DD2445 COMPLEXITY THEORY: LECTURE 15

Last lecture

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

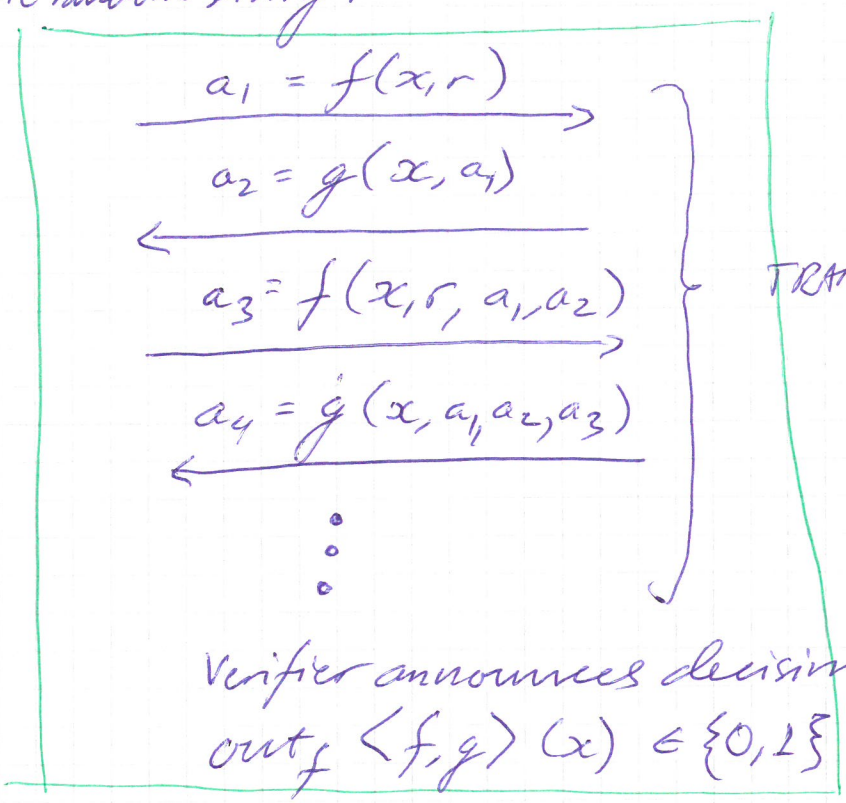
VERIFIER

Probabilistic poly time (in $|x|$)

Private random string r

PROVER

Computationally unbounded



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

COMPLETENESS

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

SOUNDNESS

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

THEOREM 8 [Lund, Fortnow, Karloff, Nisan '90] [Shamir '90]

$IP = PSPACE$

$IP \subseteq PSPACE$ not hard, as handwaved before

For $PSPACE \subseteq IP$, consider PSPACE-complete language TQBF and show $TQBF \in IP$.

We'll prove something weaker than this, but conveys the main ideas

THEOREM 9 $coNP \subseteq IP$

Want protocol for

$$\overline{3\text{-SAT}} = \{\varphi \mid \varphi \text{ unsatisfiable 3-CNF}\}$$

Design more general protocol for proving number of satisfying assignments

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

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

Key trick: ARITHMETIZATION of CNF formula

Fix some suitable finite field $\mathbb{F}$ (e.g., computing modulo some prime p)

1 = truth = multiplicative identity of $\mathbb{F}$

0 = falsity = additive identity

$\neg x_i = \overline{x_i}$ true iff $1 - x_i = 1$

$x_i \wedge x_j$ true iff $x_i \cdot x_j = 1$

$x_i \vee x_j$ true iff $1 - (1-x_i)(1-x_j) = 1$

Rewrite every 3-clause

$$c_j = x_i \vee \overline{x_j} \vee x_k$$

as degree-3 polynomial

$$p_j(x_1, \ldots, x_n) = 1 - (1-x_i)\, x_j\, (1-x_k)$$

Evaluates to 1 over 0/1-assignments iff clause true.

Represent $\varphi = \bigwedge_{j=1}^{m} c_j$ as

$$P_\varphi = \prod_{j=1}^{m} p_j(x_1, \ldots, x_n) \qquad (*)$$

Degree $\leq 3m$

P_φ has efficient representation in size $O(m)$

α satisfying assignment to $\varphi \Rightarrow P_\varphi(\alpha) = 1$

α falsifying $-\,''\,-$ $\Rightarrow P_\varphi(\alpha) = 0$

Hence, # satisfying assignments is

$$K = \sum_{\sigma_1 \in \{0,1\}} \cdots \sum_{\sigma_n \in \{0,1\}} P_\varphi(\sigma_1, \ldots, \sigma_n) \qquad (**)$$

Do calculations in $(**)$ modulo some prime $p > 2^n \geq K$ so that answer mod p correct

High-level idea

Input: $\langle \varphi, K \rangle$

Prover sends prime $p \in (2^n, 2^{2n}]$

Verifier checks that p indeed prime
(can be done in poly time)

Then run protocol to check

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

for polynomial g with efficient representation so that it can be evaluated in poly time (holds for $g = P_\varphi$)

Observation If we plug in $x_i = b_i$ for $i = 2, \ldots, n$, then $g(x_1, b_2, \ldots, b_n)$ is univariate poly, and so is

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

The equality $(\dagger)$ holds iff $h(0) + h(1) = K$

So verifier can
(i) ask prover for $h(x_1)$
(ii) check $h(0) + h(1) = K$
(iii) check that prover didn't cheat with h.

Design protocol SumCheck(g, K, n) to check that n-variate polynomial g satisfies $(\dagger)$

SUMCHECK (g, K, n)

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

If $n \geq 2$, ask prover to send (efficient representation of) $h(x_1)$ in $(\star)$

↖ easy; h univariate polynomial

P: Reply with $s(x_1)$

V: Check if $s(0) + s(1) = K$ and reject o/w

Pick random $a \in [0, p-1]$

Let $K' = s(a)$

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

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

Recursive call to protocol checks that prover wasn't cheating but answered $s(x_1) = h(x_1)$ by verifying

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

But note that a is not $\in \{0,1\}$ but an arbitrary number in $[0, p-1]$!!

LEMMA 10

The protocol SUMCHECK (g, K, n) when run on a degree-d polynomial and computing modulo prime p has
- completeness 1
- soundness error $\leq dn/p$

~~~

In our case
φ has m clauses $\Rightarrow$ P_φ degree $\leq 3m$

m clauses over n variables $\Rightarrow$

$$m \leq \binom{n}{3} 3^3 \leq 27n^3 \quad [\text{why?}]$$

Pick $p > 2^n$

Get soundness error $\leq$

$$\frac{dn}{p} < \frac{81n^4}{2^n} \to 0 \text{ as } n \to \infty$$

So we just need to prove Lemma 10.

This will have to wait till next lecture ...
~~~

THEOREM 9 $coNP \subseteq IP$

Construct protocol for more general problem

$$\#SAT_D = \{ (\varphi, K) \mid \varphi \text{ 3-CNF with exactly } K \text{ satisfying assignments} \}$$

$K=0$ gives 3-SAT as special case

Write clause C_j as polynomial p_j

$$x_i \vee \bar{x}_j \vee x_k \rightsquigarrow 1 - (1-x_i)\bar{x}_j(1-x_k)$$

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

$$K = \sum_{\sigma_1 \in \{0,1\}} \cdots \sum_{\sigma_n \in \{0,1\}} P_\varphi(\sigma_1, \ldots, \sigma_n) \qquad (**)$$

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

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

$$h(x_1) = \sum_{\sigma_2 \in \{0,1\}} \cdots \sum_{\sigma_n \in \{0,1\}} g(x_1, \sigma_2, \ldots, \sigma_n) \qquad (\#)$$

(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 $a \in_R [0, p-1]$
$K' := s(a)$
$g' := g(a, x_2, \ldots, x_n)$
Run SUMCHECK $(g', K', n-1)$

NOTE that g' also has efficient representation

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

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

So coNP $\subseteq$ 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

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

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

0% probability of being fooled.

Inductive step: Want to detect if

$$\sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} g(x_1, \ldots, x_n) \neq K$$

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 (*)
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 ▨

We proved coNP ⊆ IP

Actually most of what is needed for PSPACE ⊆ IP except for some extra twists.

What was the key idea? ARITHMETIZATION

CNF formula φ ⟹ polynomial P_φ

Evaluate polynomial in much larger field ⇒ makes it practically impossible for prover to cheat.

Can also define MULTIPROVER INTERACTIVE PROTOCOLS (MIP). Provers agree beforehand on shared strategy but cannot communicate during protocol

PROVER 1 | VERIFIER | PROVER 2

$a_1 = f_1(x, r)$ ←

$b_1 = f_2(x, r)$ →

$a_2 = g_1(x, a_1)$ →

$b_2 = g_2(x, b_2)$ ←

$a_3 = f_1(x, r, a_1, a_2, b_1, b_2)$ ←

$b_3 = f_2(x, r, b_1, b_2, b_3, a_1, a_2)$ →

⋮

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 1! [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)

THM 12 PCP THEOREM [Arora-Safra '92] [Arora-Lund-Motwani-Sudan-Szegedi '94]

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

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 NON-ISOMORPHISM $\in PCP[poly(n), O(1)]$

$GNI = \{ \langle G_0, G_1 \rangle \mid G_0 \not\cong G_1 \}$

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 $G_0 \not\cong G_1$, prover constructs table π according to specification.
Verifier's test will always accept

Soundness (sketch)

If $G_0 \stackrel{c}{\equiv} G_1$, 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 $G_0 \stackrel{c}{\equiv} G_1$

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

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

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

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

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

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

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

$$= 1/2 \quad \text{as claimed}$$