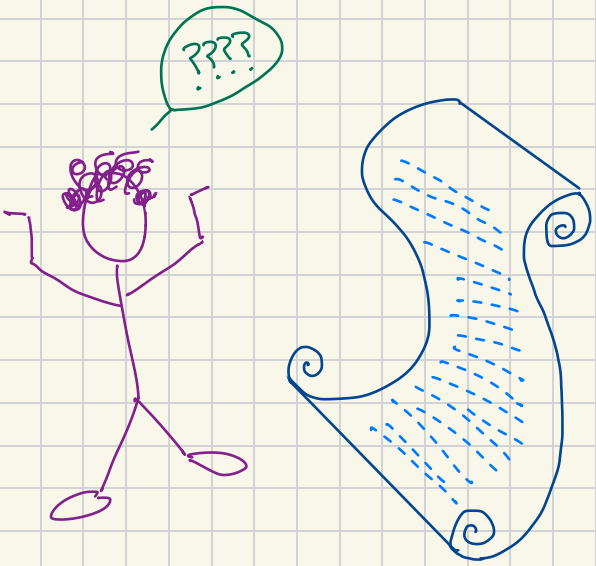
# Lecture 22

## Probabilistically Checkable Proof Systems



**Review:** $x, y \in \{0,1\}^n$

$x \cdot y = \sum x_i y_i \pmod 2$ "inner product"

$x \circ y = (x_1 y_1, x_1 y_2, \ldots, x_i y_j, \ldots x_n y_n)$ "outer product"

$x, y$: n-bit vectors; $x \circ y$: $n^2$ bit vector

**Fact** 1) if $\bar{a} \neq \bar{b}$ then $\Pr_{\bar{r} \in \{0,1\}^n}[\bar{a} \cdot \bar{r} \neq \bar{b} \cdot \bar{r}] \geq \frac{1}{2}$

2) if $A \cdot B \neq C$ then $\Pr_{\bar{r}}[A \cdot B \cdot \bar{r} \neq C \cdot \bar{r}] \geq \frac{1}{2}$

(also true for $\equiv \bmod 2$)

(Proof in pset 1)

**Self-correcting:** if $f$ $\frac{1}{8}$-linear then SC-$f$ is linear & uses $O(\log \frac{1}{\beta})$ queries to $f$ (but may fail on any input with prob $\leq \beta$)

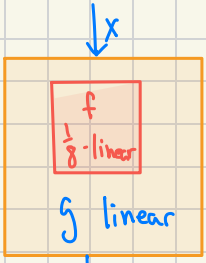
if $f$ $\frac{1}{8}$-linear

define $g(x)$: Do $\log \frac{1}{\beta}$ times

pick $y$ randomly

$\text{answer}_i \leftarrow f(y) + f(x-y)$

Output most common answer




then $\forall x$, $\Pr[g(x) = f(x)] \geq 1 - \beta$

## self-testing:

Given $f$

Do $O(1/\varepsilon)$ times:

Pick $x, y$ randomly

if $f(x+y) \neq f(x) + f(y)$ output "fail" & halt

Output "pass"

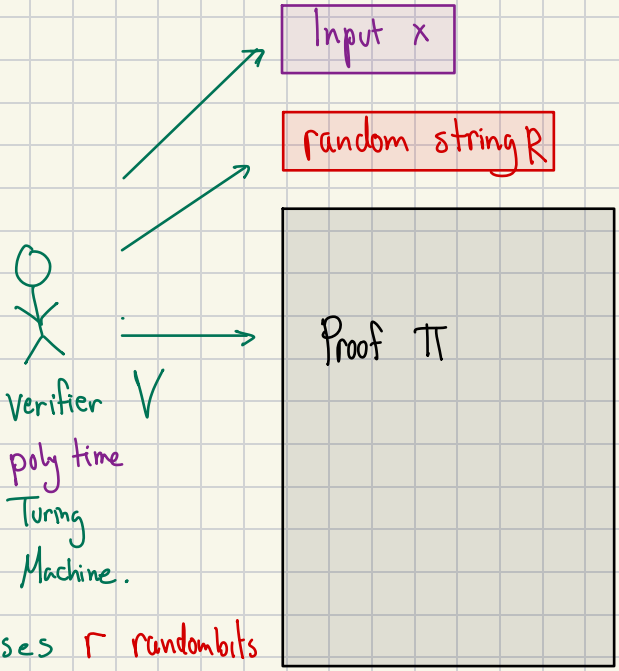
If $f$ linear, test passes

If $f$ $\varepsilon$-far from linear, $\Pr[\text{fail}] \geq 3/4$

$\Rightarrow$ Given $f$, run self-tester

if passes, safe to use with self-corrector

# Probabilistically Checkable Proofs




fixed function

"written in stone"
(or at least pen)

but

created by adversary with unlimited computational power.

does not change after queries from verifier

Query to π: "what is $i^{th}$ bit/word?"

def $L \in PCP(r, q)$ if $\exists V$ (ptime TM) s.t.

1) $\forall x \in L \quad \exists \pi$ s.t. $\Pr_{\substack{V\text{'s random} \\ \text{string}}} [V, \pi \text{ accepts}] = 1$

2) $\forall x \notin L \quad \forall \pi' \quad \Pr_{\substack{V\text{'s random} \\ \text{string}}} [V, \pi' \text{ accepts}] \leq 1/4$

note: $SAT \in PCP(0, n)$ ← proof is satisfying assignment. V doesn't need randomness to check.

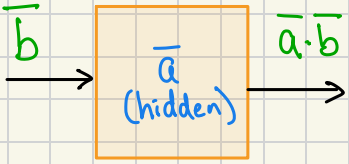
Today: $NP \subseteq PCP(O(n^3), O(1))$

Actually: $NP \subseteq PCP(O(\log n), O(1))$

Crazy? V only sees constantly many bits!

Note: runtime of verifier is poly(n), but only makes $O(1)$ queries to $\pi$

# A warm up:




(recall: Fact if $\bar{a} \neq \bar{b}$ then $\Pr_{\bar{r} \in \{0,1\}^n}[\bar{a}\cdot\bar{r} \neq \bar{b}\cdot\bar{r}] \geq \frac{1}{2}$)

Strange Setting: get "inner product queries" in one step
(only queries cost, computations are free.)

Test if $\bar{a} = \bar{0}$ in constant queries:

Do several times:
  pick $\bar{r} \in_u \{0,1\}^n$
  if $\bar{a}\cdot\bar{r} \neq 0$ output "FAIL" + halt
Output "PASS"

Behavior:
if $\bar{a} = \bar{0}$ will always pass
if $\bar{a} \neq \bar{0}$, fact $\Rightarrow$ fail each time with prob $\geq 1/2$

why should we believe answers to queries are correct?
consistent with each other?
consistent with $\bar{a}$? ← even if $\bar{a} \neq 0$, oracle could always answer 0 & try to fool us?

# Making the warmup "less strange":

fix vector $\bar{a} = (a_1, a_2, \ldots, a_n)$

allowed operations:

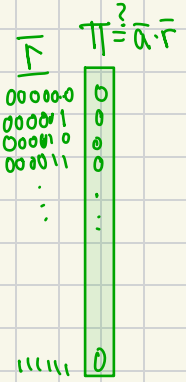
- query $a_i$
- specify $\bar{r}$ & query $\bar{a} \cdot \bar{r}$

} 1 step each

↖ 1st query is actually special case of $\bar{r} = e_i$

First idea for proof $\pi$:

note size of proof is $2^n$ → write out all answers to $\bar{a} \cdot \bar{r}$ (for each $\bar{r}$)




Test if $\bar{a} = \bar{0}$:

Do several times:
pick $\bar{r} \in_u \{0,1\}^n$
ask proof for value of $\bar{a} \cdot \bar{r}$
if $\bar{a} \cdot \bar{r} \neq 0$ output "FAIL" & halt
Output "PASS"

Problem: proof can cheat by writing all 0's in $\pi$ (and many other ways)

How can we test that $\pi$ doesn't cheat?

(will come back to this)

Idea: proof is a (extremely inefficient) way of "encoding" $\bar{a}$

# Back to 3SAT:

3SAT: $F = \bigwedge C_i$ s.t. $C_i = (y_{i_1} \vee y_{i_2} \vee y_{i_3})$

where $y_{i_j} \in \{x_1 \cdots x_n, \bar{x}_1 \cdots \bar{x}_n\}$ "literals"

↖ here $\bar{x}_i$ is "complement"

**Warning:**

FROM NOW ON, LOSING "BAR" FROM VECTORS TO PREVENT CONFUSION

## First crack:

$\Pi$ = setting of sat assignment $a$

$a_1 = T, a_2 = F, a_3 = T, \ldots$ | 1 0 1 . . .

Possible protocol for $V$, given $F, \Pi$:

pick random clause $C_i$

Check if $a$ satisfies $C_i$ ← 3 queries to $\Pi$

behavior:

if $a$ satisfies $C$, then

$\Pr[\text{Pass}] = 1$

if $a$ doesn't satisfy $C_i$ then

$\Pr[\text{fail}] \geq \Pr[\text{pick } i \text{ s.t. } C_i(A) = F] \geq \frac{1}{\#\text{clauses}}$ ☹ not very good bound

e.g. $F = (x_1 \vee \bar{x}_2 \vee x_3)(x_2 \vee \bar{x}_3 \vee x_4)$

$a = (1, 0, 0, 0, \ldots)$

random clause $(x_2 \vee \bar{x}_3 \vee x_4)$

passes F T F ✓

# Arithmetization of 3SAT:

Boolean formula $F$ $\Leftrightarrow$ arithmetic formula $A(F)$ over $\mathbb{Z}_2$ (mod 2)

$$
\begin{aligned}
T &\Leftrightarrow 1 \\
F &\Leftrightarrow 0 \\
X_i &\Leftrightarrow x_i \\
\bar{X}_i &\Leftrightarrow 1 - x_i \\
\alpha \wedge \beta &\Leftrightarrow \alpha \cdot \beta \\
\alpha \vee \beta &\Leftrightarrow 1 - (1-\alpha)(1-\beta) \\
\alpha \vee \beta \vee \gamma &\Leftrightarrow 1 - (1-\alpha)(1-\beta)(1-\gamma)
\end{aligned}
$$

example $X_1 \vee \bar{X}_2 \vee X_3 \Leftrightarrow 1 - (1-x_1)(x_2)(1-x_3)$

Key point: $F$ satisfied by $a$ iff $A(F)(a) = 1$

Issue 1: we like low degree polys. $A(F)$ can have very high degree

Idea 1: lets deal with each clause separately with some $a_i$

Issue 2: we are good at 0-testing, not 1-testing!

Idea 2: consider complement

## Vector of clause arithmetizations:

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

s.t. $\hat{C}_i$ = complement of arithmetization of clause $C_i$

$\Rightarrow$ evaluates to 0 if $x$ satisfies $C_i$

$\Rightarrow \hat{C}(x) = (0, \dots, 0)$ if $x$ satisfies $F$

Observe

(1) each $\hat{C}_i$ is deg $\leq 3$ poly in $x$

(2) $V$ knows coeffs of each $\hat{C}_i$

need to convince $V$ that $\hat{C}(a) = (\hat{C}_1(a), \hat{C}_2(a), \dots)$
$= (0, 0, \dots 0)$

without $V$ reading all of $a$

## Two big ideas

1) encoding of assignment
   - can access any linear fctn of it + any bit in consistent fast way

2) Ensure that lots of simultaneous good things happen
   - encode algebraically so that "0 ≡ good"
   - test if vector ≡ (00...0)

Summary:

High level idea: Special encoding of assignment

Encode satisfiability of $F$ as a collection of polys in vars of assignment

- one for each clause
- eval to 0 if assignment satisfies clause
- low degree
- $V$ knows coeffs - depend on structure of clause + vars of clause

Note: we are only concerned that $V$ is poly time, here will not be sublinear

but queries to proof should be constant!

## Idea for proof $\Pi$

- proof contains $\hat{C}(a) \cdot r \quad \forall r \in \{0,1\}^n$
- if $\forall i,\ \hat{C}_i(a) = 0$, $\Pr_r[\hat{C}(a) \cdot r = 0] = 1$

  if $\exists i$ s.t. $\hat{C}_i(a) \neq 0$, $\Pr_r[\hat{C}(a) \cdot r = 0] = 1/2$

  $\Pr_r[\hat{C}(a) \cdot r = 1]$

mod 2 arithmetic

recall the problem from before ...

proof could write all 0's even if $\hat{C}(a) \cdot r \neq 0$, so need to do more

## What does $\hat{C}(a) \cdot r$ look like?

sum of constant terms; degree 1 terms; degree 2 terms; degree 3 terms; V doesn't know

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

V does know: depends on $r_i$'s & coeffs in $\hat{C}_i$

From here on:

$\alpha_i \to x_i$

$\beta_{ij} \to y_{ij}$

$\gamma_{ijk} \to z_{ijk}$

} no relation to vars of 3SAT

note that $+1 = -1$ in $\mathbb{F}_2$ ($+ \bmod 2$)

example

$G = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2)$

$A(C_1) = 1 - (1-x_1)(1-x_2) = x_1 + x_2 - x_1x_2$ ← evaluates to 1 if $C_1$ satisfied

$\Rightarrow C_1^0(A) = 1 - a_1 - a_2 + a_1a_2$ ← "$1 - A(C_1)$" evaluates to 0 if $C_1$ satisfied

$A(C_2) = 1 - (x_1)(1-x_2) = 1 - x_1 + x_1x_2$

$\Rightarrow C_2^0(A) = a_1 - a_1a_2$

$C^0(x) = (1 - a_1 - a_2 + a_1a_2,\ a_1 - a_1a_2)$

$$\sum r_i C_i^0(a) = r_1(1 - a_1 - a_2 + a_1a_2) + r_2(a_1 - a_1a_2)$$

$$= \underbrace{r_1 \cdot 1 + r_2 \cdot 0}_{\deg 0} + \underbrace{(-r_1 + r_2)\cdot a_1 + (-r_1)\cdot a_2}_{\deg 1} + \underbrace{(r_1 - r_2)a_1a_2}_{\deg 2}$$

| Oracle provides answers to these queries $r_1$ $r_2$ | $\sum r_i C_i^0(a)$ | oracle for $a^+$: sat case $a^+ = (0,1)$ | oracle for $a^-$: unsat case $a^- = (0,0)$ |
|---|---|---|---|
| 0 0 | 0 | 0 | 0 |
| 0 1 | $a_1 - a_1a_2$ | 0 | 0 |
| 1 0 | $1 - a_1 - a_2 + a_1a_2$ | $1-0-1+0=0$ | $1-0-0+0=1$ |
| 1 1 | $1 - a_2$ | $1-1=0$ | $1-0=1$ |

= deg 0
= deg 1
= deg 2

# High level idea for proof: Special encoding of assignment

- proof writes out

| | | | |
|---|---|---|---|
| all | linear | fctns | of assignment |
| | deg 2 | " | " |
| | deg 3 | " | " |

↑ from this, V can compute whichentries it needs for previous test

- $A_a(x) = a \cdot x^T$

  $B_a(y) = (a \circ a)^T \cdot y$

  $C_a(z) = (a \circ a \circ a)^T \cdot z$

all linear fctns

⇒ can property test linearity & self-correct

Tester: • makes sure $A_a, B_a, C_a$ are close to linear
• uses self-corrected version of them to always get linear fctn.

But (1) what if they come from different assignments $a, a', a''$?
(2) how do we know $a$ is satisfying?

## More details:

(drop subscript "a" from fctns)

$A$ = all linear fctns evaluated at assignment $a$

$A : \mathbb{F}_2^n \to \mathbb{F}_2$ $\quad A(x) = \sum a_i x_i = a^T \cdot x$

$B$ = all degree 2 fctns evaluated at $a$

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

$C$ = all degree 3 fctns evaluated at $a$

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

**Proof II:**

Complete input/output tables of $\tilde{A}, \tilde{B}, \tilde{C}$

hopefully $A, B, C$ but need to check

actually one per choice of $r_i$'s in $\bar{0}$-test.

we only care about one row:

$x = \alpha, \; y = \beta, \; z = \gamma$ ← chosen from $r_i$'s & coeffs of $C_i$'s

but extra info helps us check consistency

# What does verifier check in proof?

(1) $\tilde{A}, \tilde{B}, \tilde{C}$ in "right form"

- all are linear fctns

can only test $\varepsilon$-linear but can self-correct to access true linear fctn

- correspond to same assignment $a$

ie. $\tilde{A}(x) = a^T x \Rightarrow \tilde{B}(y) = (a \otimes a)^T \cdot y \Rightarrow \tilde{C}(z) = (a \otimes a \otimes a)^T \cdot z$

Test consistency of self-correctors

(2) $a$ is satisfying assignment

(check entries in encoding which give value of $C(a)$)

- all $\hat{C}_i$'s evaluate to 0 on $a$

Part (2): assume $\vec{A}, \vec{B}, \vec{C}$ linear & correspond to same a

## Satisfiability Test:

Pick $r \in \mathbb{F}_2^n$

Compute $\Gamma$, $\alpha_i$'s, $\beta_{ij}$'s, $\gamma_{ijk}$'s ← fctns of r & coeffs of deg 3 polys

$\alpha_i$'s → $x_i$'s, $\beta_{ij}$'s → $y_{ij}$'s, $\gamma_{ijk}$'s → $z_{ijk}$'s

query proof to get

$SC\text{-}\tilde{A}(\alpha_1 \dots \alpha_n) = w_0$

$SC\text{-}\tilde{B}(\beta_{11} \dots \beta_{nn}) = w_1$

$SC\text{-}\tilde{C}(\gamma_{111} \dots \gamma_{nnn}) = w_2$

fix after next page: can only test "close" to linear so call self-corrector to ensure linearity

Verify $0 = \Gamma + w_0 + w_1 + w_2 \pmod 2$

↑ hopefully means $\sum r_i \hat{C}_i(a) = 0$

### Behavior of test:

if $\forall i \ \hat{C}_i(a) = 0$, $\Pr[\text{pass}] = 1$ ✓

if $\exists i$ s.t. $\hat{C}_i(a) \neq 0$,

Fact $\Rightarrow \Pr[\sum r_j \hat{C}_j(a) = 0] = 1/2$

So after $k$ times, $\Pr[\text{pass}] \leq 1/2^k$

# random bits = $O(n)$

# queries to proof = $O(1)$

# Testing 1:

(1) $\tilde{A}, \tilde{B}, \tilde{C}$ in "right form"

- all are linear fctns

Problem? can only test $\varepsilon$-linear but can self-correct to access "true" linear fctn

- correspond to same assignment $a$

i.e. $\tilde{A}(x) = a^T x \Rightarrow \tilde{B}(y) = (a \otimes a)^T \cdot y \Rightarrow \tilde{C}(z) = (a \otimes a \otimes a)^T z$

Test consistency of self-corrections

- Use linearity test on $\tilde{A}, \tilde{B}, \tilde{C}$

if linear pass

if $\gamma\delta$-far, fail with prob $\geq 1 - \delta'$

(pick $\delta'$ really small, but constant, so that all $(\delta')$'s add up to $< \delta$)

if pass $\Rightarrow$ $\gamma\delta$-linear $\Rightarrow$ can use with self-corrector to get linear fctns $sc\text{-}\tilde{A}, sc\text{-}\tilde{B}, sc\text{-}\tilde{C}$

$= a^T \cdot x \quad = b^T \cdot y \quad = c^T \cdot z$

from now on.

- want that $sc\text{-}\tilde{A} = a^T \cdot x \Rightarrow sc\text{-}\tilde{B} = (a \otimes a)^T \cdot y \Rightarrow sc\text{-}\tilde{C} = (a \otimes a \otimes a)^T z$

Now we can "finish" part 2 assuming part 1 done:

**Testing 2:** assuming $\tilde{A}, \tilde{B}, \tilde{C}$ $\Sigma$-linear + correspond to same $a$

(2) $a$ is satisfying assignment

(check entries in encoding which give value of $\mathcal{C}(a)$)

- all $\hat{C}_i$'s evaluate to 0 on $a$

- call self-correctors $\Rightarrow$ recover linear fctns $a$, $aoa$, $aoaoa$
- $a$ represents assignment, but we dont know it
- $a$ satisfies formula $\Leftrightarrow$ $\mathcal{C}(a) = (\hat{C}_1(a), \hat{C}_2(a), \ldots) = (0, 0, 0 \ldots, 0)$

**Modification:** replace queries to $\tilde{A}, \tilde{B}, \tilde{C}$ by calls to self corrector in previous:

query proof to get

$$SC\text{-}\tilde{A}(\alpha_1, \ldots \alpha_n) = w_0$$
$$SC\text{-}\tilde{B}(\beta_{11} \cdots \beta_{nn}) = w_1$$
$$SC\text{-}\tilde{C}(\gamma_{111} \cdots \gamma_{nnn}) = w_2$$

Can only test "close" to linear so call self-corrector to ensure linearity

Back to testing 1: (still need to make sure $\tilde{A}, \tilde{B}, \tilde{C}$ correspond to same $a$)

Test that $sc\text{-}\tilde{A} = a^T \cdot x \Rightarrow sc\text{-}\tilde{B} = (a \circ a)^T \cdot y \Rightarrow sc\text{-}\tilde{C} = (a \circ a \circ a)^T \cdot z$ :

## Outer Product Tester:

Pick random $x_1, x_2, x, y$

Test $sc\text{-}\tilde{A}(x_1) \cdot sc\text{-}\tilde{A}(x_2)$

$$= \left(\sum a_i x_{1i}\right) \cdot \left(\sum a_j x_{2j}\right)$$

$$= \sum_{ij} a_i a_j x_{1i} x_{2j} = \sum_{ij} b_{ij} x_{ij} x_{2j}$$

true if $b_{ij} = a_i \cdot a_j \; \forall i,j$

$$= sc\text{-}\tilde{B}(x_1 \circ x_2)$$

not uniformly distributed

holds due to $\varepsilon$-linear + self-corrector

Test $sc\text{-}\tilde{A}(x) \cdot sc\text{-}\tilde{B}(y)$

$$= \left(\sum a_i x_i\right) \cdot \left(\sum b_{jk} \cdot y_{jk}\right)$$

$$= \sum_{ijk} (a_i \cdot b_{jk}) x_i y_{jk} = \sum a_i a_j a_k x_i y_{jk}$$

true if $a_i \cdot b_{jk} = c_{ijk}$ $\forall i,j,k$

$$= \sum_{ijk} c_{ijk} x_i \cdot y_{jk}$$

$$= sc\text{-}\tilde{C}(x \circ y)$$

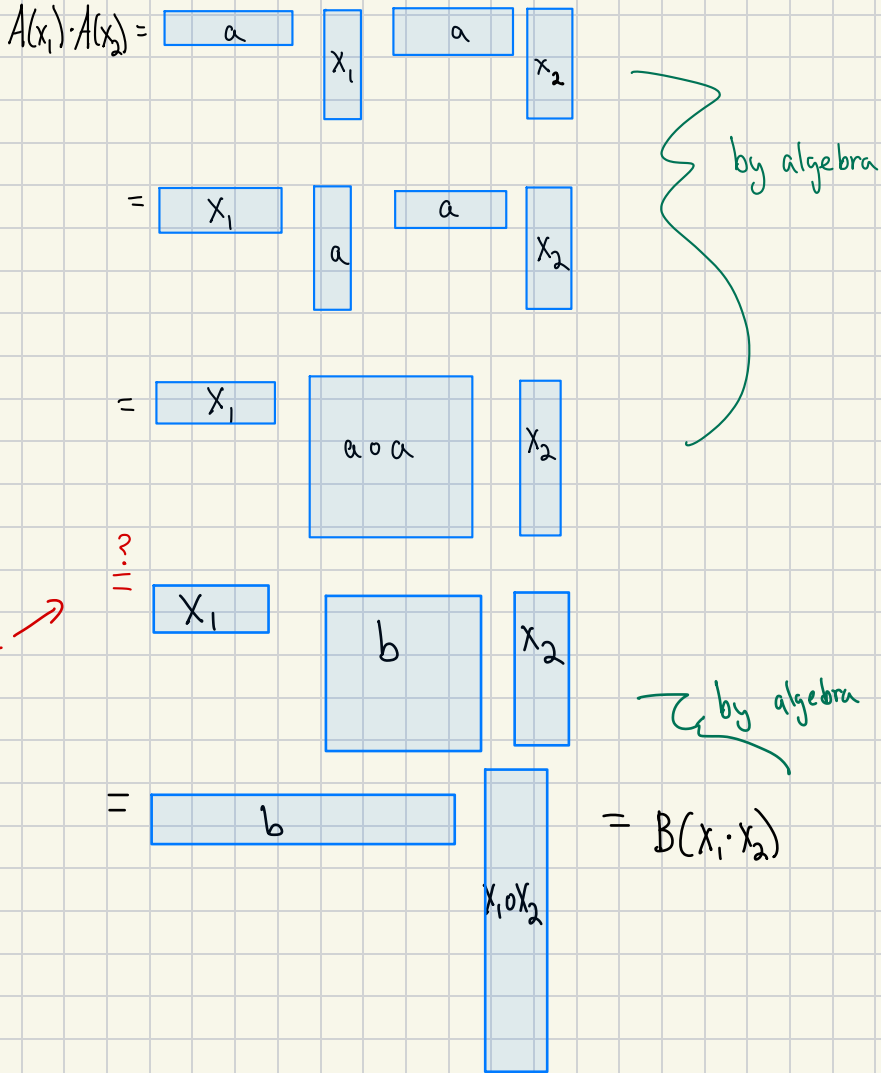
not unif distributed

we use self-corrector since not all queries uniform

# Analysis of outer-product tester:

if $b = a \circ a$, test passes (green equalities hold)

if $b \neq a \circ a$:



if $b = a \circ a$, $\Pr_{x_1, x_2}[X_1 (a \circ a) X_2 = X_1 \, b \, X_2] = 1 \Rightarrow$ test always passes

if $b \neq a \circ a$:

Fact $\Rightarrow \Pr_{x_2}[(a \circ a) \cdot x_2 \neq b \cdot x_2] = 1/2$ (*)

if $(a \circ a) \cdot x_2 \neq b \cdot x_2$ then

Fact $\Rightarrow \Pr_{x_1}[X_1 (a \circ a) X_2 \neq X_1 \, b \, X_2] = 1/2$ (**)

$\Rightarrow \Pr[\text{fail test}] = \Pr[(*) \text{ and } (**) \text{ happen}]$

$\geq 1/4$

So test pass $\Rightarrow$ safe to assume $b = a \circ a$

Similar argument $\Rightarrow$ safe to assume $c = a \circ a \circ a$

## Query Complexity for part 1

# random bits $= O(n^3)$

# queries to proof $= O(1)$

## Conclusion

PCP protocol for 3SAT in which verifier uses:

$O(n^3)$ random bits
+ $O(1)$ queries