = \sum_{b_1 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(b_1, \ldots, b_n) \qquad (\dagger)$$

iff $h(0) + h(1) = K$ (obviously)

Idea of protocol

- Ask prover for prime $p \in (2^n, 2^{2n}]$
- Check that p prime
- Ask prover for $h(x_1)$
- Check $h(0) + h(1) = K$
- Check that prover was honest when giving $h(x_1)$.

SUMCHECK (g, K, n)

V: If $n = 1$, accept if $g(0) + g(1) = K$, reject otherwise

If $n \geq 2$, ask prover for $h(x_1)$ in $(\dagger)$

P: Sends $s(x_1)$

V: Check if $s(0) + s(1) = K$; reject otherwise

Pick $a \in_R [0, p-1]$

$K' := s(a)$

$g' := g(a, x_2, \ldots, x_n)$

NOTE that g' also has efficient representation

Run SUMCHECK $(g', K', n-1)$

Recursive call checks that $s(x_1) = h(x_1)$ by verifying

$$s(a) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(a, b_1, \ldots, b_n)$$

LEMMA 10

If g degree-d polynomial and p prime, then SUMCHECK (g, K, n) has
- completeness 1
- soundness error $\leq dn/p$

φ m clauses $\Rightarrow \deg(P_\varphi) \leq 3m$

φ 3-CNF over n variables $\Rightarrow m \leq 27n^3$

Pick $p > 2^n$. Get soundness error

$$\leq \frac{dn}{p} < \frac{81n^4}{2^n} \rightarrow 0$$

So $\mathrm{coNP} \subseteq \mathrm{IP}$ follows from Lemma 10

~~Proof of Lemma 10~~

Completeness: Obvious. Prover answers honestly, and all verifier checks pan out.

Soundness: By induction over n.

Base case ($n = 1$): Want to detect if

$$\sum_{b \in \{0,1\}} g(b) \neq K$$

Compute $g(0) + g(1)$

0% probability of being fooled.

Inductive step: Want to detect if

$$\sum_{b_1 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(b_1, \ldots, b_n) \neq K$$

Induction hypothesis says that SUMCHECK$(g', K', n-1)$ has soundness error $\leq \frac{d}{p}(n-1)$

Two cases:

(a) Prover honestly replies with $h(x_1)$ as in $(\ddagger)$

But then $h(0)+h(1) \neq K$ and verifier has 0% probability of being fooled

(b) Prover replies with $s(x_1) \neq h(x_1)$

$\deg(s(x_1) - h(x_1)) \leq d$

$\Rightarrow$ $s(x_1) - h(x_1)$ has $\leq d$ roots

$\Rightarrow$ at most d values for a such that $s(a) = h(a)$

(i) If prover is lucky and verifier picks a s.t. $s(a) = h(a)$, then verifier fooled

(ii) Otherwise, get sumcheck instance for polynomial g' over $n-1$ variables with wrong value K

$$\Pr[\text{verifier } V \text{ fooled}] = \Pr[V \text{ fooled in case (i)}] + \Pr[V \text{ fooled in case (ii)}]$$

$$\leq \frac{d}{p} + \frac{d}{p}(n-1) = \frac{dn}{p}$$

The lemma follows by the induction principle $\blacksquare$

We proved coNP $\subseteq$ IP
Actually most of what is needed for PSPACE $\subseteq$ IP except for some extra twists.

What was the key idea? ARITHMETIZATION
CNF formula $\varphi \rightsquigarrow$ polynomial P_φ
Evaluate polynomial in much larger field $\Rightarrow$ makes it practically impossible for prover to cheat.

Can also define $\boxed{\text{MULTIPROVER INTERACTIVE PROTOCOLS (MIP).}}$ Provers agree beforehand on shared strategy but <u>cannot</u> communicate <u>during protocol</u>

PROVER 1		VERIFIER		PROVER 2
	$\xleftarrow{a_1 = f_1(x,r)}$		$\xrightarrow{b_1 = f_2(x,r)}$	
	$\xrightarrow{a_2 = g_1(x,a_1)}$		$\xleftarrow{b_2 = g_2(x,b_2)}$	
	$\xleftarrow{a_3 = f_1(x,r,a_1,a_2,b_1,b_2)}$		$\xrightarrow{b_3 = f_2(x,r,b_1,b_2,a_1,a_2)}$	

$\vdots$

Can allow up to polynomially many provers (but verifier needs to have enough time to read all answers)

In fact, just going from 1 to 2 provers gives as much power as polynomially many provers.

Define MIP analogously to IP

Clearly, $IP \subseteq MIP$ [can always ignore one prover]

THM 11 [Babai, Fortnow, Lund '90]

$MIP = NEXP$

Why are 2 provers more useful?

Can use 2nd prover to force nonadaptivity of 1st prover

Suppose prover 1 gets questions

$q_1, q_2, \ldots, q_m$

Prover 1 sees context and can choose answer to q_j depending on $q_1, \ldots, q_{j-1}$

But if verifier randomly picks $i \in_R [m]$ and asks q_i from prover 2, and requires that provers 1 & 2 should give same answer to q_i, then prover 1 can no longer answer adaptively (because prover 2 cannot answer adaptively)

So provers might as well write down and publish big table with answers to all possible questions. [This needs a formal argument, of course.]

Verifier questions = random look-ups in table

PCP[r, q] = set of languages
that can be decided by q random
checks in table of size 2^r
[informal definition]

Can restate Thm 11 as

$$NEXP = PCP[poly, poly] = \bigcup_{c \in \mathbb{N}^+} PCP[n^c, n^c]$$

Can be "scaled down" to

$$NP = PCP[polylog, polylog]$$

And further improved (with LOTS of work)
~~to~~

THM 12 PCP THEOREM [Arora-Safra '92]
[Arora-Lund-Motwani-Sudan-Szegedi '98]

$$NP = PCP[O(\log n), O(1)]$$

Means that for any language $L \in NP$
can write down proofs π of $x \in L$ s.t.

- π has size $poly(|x|)$
- π can be checked by reading constant #bits (independent of size of x)
- if $x \in L$, accept whp
  if $x \notin L$, reject whp

Now if this isn't magic...
Proof is highly nontrivial and would take several lectures even just for an overview

But let us look at a nontrivial example

EXAMPLE 13 GRAPH NONISOMORPHISM $\in PCP[\mathrm{poly}(n), O(1)]$

$$GNI = \{ \langle G_0, G_1 \rangle \mid G_0 \ncong G_1 \}$$

Graphs on n vertices
Represent by adjacency matrix
Binary string of length $n^2 \leftrightarrow$ number in $[0, 2^{n^2}-1]$

Proof π: Binary string of length 2^{n^2}
Let position $p \in [0, 2^{n^2}-1]$ correspond to graph H_p.
Expected format of proof
Bit in position p is
a) 0 if $H_p \cong G_0$
b) 1 if $H_p \cong G_1$
c) don't care otherwise

(Could have replaced n^2 by $\binom{n}{2}$ — don't care)

Verifier test
1. Flip $b \in_R \{0,1\}$
2. Choose random permutation $\sigma : [n] \to [n]$
3. Let $H_p = \sigma(G_b)$
4. Look up bit b' in position p
5. Accept if $b = b'$; reject otherwise

Analysis

Completeness If $G_0 \ncong G_1$, prover constructs table π according to specification.
Verifier's test will always accept

Soundness (sketch)

If $G_0 \cong G_1$, then probability of checking position p is independent of b.

So, mentally, we can

(i) Choose random σ

(ii) Look up b' in position p for $H_p = \pi(G_1)$

(iii) Only now flip $b \in_R \{0, 1\}$

(iv) Accept if $b = b'$ [with probability $= 1/2$]

More formal proof of soundness

Suppose $G_0 \cong G_1$

Consider for $i \in \{0, 1\}$ distributions

$$\mathcal{D}_i = \{ \sigma(G_i) \mid \sigma : [n] \to [n] \text{ uniformly sampled random permutation} \}$$

Then $\mathcal{D}_0$ and $\mathcal{D}_1$ are identical.

Because following two experiments give same distribution

(1) Pick random permutation $\sigma : [n] \to [n]$
and return σ

(2) Fix arbitrary permutation $\sigma^* : [n] \to [n]$
Pick random permutation $\sigma : [n] \to [n]$
Return $\sigma \circ \sigma^*$

So we can let σ^* be permutation such that $\sigma^*(G_0) = G_1$ exact equality

$$\Pr[\text{accept}] = \sum_{\substack{\text{position } p \\ \text{in table}}} \Pr[\text{read pos } p] \cdot \Pr[\text{read bit} = b \mid \text{read pos } p] \qquad (*)$$

$\Pr[\text{read pos } p]$ independent of b by argument above

Hence, what bit $b' = b[p]$ verifier reads is independent of coin flip b. So

$$(*) = \sum_{\text{pos } p} \Pr[\text{read pos } p] \cdot \Pr[b = \text{some fixed bit}]$$

$$= \Pr[b = \text{some fixed bit}] \cdot \sum_{\text{pos } p} \Pr[\text{read } p]$$

$$= \Pr[b = \text{some fixed bit}]$$

$= 1/2$ as claimed

More on to

CRYPTOGRAPHY

Just scratch the surface
DD2448 Foundations of Cryptography
given in spring
[Necessary for a well-rounded (T)CS
education, if you ask me]

"HUMAN INGENUITY CANNOT CONCOCT
A CIPHER WHICH HUMAN INGENUITY
CANNOT RESOLVE"
Edgar Allan Poe 1841

Cat-and-mouse game throughout
the ages.

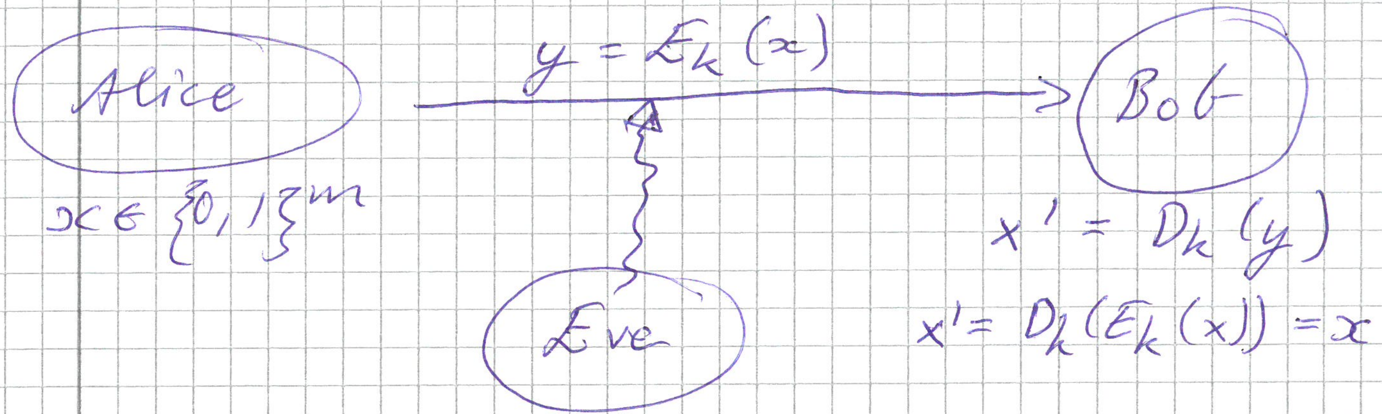
Shannon (late '40s): rigorous definition
of security

1970s: Birth of modern cryptography
Connection to computational complexity theory
Make code breaking a computationally
hard problem (so hardness ⇒ good news!)

Cross-fertilization Many ideas from
crypto have turned out to be extremely
useful in complexity theory

BASIC TASK

key $k \in_R \{0,1\}^n$



<u>DEF 14</u> Encryption scheme is <u>perfectly secret</u> if $\forall x, x' \in \{0,1\}^m$ the distributions $E_{U_n}(x)$ and $E_{U_n}(x')$ are identical

> Recall: U_n = uniformly random n-bit strings

<u>EXAMPLE 15</u> Suppose $n = m$

Let y = bitwise XOR of x and k

($y_i = x_i + k_i \pmod 2$) ONE-TIME PAD

Not hard to prove $E_k(x)$ looks perfectly random to outside observer

<u>CLAIM 16</u> If (E, D) is an encryption scheme with $n < m$, then it is <u>not</u> perfectly secret

Proof: Nice exercise

Solution? Drop perfect secrecy
Require secrecy only w.r.t.
<u>computationally bounded adversaries</u>

[Even NSA is computationally bounded...]

If this is to be possible, need $P \neq NP$
(see Lem 9.2 in Arora-Barak)
But this is <u>not sufficient</u>

Let us <u>very briefly</u> sketch a basic assumption of modern crypto and some consequences of it that allow us to recover a "computationally secure one-time pad"

<u>DEF 17</u> Function $\varepsilon : \mathbb{N} \to [0,1]$ is NEGLIGIBLE if $\varepsilon(n) = n^{-\omega(1)}$
[That is, $\forall c \; \exists N$ s.t. $\forall n > N$
$\varepsilon(n) < n^{-c}$]

<u>DEF 18</u> A poly-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is a ONE-WAY FUNCTION if for every probabilistic poly-time algorithm A there is a negligible function ε s.t.

$$\Pr_{\substack{x \in_R \{0,1\}^n \\ y = f(x)}} \left[\cancel{A(y)}\, A(1^n, y) = x' \text{ s.t. } f(x') = y \right] < \varepsilon(n)$$

Note that input 1^n is needed to guarantee that A has time to output answer

CONJECTURE/ASSUMPTION 19

One-way functions exist

CLAIM 20

If one-way functions exist, then $P \neq NP$

Proof: Good exercise; not hard.

And if one-way functions exist, then computationally secure encryption schemes exist.

Will talk a little bit more about this (and some other aspects of cryptography) next lecture