**6.842 Randomness and Computation** April 27, 2022

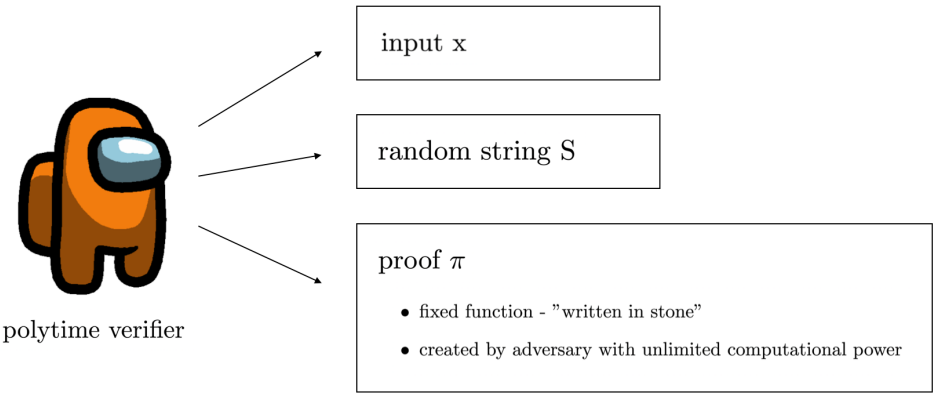
# Lecture 23

*Lecturer: Ronitt Rubinfeld* *Scribe: Jessica Wu, Guang Cui*

In this lecture, we covered PCPs (probabilistically checkable proofs), which give a way of writing a proof in a format where a verifier can check that the proof is correct without going line by line. In other words, the verifier can check that the proof is correct by looking only at a constant number of locations of the proof. Specifically, we showed that SAT (and thus NP) is a subset of PCP($n^3, 1$) by using arithmetization and several instances of tests that are based on Freivald's Matrix Multiplication checking algorithm.

## 1 Probabilistically Checkable Proofs

### 1.1 The PCP Model




The polytime verifier can look at the input, a random string, and a proof given by a fixed function.

**Definition 1** *$L \in PCP(r, q)$ if there exists P-TIME Turing machine $V$ such that*

*1) $\forall x \in L$ $\exists \pi$ such that $Pr_{random\ strings}[V, \pi\ accepts] = 1$*

*2) $\forall x \notin L$ $\forall \pi'$ such that $Pr_{random\ strings}[V, \pi'\ accepts] < \frac{1}{4}$*

*and $V$ uses $\leq r(n)$ random bits and makes $q(n)$ 1-bit queries to $\pi$.*

In other words, $L$ is in the $PCP$ class if there is some polynomial time verifier $V$ so that for every $x$ in the language, there is some way of writing the proof $\pi$ that convinces the verifier to accept, and for every $x$ not in the language there is no way to write the proof $\pi$ that makes the verifier convince more than a quarter of the random strings.

Note that the $\frac{1}{4}$ here is arbitrary, and any constant less than 1 would work. The parameters correspond to how long the random string needs to be, and how many queries the verifier needs to see of the proof. The existence of $PCP$s with good parameters for $r$ and $q$ have been linked to hardness of approximation problems.

### 1.2 Relation to NP

It's clear that $SAT \in PCP(0, n)$, where the proof could just be to write down the truth assignment for every variable of the satisfying assignment. By querying $n$ bits, the verifier could figure out the satisfying assignment and check that it is correct in polynomial time.

More generally, $NP \subseteq PCP(0, poly(n))$. We could write down the $NP$ proof, the verifier could look at it and verify that it is legal, and we would be set.

Can we reduce the number of query bits with an increase in random bits?

**Theorem 2** $SAT \subseteq PCP(O(n^3), O(1))$

We will prove this today.

**Corollary 3** $NP \subseteq PCP(O(n^3), O(1))$

**Theorem 4** $NP \subseteq PCP(O(\log n), O(1))$

This won't be covered, but it is a result of a recursive technique which uses Theorem 2 as a base case.

## 1.3 3-SAT

The 3-SAT problem is as follows:

- $F = \wedge C_i$ such that $C_i = (y_{i_1} \vee y_{i_2} \vee y_{i_3})$ where $y_{i_j} \in \{x_1 ... x_n \overline{x_1} ... \overline{x_n}\}$
- Is $F$ satisfiable?

A first crack at proving 3-SAT would be to use the $NP$ proof for SAT. We set the proof and protocol to be

$$\pi = \text{ settings of the satisfying assignment } a$$

$$V: \text{ pick random clause } C_i, \text{ check if setting } \overline{a} \text{ satisfies } C_i.$$

$V$ checks that the clause is satisfied by the fixed assignment. If $\overline{a}$ satisfies $C$ then $Pr[\ V \text{ succeeds}] = 1$.

However, if $\overline{a}$ doesn't satisfy $C$, there exists some clause $i$ such that $\overline{a}$ doesn't satisfy $C_i$, and the probability that the verifier is able to find such a clause could be very low. The $Pr[\ V \text{ finds unsatisfying } C_i] \geq \frac{1}{m}$, and since $m$ can be very large we would need to repeat $O(m)$ times to find the clause.

So, this solution doesn't work.

## 1.4 Arithmetization of SAT

We can arithmetize a Boolean formula $f$ using the following transformations:

| **boolean formula** $F$ | **arithmetic formula** $A(F)$ **over** $\mathcal{Z}_2$ |
|---|---|
| T | 1 |
| F | 0 |
| $X_i$ | $X_i$ |
| $\overline{X_i}$ | $1 - X_i$ |
| $\alpha \wedge \beta$ | $\alpha \cdot \beta$ |
| $\alpha \vee \beta$ | $1 - (1 - \alpha)(1 - \beta)$ |
| $\alpha \vee \beta \vee \gamma$ | $1 - (1 - \alpha)(1 - \beta)(1 - \gamma)$ |

For example,

- $(x_1 \vee x_2) \wedge \overline{x_3} \longrightarrow (1 - (1 - x_1)(1 - x_2)) \cdot (1 - x_3)$
- $x_1 \vee \overline{x_2} \vee x_3 \longrightarrow 1 - (1 - x_1)(1 - (1 - x_2))(1 - x_3) = 1 - (1 - x_1)x_2(1 - x_3)$. Observe that when we arithmetize a clause with 3 literals, the max degree of $A(F)$ is 3.

**Theorem 5** *$F$ satisfied by assignment $a$ iff $A(a) = 1$.*

## 1.5 Strange Arithmetization

Going back to our formulas for 3-SAT, we are going to arithmetize each clause separately and think of it as a vector, rather than multiply them all together (to limit the degree).

For $x = (x_1...x_n)$, set

$$C(x) = (\hat{C}_1(x), \hat{C}_2(x), ...)$$

where the components of the vector are the complements of each clause $C_i$, which evaluate to 0 if $x$ satisfies $C_i$. The motivation for taking the complement here is that it's easier to check if all the clauses evaluate to 0 rather than 1 in our vector, and all values should be 1 if the assignment is indeed satisfying.

Each $\hat{C}_i(x)$ is degree $\leq 3$ polynomial in $X$, and the verifier knows the coefficients of the arithmetization (but not the assignment). We need to convince the verifier that $C(a) = (0, 0, ..., 0)$ without sending assignment $a$.

*An idea*: How can we apply Freivald's test (HW 1)? Assume there exists some little birdie who tells $V$ dot products of $C(a)$ with random vectors mod 2.

**Theorem 6** *(Freivald's Test)*

- *If vectors $a \neq b$ then $Pr_{r\in\{0,1\}^n}[a \cdot r \neq b \cdot r] \geq \frac{1}{2}$.*
- *If matrices $A \cdot B \neq C$ then $Pr_{r\in\{0,1\}^n}[A \cdot B \cdot r \neq C \cdot r] \geq \frac{1}{2}$.*

**Proof** Pair vectors that differ in coordinate $i$ such that $a_i \neq b_i$ or $A \cdot B_{ij} = C_{ij}$ (also similar to the proof of orthogonality). ■

## 1.6 Freivald's test on $C(a)$

Suppose trustworthy birdie tells me, on my request $r$, dot products of $C \cdot r$.

Fix $a$:

$$(\hat{C}_1(a), ..., \hat{C}_m(a)) \cdot (r_1...r_m) = \sum r_i \hat{C}_i(a) \bmod 2$$

By an application of Frievald's with $b = 0$

$$Pr[\sum r_i \hat{C}_i(a) \equiv 0 \bmod 2] = \begin{cases} 1 & \text{if } \forall i\ \hat{C}_i(a) = 0 \ (C(a) \text{ satisfied}) \\ 1/2 & \text{otherwise } (C(a) \text{ not satisfied}) \end{cases}$$

But why would we believe the birdie? We need to hard code the answers to the birdie into the proof.

## 1.7 Believing the Birdie

Remember that we choose the $r_i$'s (so we don't have to worry about that), we know the coefficients of polynomials in the $\hat{C}_i$'s, and polynomials of $\hat{C}_i$'s are degree at most 3 in $a_i$'s.

Here's an example for clarity: $(x_1 \vee \overline{x_2} \vee x_3)(\overline{x_1} \vee x_2)$ when arithmetized becomes

$$(\overline{(1 - x_2 + x_1x_2 + x_2x_3 - x_1x_2x_3)}, \overline{(1 - x_1 + x_1x_2)})$$
$$\implies ((x_2 - x_1x_2 - x_2x_3 + x_1x_2x_3), (x_1 - x_1x_2))$$

Upon doing an inner product with a random vector $(r_1, r_2)$, this becomes

$$r_1 \cdot (x_2 - x_1x_2 - x_2x_3 + x_1x_2x_3) + r_2 \cdot (x_1 - x_1x_2)$$
$$= 0 \cdot 1 + r_2 \cdot x_1 + r_1 \cdot x_2 - (r_1 + r_2) \cdot x_1x_2 - r_1 \cdot x_2x_3 + 0 \cdot x_1x_3 + r_1 \cdot x_1x_2x_3$$

after separating by terms $x_i$. In general, for any 3-SAT expression,

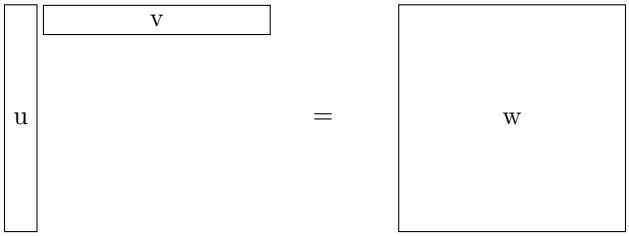
$$\sum r_i \hat{C}_i(a) = \Gamma + \sum_i a_i \alpha_i + \sum_{i,j} a_i a_j \beta_{ij} + \sum_{i,j,k} a_i a_j a_k \gamma_{ijk} \pmod 2$$

Where we're replacing $\alpha_i \to x_i, \beta_{ij} \to y_{ij}, \gamma_{ijk} \to z_{ijk}$, which has no relation to the 3-SAT variables.

V knows $\Gamma, \alpha_i, \beta_{ij}, \gamma_{ijk}$, since they depend on $r_i$'s and coefficients of polynomials. They don't depend on $a_i$'s, are computed by V, and are values $\in \{0, 1\}$ since we are working in mod 2. V does not know the $a_i$'s (formerly the $x_i$'s).

Crucially, the max total degree is 3. A couple definitions before we describe the proof:

**Definition 7** *Outer product* $w = u \circ v$ *if* $w_{ij} = u_i \cdot v_j$




Sort of like the opposite of an inner product in some sense, when you multiply two polynomials you're doing an outer product.

**Definition 8**

$$A\text{: } \mathbb{F}_2^n \to \mathbb{F}_2 \qquad A(x) = \textstyle\sum_i a_i x_i = a^T \cdot x$$

$$B\text{: } \mathbb{F}_2^{n^2} \to \mathbb{F}_2 \qquad B(y) = \textstyle\sum_{i,j} a_i a_j y_{ij} = (a \circ a)^T \cdot y$$

$$C\text{: } \mathbb{F}_2^{n^3} \to \mathbb{F}_2 \quad C(z) = \textstyle\sum_{i,j,k} a_i a_j a_k z_{ijk} = (a \circ a \circ a)^T \cdot z$$

Note that the verifier knows the $x, y, z$.

## 1.8 Proof $\Pi$

Now we are finally ready for the proof! Proof $\Pi$ will give the complete truth tables $\tilde{A}, \tilde{B}, \tilde{C}$ for all possible settings of $x, y, z$, which the verifier will need to check are actually correct and valid A, B, and C.

This is quite a long proof ($2^{n^3}$ size), and V only really needs to know A, B, C at some locations, but the other entries will help in checking.

Essentially, every time you call the birdie, you ask for a different value. The verifier needs to check two things in $\Pi$.

1. $\tilde{A}, \tilde{B}, \tilde{C}$ are of the right form:
   - all are linear functions (we can only test close-to-linear but can from previous lecture use the self-corrector to get the corresponding linear function)
   - they correspond to the same assignment $a$ (test that the self-corrections are consistent with $\tilde{A}(x) = a^T \cdot x \implies \tilde{B}(y) = (a \circ a)^T \cdot y \implies \tilde{C}(z) = (a \circ a \circ a)^T \cdot z$
2. Check that $a$ is a SAT assignment. In particular, that all $\hat{C}_i$'s evaluate to 0 on $a$.

Remember that it needs to do all of this in a constant number of queries!

## 1.9 Solving 2.

We call the self-corrector to obtain $a$, $a \circ a$, $a \circ a \circ a$. We don't know $a$ but it represents the assignment.

To test satisfiability, we pick $r \in \mathbb{Z}_2^n$, compute $\Gamma, \alpha_i(x), \beta_{ij}, \gamma_{i,j,k}(x,y,z)$, query the proof to get $SC - \tilde{A}(\alpha) = w_0$ and corresponding $w_1, w_2$, and finally verify that $0 = \Gamma + w_0 + w_1 + w_2$.

If $\hat{C}_i$ didn't evaluate to 0, then by Frievald's, there's $\leq 1/2$ chance it passes, so our test is $\geq 1/2$ accurate. To get higher accuracy, we simply run this $k$ times.

## 1.10 Solving 1.

First we test that $\tilde{A}, \tilde{B}, \tilde{C}$ are all 1/8-close to linear via a linearity test and fail if it's not. We've shown earlier this is possible using constant queries.

Now, we access $\tilde{A}$ etc. via the self-corrector on all inputs. The question is, how can we trust that the self-correcting values are consistent with other ($b$ actually equals $a \circ a$)?

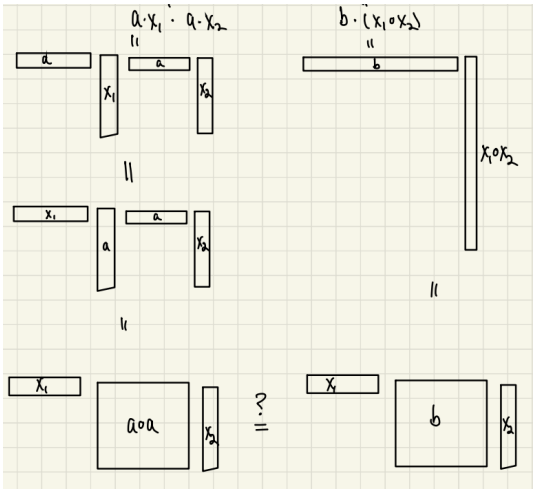
Again, we'll use more Frievald's!

We test that:

$$SC - \tilde{A}(x_1) \cdot SC - \tilde{A}(x_2) = \sum_i a_i x_{1i} \cdot \sum_j a_j x_{2j} = \sum_{ij} a_i a_j x_{1i} x_{2j} = SC - \tilde{B}(x_1 \circ x_2)$$

.

And the corresponding expression for $\tilde{C}$. Clearly, if $b = a \circ a$, then the test passes. If not, we'll use commutative properties of outer products and Frievald's to upper bound the chance it passes by 1/4.




Again, a similar argument holds for $\tilde{C}$.

Whew, that was a lot, but we're finally done! We have a proof of 3-SAT where a verifier can check only a constant number of bits and then accept or reject the proof with high probability of being correct.