Lecture 18

Jonathan Katz

1 The Power of $\mathcal{IP}$

We have seen a (surprising!) interactive proof for graph non-isomorphism. This begs the question: how powerful is $\mathcal{IP}$?

1.1 $\mathsf{co}\mathcal{NP} \subseteq \mathcal{IP}$

As a "warm-up" we show that $\mathsf{co}\mathcal{NP} \subseteq \mathcal{IP}$. We have seen last time that $\mathsf{co}\mathcal{NP}$ is unlikely to have a constant-round interactive proof system (since this would imply[1] that the polynomial hierarchy collapses). For this reason it was conjectured at one point that $\mathcal{IP}$ was not "too much more powerful" than $\mathcal{NP}$. Here, however, we show this intuition wrong: any language in $\mathsf{co}\mathcal{NP}$ has a proof system using a *linear* number of rounds.

We begin by *arithmetizing* a 3CNF formula ϕ to obtain a polynomial expression that evaluates to 0 iff ϕ has no satisfying assignments. (This powerful technique, by which a "combinatorial" statement about satisfiability of a formula is mapped to an *algebraic* statement about a polynomial, will come up again later in the course.) We then show how to give an interactive proof demonstrating that the expression indeed evaluates to 0.

To arithmetize ϕ, the prover and verifier proceed as follows: identify 0 with "false" and positive integers with "true." The literal x_i becomes the variable x_i, and the literal $\bar{x}_i$ becomes $(1-x_i)$. We replace "$\wedge$" by multiplication, and "$\vee$" by addition. Let Φ denote the polynomial that results from this arithmetization; note that this is an n-variate polynomial in the variables $x_1, \ldots, x_n$, whose total degree is at most the number of clauses in ϕ.

Now consider what happens when the $\{x_i\}$ are assigned boolean values: all literals take the value 1 if they evaluate to "true," and 0 if they evaluate to "false." Any clause (which is a disjunction of literals) takes a positive value iff at least one of its literals is true; thus, a clause takes a positive value iff it evaluates to "true." Finally, note that Φ itself (which is a conjunction of clauses) takes on a positive value iff all of its constituent clauses are positive. We can summarize this as: $\Phi(x_1, \ldots, x_n) > 0$ if $\phi(x_1, \ldots, x_n) = \text{TRUE}$, and $\Phi(x_1, \ldots, x_n) = 0$ if $\phi(x_1, \ldots, x_n) = \text{FALSE}$. Summing over all possible (boolean) settings to the variables, we see that

$$\phi \in \overline{\mathsf{SAT}} \Leftrightarrow \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(x_1, \ldots, x_n) = 0.$$

If ϕ has m clauses, then Φ has degree (at most) m (where the [total] degree of a polynomial is the maximum degree on any of its monomials, and the degree of a monomial is the sum of the degrees of its constituent variables). Furthermore, the sum above is at most $2^n \cdot 3^m$. So, if we work

[1] In more detail: a constant-round proof system for $\mathsf{co}\mathcal{NP}$ would imply a constant-round *public-coin* proof system for $\mathsf{co}\mathcal{NP}$, which would in turn imply $\mathsf{co}\mathcal{NP} \subseteq \mathbf{AM}$. We showed last time that the latter implies the collapse of PH.

modulo a prime $q > 2^n \cdot 3^m$ the above is equivalent to:

$$\phi \in \overline{\mathsf{SAT}} \;\Leftrightarrow\; \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(x_1, \ldots, x_n) = 0 \bmod q.$$

Working modulo a prime (rather than over the integers) confers two advantages: it keeps the numbers from getting too large (since all numbers will be reduced modulo q; note that $|q| = \log q$ is polynomial) and it means that we are working over the finite field $\mathbb{F}_q$ (which simplifies the analysis).

We have now reduced the question of whether ϕ is unsatisfiable to the question of proving that a particular polynomial expression sums to 0! This already hints at the power of arithmetization: it transforms questions of logic (e.g., satisfiability) into questions of abstract mathematics (polynomials, group theory, algebraic geometry, ...) and we can then use all the powerful tools of mathematics to attack our problem. Luckily, for the present proof the only "deep" mathematical result we need is that a non-zero polynomial of degree m over a field has at most m roots. An easy corollary is that two different polynomials of degree (at most) m can agree on at most m points.

We now show the *sum-check protocol*, which is an interactive proof that ϕ is not satisfiable.

- Both prover and verifier have ϕ. They both generate the polynomial Φ. Note that the (polynomial-time) verifier cannot write out Φ explicitly, but it suffices for the verifier to be able to evaluate Φ on any given values of $x_1, \ldots, x_n$. The prover wants to show that $0 = \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(x_1, \ldots, x_n)$.
- The prover sends a prime q such that $q > 2^n \cdot 3^m$. The verifier checks the primality of q. (The verifier could also generate q itself, and send it to the prover.) All subsequent operations are performed modulo q.
- The verifier initializes $v_0 = 0$.
- The following is repeated for $i = 1$ to n:
  - The prover sends a polynomial $\hat{P}_i$ (in one variable) of degree at most m.
  - The verifier checks that $\hat{P}_i$ has degree at most m and that $\hat{P}_i(0) + \hat{P}_i(1) = v_{i-1}$ (addition is done in $\mathbb{F}_q$). If not, the verifier rejects. Otherwise, the verifier chooses a random $r_i \in \mathbb{F}_q$, computes $v_i = \hat{P}_i(r_i)$, and sends r_i to the prover.
- The verifier accepts if $\Phi(r_1, \ldots, r_n) = v_n \bmod q$ and rejects otherwise. (Note that even though we originally only "cared" about the values Φ takes when its inputs are *boolean*, nothing stops us from evaluating Φ at any points in the field.)

Claim 1 *If ϕ is unsatisfiable then a prover can make the verifier accept with probability 1.*

For every $1 \leq i \leq n$ (and given the verifier's choices of $r_1, \ldots, r_{i-1}$) define the degree-m polynomial:

$$P_i(x_i) \stackrel{\text{def}}{=} \sum_{x_{i+1} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(r_1, \ldots, r_{i-1}, x_i, x_{i+1}, \ldots, x_n).$$

We claim that if ϕ is unsatisfiable and the prover always sends $\hat{P}_i = P_i$, then the verifier always accepts. In the first iteration ($i = 1$), we have

$$P_1(0) + P_1(1) = \sum_{x_1 \in \{0,1\}} \left(\sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(x_1, \ldots, x_n) \right) \;=\; 0 \;=\; v_0,$$

since ϕ is unsatisfiable. For $i > 1$ we have:

$$\begin{aligned} P_i(0) + P_i(1) &= \sum_{x_i \in \{0,1\}} \sum_{x_{i+1} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(r_1, \ldots, r_{i-1}, x_i, \ldots, x_n) \\ &= P_{i-1}(r_{i-1}) = v_{i-1}. \end{aligned}$$

Finally, $v_n \stackrel{\text{def}}{=} P_n(r_n) = \Phi(r_1, \ldots, r_n)$ so the verifier accepts.

Claim 2 *If ϕ is satisfiable, then no matter what the prover does the verifier will accept with probability at most nm/q.*

The protocol can be viewed recursively, where in iteration i the prover is trying to convince the verifier that

$$v_{i-1} = \sum_{x_i \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi_i(x_i, \ldots, x_n) \tag{1}$$

for some degree-m polynomial Φ_i that the verifier can evaluate. (In an execution of the protocol, we have $\Phi_1 = \Phi$; for $i > 1$ we have $\Phi_i(x_i, \ldots, x_n) = \Phi(r_1, \ldots, r_{i-1}, x_i, \ldots, x_n)$.) Each iteration i proceeds as follows: the prover sends some degree-m polynomial $P_i'(x_i)$ (this polynomial is supposed to be equal to $\sum_{x_{i+1} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi_i(x_i, \ldots, x_n)$ but may not be if the prover is cheating). The verifier checks that $\sum_{x_i \in \{0,1\}} P_i'(x_i) = v_{i-1}$ and, if so, then chooses a random point r_i; the prover then needs to convince the verifier that

$$v_i \stackrel{\text{def}}{=} P_i'(r_i) = \sum_{x_{i+1} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi_{i+1}(x_{i+1}, \ldots, x_n),$$

where $\Phi_{i+1}(x_{i+1}, \ldots, x_n) = \Phi_i(r_i, x_{i+1}, \ldots, x_n)$.

Looking now abstractly at Eq. (1), we claim that if Eq. (1) does *not* hold then the prover can make the verifier accept with probability at most km/q, where $k = n - i + 1$ is the number of variables we are summing over. The proof is by induction on k:

Base case. When $k = 1$ we have

$$v_n \neq \sum_{x_n \in \{0,1\}} \Phi_n(x_n) \tag{2}$$

but the prover is trying to convince the verifier otherwise. The prover sends some polynomial $P_n'(x_n)$. If $P_n' = \Phi_n$ the verifier always rejects since, by Eq. (2), $P_n'(0) + P_n'(1) \neq v_n$. If $P_n' \neq \Phi_n$, then the polynomials P_n' and Φ_n agree on at most m points; since the verifier chooses random r_n and accepts only if $P_n'(r_n) = \Phi_n(r_n)$, the verifier accepts with probability at most m/q.

Inductive step. Say the claim is true for some value of k, and look at Eq. (1) for $k+1$. Renumbering the variables, we have

$$v \neq \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_{k+1} \in \{0,1\}} \Phi(x_1, \ldots, x_{k+1}),$$

but the prover is trying to convince the verifier otherwise. The prover sends some polynomial $P'(x_1)$. Let $\hat{P}(x_1) = \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_{k+1} \in \{0,1\}} \Phi(x_1, \ldots, x_{k+1})$ (this is what the prover is "supposed" to send). There are again two possibilities: if $P' = \hat{P}$, then $P'(0) + P'(1) \neq v$ and the verifier always rejects. If $P' \neq \hat{P}$ then these polynomials agree on at most m points. So with probability at most m/q the verifier chooses a point r_1 for which $P'(r_1) = \hat{P}(r_1)$; if this happens, we will just say the prover succeeds. If this does *not* occur, then

$$v' \stackrel{\text{def}}{=} P'(r_1) \neq \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_{k+1} \in \{0,1\}} \Phi(r_1, x_2, \ldots, x_{k+1}),$$

and we have reduced to a case where we are summing over k variables. By our inductive assumption, the prover succeeds with probability at most km/q in that case. Thus, the overall success probability of the prover is at most $m/q + km/q \leq (k+1)m/q$. This completes the proof.

1.2 $\#\mathcal{P} \subseteq \mathcal{IP}$

(We have not yet introduced the class $\#\mathcal{P}$, but we do not use any properties of this class here.) It is relatively straightforward to extend the protocol of the previous section to obtain an interactive proof regarding the *number* of satisfying assignments of some 3CNF formula ϕ. We need only change the way we do the arithmetization: now we want our arithmetization Φ to evaluate to *exactly* 1 on any satisfying assignment to ϕ, and to 0 otherwise. For literals we proceed as before, transforming x_i to x_i and $\bar{x}_i$ to $1 - x_i$. For clauses, we do something different: given clause $a \vee b \vee c$ (where a, b, c are literals), we construct the polynomial:

$$1 - (1 - \hat{a})(1 - \hat{b})(1 - \hat{c}),$$

where $\hat{a}$ represents the arithmetization of a, etc. Note that if all of a, b, c are set to "false" (i.e., $\hat{a} = \hat{b} = \hat{c} = 0$) the above evaluates to 0 (i.e., false), while if any of a, b, c are "true" the above evaluates to 1 (i.e., true). Finally, arithmetization of the entire formula ϕ (which is the "and" of a bunch of clauses) is simply the product of the arithmetization of its clauses. This gives a polynomial Φ with the desired properties. Note that the degree of Φ is now (at most) $3m$, rather than m.

Using the above arithmetization, a formula ϕ has exactly K satisfying assignments iff:

$$K = \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi(x_1, \ldots, x_n).$$

Using the exact same protocol as before, except with $q > 2^n$ (since the above summation can now be at most 2^n) and setting $v_0 = K$ (the claimed number of satisfying assignments), gives an interactive proof for $\#SAT$ with soundness error $3mn/q$.