Lecture 19

Jonathan Katz

1 $\mathcal{IP}$ = PSPACE

A small modification of the previous protocol gives an interactive proof for any language in PSPACE, and hence PSPACE $\subseteq \mathcal{IP}$. Before showing this, however, we quickly argue that $\mathcal{IP} \subseteq$ PSPACE. To see this, fix some proof system $(\mathbf{P}, \mathbf{V})$ for a language L (actually, we really only care about the verifier algorithm $\mathbf{V}$). We claim that $L \in$ PSPACE. Given an input $x \in \{0,1\}^n$, we compute exactly (using polynomial space) the maximum probability with which a prover can make $\mathbf{V}$ accept. (Although the prover is allowed to be all-powerful, we will see that the optimal strategy can be computed in PSPACE and so it suffices to consider PSPACE provers in general.) Imagine a tree where each node at level i (with the root at level 0) corresponds to some sequence of i messages exchanged between the prover and verifier. This tree has polynomial depth (since $\mathbf{V}$ can only run for polynomially many rounds), and each node has at most 2^{n^c} children (for some constant c), since messages in the protocol have polynomial length. We recursively assign values to each node of this tree in the following way: a leaf node is assigned 0 if the verifier rejects, and 1 if the verifier accepts. The value of an internal node where the prover sends the next message is the *maximum* over the values of that node's children. The value of an internal node where the verifier sends the next message is the (weighted) *average* over the values of that node's children. The value of the root determines the maximum probability with which a prover can make the verifier accept on the given input x, and this value can be computed in polynomial space. If this value is greater than 2/3 then $x \in L$; if it is less than 1/3 then $x \notin L$.

1.1 PSPACE $\subseteq \mathcal{IP}$

We now turn to the more interesting direction, namely showing that PSPACE $\subseteq \mathcal{IP}$. We will now work with the PSPACE-complete language TQBF, which (recall) consists of true quantified boolean formulas of the form:

$$\forall x_1 \exists x_2 \cdots Q_n x_n \quad \phi(x_1, \ldots, x_n),$$

where ϕ is a 3CNF formula. We begin by arithmetizing ϕ as we did in the case of $\#\mathcal{P}$; recall, if ϕ has m clauses this results in a degree-$3m$ polynomial Φ such that, for $x_1, \ldots, x_n \in \{0,1\}$, we have $\Phi(x_1, \ldots, x_n) = 1$ if $\phi(x_1, \ldots, x_n)$ is true, and $\Phi(x_1, \ldots, x_n) = 0$ if $\phi(x_1, \ldots, x_n)$ is false.

We next must arithmetize the quantifiers. Let Φ be an arithmetization of ϕ as above. The arithmetization of an expression of the form $\forall x_n \, \phi(x_1, \ldots, x_n)$ is

$$\prod_{x_n \in \{0,1\}} \Phi(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \Phi(x_1, \ldots, x_{n-1}, 0) \cdot \Phi(x_1, \ldots, x_{n-1}, 1).$$

If we fix values for $x_1, \ldots, x_{n-1}$, then the above evaluates to 1 if the expression $\forall x_n \, \phi(x_1, \ldots, x_n)$ is true, and evaluates to 0 if this expression is false. The arithmetization of an expression of the form $\exists x_n\, \phi(x_1, \ldots, x_n)$ is

$$\coprod_{x_n \in \{0,1\}} \Phi(x_1, \ldots, x_n) \stackrel{\text{def}}{=} 1 - \big(1 - \Phi(x_1, \ldots, x_{n-1}, 0)\big) \cdot \big(1 - \Phi(x_1, \ldots, x_{n-1}, 1)\big)\,.$$

Note again that if we fix values for $x_1, \ldots, x_{n-1}$ then the above evaluates to 1 if the expression $\exists x_n\, \phi(x_1, \ldots, x_n)$ is true, and evaluates to 0 if this expression is false. Proceeding in this way, a quantified boolean formula $\exists x_1 \forall x_2 \cdots \forall x_n \phi(x_1, \ldots, x_n)$ is true iff

$$1 = \coprod_{x_1 \in \{0,1\}} \prod_{x_2 \in \{0,1\}} \cdots \prod_{x_n \in \{0,1\}} \Phi(x_1, \ldots, x_n). \tag{1}$$

A natural idea is to use Eq. (1) in the protocols we have seen for $\mathsf{co}\mathcal{NP}$ and $\#\mathcal{P}$, and to have the prover convince the verifier that the above holds by "stripping off" operators one-by-one. While this works in principle, the problem is that the *degrees* of the intermediate results are too large. For example, the polynomial

$$P(x_1) = \prod_{x_2 \in \{0,1\}} \cdots \prod_{x_n \in \{0,1\}} \Phi(x_1, \ldots, x_n)$$

may have degree as high as $2^n \cdot 3m$ (note that the degree of x_1 doubles each time a $\prod$ or $\coprod$ operator is applied). Besides whatever effect this will have on soundness, this is even a problem for completeness since a polynomially bounded verifier cannot read an exponentially large polynomial (i.e., with exponentially many terms).

To address the above issue, we use a simple[1] trick. In Eq. (1) the $\{x_i\}$ only take on *boolean* values. But for any $k > 0$ we have $x_i^k = x_i$ when $x_i \in \{0,1\}$. So we can in fact reduce the degree of every variable in any intermediate polynomial to (at most) 1. (For example, the polynomial $x_1^5 x_2^4 + x_1^6 + x_1^7 x_2$ would become $2x_1 x_2 + x_1$.) Let R_{x_i} be an operator denoting this "degree reduction" operation applied to variable x_i. Then the prover needs to convince the verifier that

$$1 = \coprod_{x_1 \in \{0,1\}} R_{x_1} \prod_{x_2 \in \{0,1\}} R_{x_1} R_{x_2} \coprod_{x_3 \in \{0,1\}} \cdots R_{x_1} \cdots R_{x_{n-1}} \prod_{x_n \in \{0,1\}} R_{x_1} \cdots R_{x_n} \Phi(x_1, \ldots, x_n).$$

As in the previous protocols, we will actually evaluate the above modulo some prime q. Since the above evaluates to either 0 or 1, we can take q any size we like (though soundness will depend inversely on q as before).

We can now apply the same basic idea from the previous protocols to construct a new protocol in which, in each round, the prover helps the verifier "strip" one operator from the above expression. Denote the above expression abstractly by:

$$F_\phi = \mathcal{O}_1 \mathcal{O}_2 \cdots \mathcal{O}_\ell\ \Phi(x_1, \ldots, x_n) \bmod q\,,$$

where $\ell = \sum_{i=1}^{n} (i+1)$ and each O_j is one of $\prod_{x_i}$, $\coprod_{x_i}$, or R_{x_i} (for some i). At every round k the verifier holds some value v_k and the prover wants to convince the verifier that

$$v_k = \mathcal{O}_{k+1} \cdots \mathcal{O}_\ell\ \Phi_k \bmod q,$$

[1] Of course, it seems simple in retrospect...

where Φ_k is some polynomial. At the end of the round the verifier will compute some v_{k+1} and the prover then needs to convince the verifier that

$$v_{k+1} = \mathcal{O}_{k+2} \cdots \mathcal{O}_\ell \; \Phi_{k+1} \bmod q,$$

for some Φ_{k+1}. We explain how this is done below. At the beginning of the protocol we start with $v_0 = 1$ and $\Phi_0 = \Phi$ (so that the prover wants to convince the verifier that the given quantified formula is true); at the end of the protocol the verifier will be able to compute Φ_ℓ itself and check whether this is equal to v_ℓ.

It only remains to describe each of the individual rounds. There are three cases corresponding to the three types of operators (we omit the " mod q" from our expressions from now on, for simplicity):

Case 1: $\mathcal{O}_{k+1} = \prod_{x_i}$ (for some i). Here, the prover wants to convince the verifier that

$$v_k = \prod_{x_i} R_{x_1} \cdots \coprod_{x_{i+1}} \cdots \prod_{x_n} R_{x_1} \cdots R_{x_n} \Phi(r_1, \ldots, r_{i-1}, x_i, \ldots, x_n). \tag{2}$$

(*Technical note*: when we write an expression like the above, we really mean

$$\left(\prod_{x_i} R_{x_1} \cdots \coprod_{x_{i+1}} \cdots \prod_{x_n} R_{x_1} \cdots R_{x_n} \Phi(x_1, \ldots, x_{i-1}, x_i, \ldots, x_n) \right) [r_1, \ldots, r_{i-1}].$$

That is, first the expression is computed symbolically, and then the resulting expression is evaluated by setting $x_1 = r_1, \ldots, x_{i-1} = r_{i-1}$.) This is done in the following way:

- The prover sends a degree-1 polynomial $\hat{P}(x_i)$.
- The verifier checks that $v_k = \prod_{x_i} \hat{P}(x_i)$. If not, reject. Otherwise, choose random $r_i \in \mathbb{F}_q$, set $v_{k+1} = \hat{P}(r_i)$, and enter the next round with the prover trying to convince the verifier that:

$$v_{k+1} = R_{x_1} \cdots \coprod_{x_{i+1}} \cdots \prod_{x_n} R_{x_1} \cdots R_{x_n} \Phi(r_1, \ldots, r_{i-1}, r_i, x_{i+1}, \ldots, x_n). \tag{3}$$

To see completeness, assume Eq. (2) is true. Then the prover can send

$$\hat{P}(x_i) = P(x_i) \stackrel{\text{def}}{=} R_{x_1} \cdots \coprod_{x_{i+1}} \cdots \prod_{x_n} R_{x_1} \cdots R_{x_n} \Phi(r_1, \ldots, r_{i-1}, x_i, \ldots, x_n);$$

the verifier will not reject and Eq. (3) will hold for any choice of r_i. As for soundness, if Eq. (2) does *not* hold then the prover must send $\hat{P}(x_i) \neq P(x_i)$ (or else the verifier rejects right away); but then Eq. (3) will not hold except with probability $1/q$.

Case 2: $\mathcal{O}_{k+1} = \coprod_{x_i}$ (for some i). This case and its analysis are similar to the above and are therefore omitted.

Case 3: $\mathcal{O}_{k+1} = R_{x_i}$ (for some i). Here, the prover wants to convince the verifier that

$$v_k = R_{x_i} \cdots \prod_{x_n} R_{x_1} \cdots R_{x_n} \Phi(r_1, \ldots, r_j, x_{j+1}, \ldots, x_n), \tag{4}$$

where $j \geq i$. This case is a little different from anything we have seen before. Now:

- The prover sends a polynomial $\hat{P}(x_i)$ of appropriate degree (see below).
- The verifier checks that $\left(R_{x_i}\hat{P}(x_i)\right)[r_i] = v_k$. If not, reject. Otherwise, choose a **new** random $r_i \in \mathbb{F}_q$, set $v_{k+1} = \hat{P}(r_i)$, and enter the next round with the prover trying to convince the verifier that:

$$v_{k+1} = \mathcal{O}_{k+2} \cdots \prod_{x_n} R_{x_1} \cdots R_{x_n} \Phi(r_1, \ldots, r_i, \ldots, r_j, x_{j+1}, \ldots, x_n). \tag{5}$$

Completeness is again easy to see: assuming Eq. (4) is true, the prover can simply send

$$\hat{P}(x_i) = P(x_i) \stackrel{\text{def}}{=} \mathcal{O}_{k+2} \cdots \prod_{x_n} R_{x_1} \cdots R_{x_n} \Phi(r_1, \ldots, r_{i-1}, x_i, r_{i+1}, \ldots, r_j, x_{j+1}, \ldots, x_n)$$

and then the verifier will not reject and also Eq. (5) will hold for any (new) choice of r_i. As for soundness, if Eq. (4) does *not* hold then the prover must send $\hat{P}(x_i) \neq P(x_i)$; but then Eq. (5) will not hold except with probability d/q where d is the degree of $\hat{P}$.

This brings us to the last point, which is what the degree of $\hat{P}$ should be. Except for the innermost n reduce operators, the degree of the intermediate polynomial is at most 2; for the innermost n reduce operators, the degree can be up to $3m$.

We may now compute the soundness error of the entire protocol. There is error $1/q$ for each of the n operators of type $\prod$ or $\coprod$, error $3m/q$ for each of the final n reduce operators, and error $2/q$ for all other reduce operators. Applying a union bound, we see that the soundness error is:

$$\frac{n}{q} + \frac{3mn}{q} + \frac{2}{q} \cdot \sum_{i=1}^{n-1} i = \frac{3mn + n^2}{q}.$$

Thus, a polynomial-length q suffices to obtain negligible soundness error.

Bibliographic Notes

The result that $\mathsf{PSPACE} \subseteq \mathcal{IP}$ is due to Shamir [3], building on [2]. The "simplified" proof given here is from [4]. Guruswami and O'Donnell [1] have written a nice survey of the history behind the discovery of interactive proofs (and the PCP theorem that we will cover in a few lectures).

References

[1] V. Guruswami and R. O'Donnell. A History of the PCP Theorem. Available at `http://www.cs.washington.edu/education/courses/533/05au/pcp-history.pdf`

[2] C. Lund, L. Fortnow, H.J. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *J. ACM* 39(4): 859–868 (1992). The result originally appeared in FOCS '90.

[3] A. Shamir. $\mathcal{IP} = \mathsf{PSPACE}$. *J. ACM* 39(4): 869–877 (1992). Preliminary version in FOCS '90.

[4] A. Shen. $\mathcal{IP} = \mathsf{PSPACE}$: Simplified Proof. *J. ACM* 39(4): 878–880 (1992).