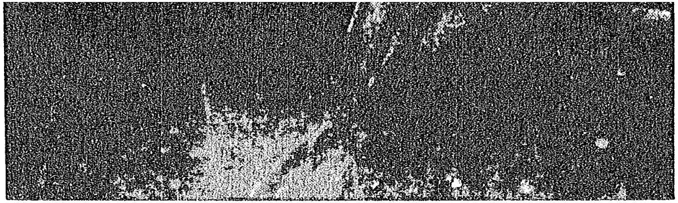
PCP theorem with some error correcting codes



Stephen T. Emlen/Cornell University

Kenyan white-breasted bee eater parent, left, and offspring. Parent bee eaters, usually the males, cajole, manipulate and do anything else in their power to force their young sons to leave their mates and return with their parent to help raise another brood.

New Short Cut Found For Long Math Proofs

A proof can be tested by checking just a

showing something is impossible, can be just as important and open just as many new areas of research as a positive one.

The discovery was made by San-

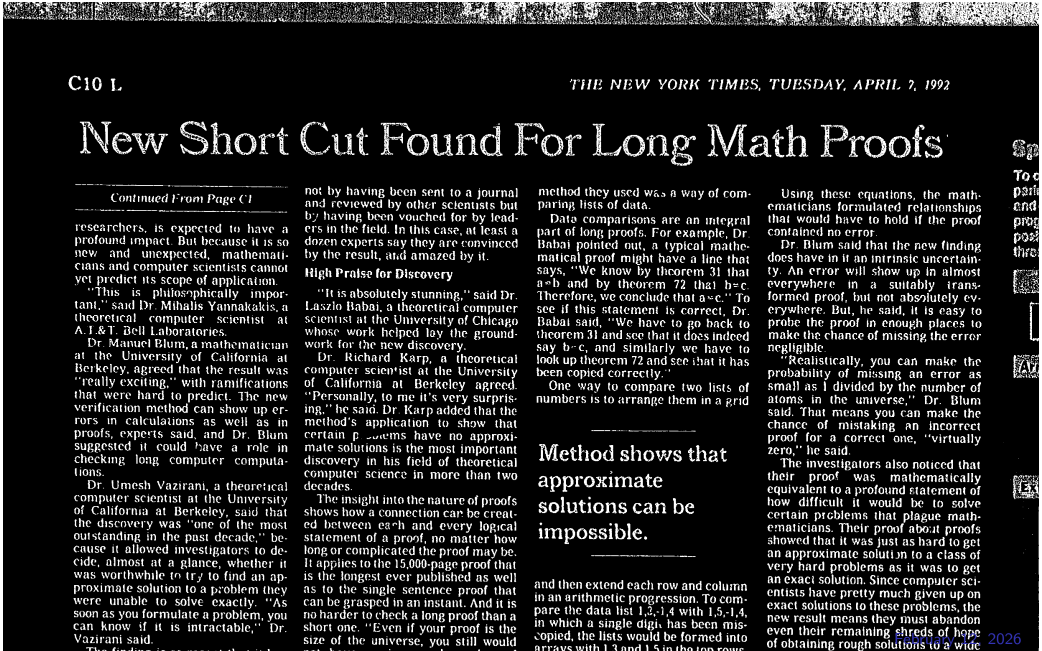
burst of radian

Asteroids of this siz
atmospheric burning
into the atmosphere. T

● Asteroid
impact crate

0.6

Asteroids of this siz



C10 L THE NEW YORK TIMES, TUESDAY, APRIL 7, 1992

New Short Cut Found For Long Math Proofs

Continued From Page C1

researchers, is expected to have a profound impact. But because it is so new and unexpected, mathematicians and computer scientists cannot yet predict its scope of application.

"This is philosophically important," said Dr. Mihalis Yannakakis, a theoretical computer scientist at A.T.&T. Bell Laboratories.

Dr. Manuel Blum, a mathematician at the University of California at Berkeley, agreed that the result was "really exciting," with ramifications that were hard to predict. The new verification method can show up errors in calculations as well as in proofs, experts said, and Dr. Blum suggested it could have a role in checking long computer computations.

Dr. Umesh Vazirani, a theoretical computer scientist at the University of California at Berkeley, said that the discovery was "one of the most outstanding in the past decade," because it allowed investigators to decide, almost at a glance, whether it was worthwhile to try to find an approximate solution to a problem they were unable to solve exactly. "As soon as you formulate a problem, you can know if it is intractable," Dr. Vazirani said.

not by having been sent to a journal and reviewed by other scientists but by having been vouched for by leaders in the field. In this case, at least a dozen experts say they are convinced by the result, and amazed by it.

High Praise for Discovery

"It is absolutely stunning," said Dr. Laszlo Babai, a theoretical computer scientist at the University of Chicago whose work helped lay the groundwork for the new discovery.

Dr. Richard Karp, a theoretical computer scientist at the University of California at Berkeley agreed. "Personally, to me it's very surprising," he said. Dr. Karp added that the method's application to show that certain problems have no approximate solutions is the most important discovery in his field of theoretical computer science in more than two decades.

The insight into the nature of proofs shows how a connection can be created between each and every logical statement of a proof, no matter how long or complicated the proof may be. It applies to the 15,000-page proof that is the longest ever published as well as to the single sentence proof that can be grasped in an instant. And it is no harder to check a long proof than a short one. "Even if your proof is the size of the universe, you still would

method they used was a way of comparing lists of data.

Data comparisons are an integral part of long proofs. For example, Dr. Babai pointed out, a typical mathematical proof might have a line that says, "We know by theorem 31 that a=b and by theorem 72 that b=c. Therefore, we conclude that a=c." To see if this statement is correct, Dr. Babai said, "We have to go back to theorem 31 and see that it does indeed say b=c, and similarly we have to look up theorem 72 and see that it has been copied correctly."

One way to compare two lists of numbers is to arrange them in a grid

Method shows that approximate solutions can be impossible.

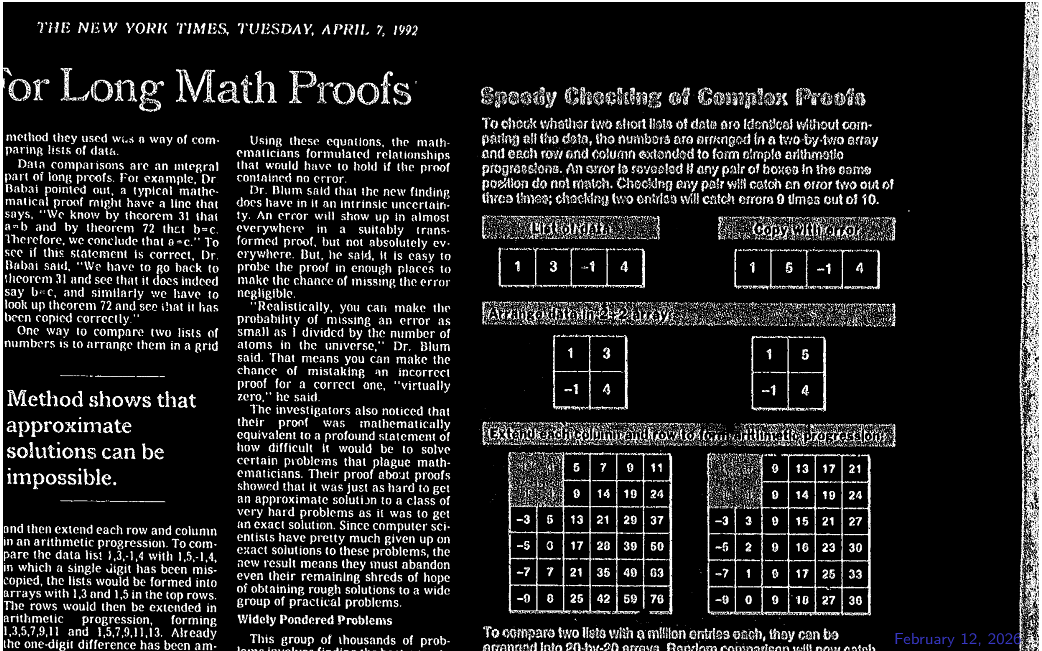
and then extend each row and column in an arithmetic progression. To compare the data list 1,3,-1,4 with 1,5,-1,4, in which a single digit has been miscopied, the lists would be formed into arrays with 1,3 and 1,5 in the top rows

Using these equations, the mathematicians formulated relationships that would have to hold if the proof contained no error.

Dr. Blum said that the new finding does have in it an intrinsic uncertainty. An error will show up in almost everywhere in a suitably transformed proof, but not absolutely everywhere. But, he said, it is easy to probe the proof in enough places to make the chance of missing the error negligible.

"Realistically, you can make the probability of missing an error as small as 1 divided by the number of atoms in the universe," Dr. Blum said. That means you can make the chance of mistaking an incorrect proof for a correct one, "virtually zero," he said.

The investigators also noticed that their proof was mathematically equivalent to a profound statement of how difficult it would be to solve certain problems that plague mathematicians. Their proof about proofs showed that it was just as hard to get an approximate solution to a class of very hard problems as it was to get an exact solution. Since computer scientists have pretty much given up on exact solutions to these problems, the new result means they must abandon even their remaining shreds of hope of obtaining rough solutions to a wide



THE NEW YORK TIMES, TUESDAY, APRIL 7, 1992

or Long Math Proofs

method they used was a way of comparing lists of data.

Data comparisons are an integral part of long proofs. For example, Dr. Babai pointed out, a typical mathematical proof might have a line that says, "We know by theorem 31 that a=b and by theorem 72 that b=c. Therefore, we conclude that a=c." To see if this statement is correct, Dr. Babai said, "We have to go back to theorem 31 and see that it does indeed say b=c, and similarly we have to look up theorem 72 and see that it has been copied correctly."

One way to compare two lists of numbers is to arrange them in a grid

Method shows that approximate solutions can be impossible.

and then extend each row and column in an arithmetic progression. To compare the data list 1,3,-1,4 with 1,5,-1,4, in which a single digit has been miscopied, the lists would be formed into arrays with 1,3 and 1,5 in the top rows. The rows would then be extended in arithmetic progression, forming 1,3,5,7,9,11 and 1,5,7,9,11,13. Already the one-digit difference has been am-

Using these equations, the mathematicians formulated relationships that would have to hold if the proof contained no error.

Dr. Blum said that the new finding does have in it an intrinsic uncertainty. An error will show up in almost everywhere in a suitably transformed proof, but not absolutely everywhere. But, he said, it is easy to probe the proof in enough places to make the chance of missing the error negligible.

"Realistically, you can make the probability of missing an error as small as 1 divided by the number of atoms in the universe," Dr. Blum said. That means you can make the chance of mistaking an incorrect proof for a correct one, "virtually zero," he said.

The investigators also noticed that their proof was mathematically equivalent to a profound statement of how difficult it would be to solve certain problems that plague mathematicians. Their proof about proofs showed that it was just as hard to get an approximate solution to a class of very hard problems as it was to get an exact solution. Since computer scientists have pretty much given up on exact solutions to these problems, the new result means they must abandon even their remaining shreds of hope of obtaining rough solutions to a wide group of practical problems.

Widely Pondered Problems

This group of thousands of prob-

Speedy Checking of Complex Proofs

To check whether two short lists of data are identical without comparing all the data, the numbers are arranged in a two-by-two array and each row and column extended to form simple arithmetic progressions. An error is revealed if any pair of boxes in the same position do not match. Checking any pair will catch an error two out of three times; checking two entries will catch errors 9 times out of 10.

List of data

1	3	–1	4

Copy with error

1	5	–1	4

Arrange data in 2-by-2 array:

1	3
–1	4

1	5
–1	4

Extend each column and row to form arithmetic progression:

1	3	5	7	9	11
–1	4	9	14	19	24
–3	5	13	21	29	37
–5	6	17	28	39	50
–7	7	21	35	49	63
–9	8	25	42	59	76

1	5	9	13	17	21
–1	4	9	14	19	24
–3	3	9	15	21	27
–5	2	9	16	23	30
–7	1	9	17	25	33
–9	0	9	18	27	36

To compare two lists with a million entries each, they can be arranged into 20-by-20 arrays. Random comparison will now catch

A codeword is a codeword is a codeword

We will need to see codewords as “oracles” a.k.a. functions

Definition

A code C over the alphabet $\mathbb{F}_q$ over domain D is a subset $C \subset \mathbb{F}_q^D$

Thus a codeword c is a function

$$\begin{array}{rccc} c: & D & \to & \mathbb{F}_q \\ & x & \mapsto & c(x) \end{array}$$

- write $c(i)$ instead of c_i
- RS code with support $(x_1, \dots, x_n)$: write $c(x_i)$ instead of c_i
- Reed-Muller code: write $c(P)$ instead of c_i
  (where $i = i(P)$ would be given by some indexing map of $\mathbb{F}_q^m$)

But we may still use at some occasion the index notation

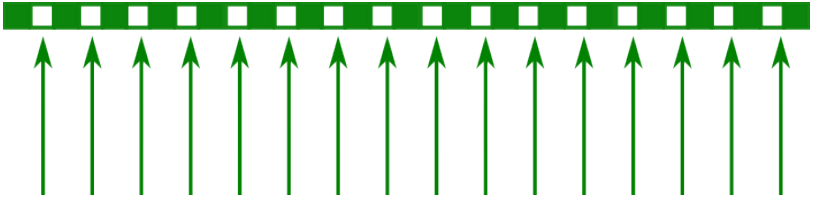
$$c_i, \quad c_{x_i}, \quad c_P$$

NP class

NP

A language $\mathcal{L}$ is in NP if there exists a polynomial time deterministic algorithm $\mathcal{V}$ s.t.

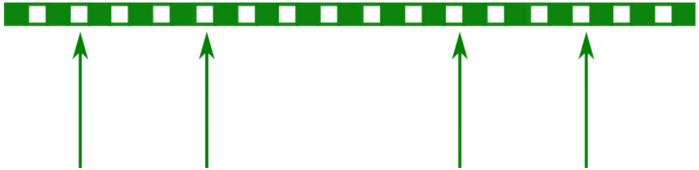
1. **(Completeness)** for every instance $\boldsymbol{x} \in \mathcal{L}$ there exists w such that $\mathcal{V}(\boldsymbol{x}, \boldsymbol{w}) = 1$
2. **(Soundness)** for every instance $\boldsymbol{x} \notin \mathcal{L}$, for every w, $\mathcal{V}(\boldsymbol{x}, \boldsymbol{w}) = 0$



- the language of non prime numbers N. The witness of "N is non prime" is $w = (P, Q)$ such that $N = PQ$
- the langage of valid transactions with respect to the current state of a blockchain

Probabilistically checkable proofs

$\mathcal{V}^{\pi}$ notation for $\mathcal{V}$ with "oracle access" to π



PCP$(r(n), q(n))$ verifier

A PCP$(r(n), q(n))$ verifier for a language $\mathcal{L}$ is PPT oracle-machine $\mathcal{V}(\cdot)^{(\cdot)}$ s.t.

1. **(Completeness)** For every $x \in \mathcal{L}$, there exists a proof π s.t. $\mathbb{P}\left[\mathcal{V}^{\pi}(x) = 1\right] = 1$
2. **(Soundness)** For every $x \notin \mathcal{L}$ and every proof π, $\mathbb{P}\left[\mathcal{V}^{\pi}(x) = 1\right] \leq \frac{1}{2}$
3. **(Efficiency)** On input x and oracle access to π, $\mathcal{V}$ uses
   - $r(n)$ random bits
   - $q(n)$ queries of 1-bit entries
4. *Nota bene* $n = |x|$

PCP class

PCP class

A langage $\mathcal{L}$ is in $\mathrm{PCP}(r(n), q(n))$ if it admits a $\mathrm{PCP}(O(q(n)), O(r(n)))$ verifier

PCP Theorem

$$\mathrm{NP} = \mathrm{PCP}(1, \log n)$$

(Arora-Safra 92, Arora-Lund-Motwani-Sudan-Szegedy 1998)

Workflow

Witness $w \Longrightarrow$ algebraic arithmetization $\Longrightarrow$ Encoding π of witness $w \Longrightarrow$ Test π

CKT-SAT

An algebraic circuit C is a directed acyclic graph with fan-in 2
Some nodes are input nodes
Other nodes are internal nodes, marked with $+$ and $\times$
One node is the output node
The size $n = |C|$ of a circuit is the number of nodes
An assignment $\boldsymbol{w}$ is a map $\boldsymbol{w} : \{\text{input nodes}\} \to \mathbb{F}_2$
$C(\boldsymbol{w})$ is the value of the output node on the assignment $\boldsymbol{w}$
If $C(\boldsymbol{w}) = 1$ then $\boldsymbol{w}$ is a satisying assignment

CKT-SAT

- Input: an algebraic circuit C
- Question: does C has a satisfying assignement

Proposition

CKT-SAT is NP-complete

Simplex (Walsh-Hadamard) code

$$\begin{array}{rcl} H_m : \quad \mathbb{F}_2^m & \to & \mathbb{F}_2^{2^m} \\ \boldsymbol{w} \in \mathbb{F}_2^m & \mapsto & H_m(\boldsymbol{w}) = (\langle \boldsymbol{w}, x \rangle)_{x \in \mathbb{F}_2^m} \end{array}$$

- The image of H_m is the **Walsh-Hadamard code**

$$[2^m, m, 2^{m-1}]_2 \text{ code}$$

Lin-test$_m^{\boldsymbol{c}}$

Oracle $\boldsymbol{c} \in \mathbb{F}_2^{2^m}$

1. Pick random $x, y \in_R \mathbb{F}_2^m$
2. Query $\boldsymbol{c}$ at $x, y, x + y$
3. Receive $\boldsymbol{c}(x), \boldsymbol{c}(y), \boldsymbol{c}(x + y)$
4. Output $\boldsymbol{c}(x + y) \overset{?}{=} \boldsymbol{c}(x) + \boldsymbol{c}(y)$

Blum Luby Rubinfeld lineary test

- randomness $2m$
- query 3
- circuit size 3
- answer size 1

Theorem

1. *If* $\boldsymbol{c} \in H_m$ *then* $\text{Test-lin}_m^{\boldsymbol{c}}$ *accepts with probability 1*
2. *If* $\Delta(\boldsymbol{c}, H_m) > 2\delta$ *then*
$$\mathbb{P}(\text{Lin-test}_m^{\boldsymbol{c}} \ \textit{accepts}) < 1 - \delta$$

Contrapositive

If
$$\mathbb{P}(\text{Lin-test}_m^{\boldsymbol{c}} \ \text{accepts}) \geq 1 - \delta$$
then $\Delta(\boldsymbol{c}, H_m) \leq 2\delta$

"Self correction"

Self-corr$_m^{\boldsymbol{c}}(x)$

Input $x \in \mathbb{F}_2^m$

Oracle $\boldsymbol{c} \in \mathbb{F}_2^{2^m}$

1. $y \in_R \mathbb{F}_2^m$
2. Output $\boldsymbol{c}(x+y) - \boldsymbol{c}(y)$

Theorem

1. *If* $\boldsymbol{c} \in H_m$ *then* Self-corr$_m^{\boldsymbol{c}}(x)$ *output* $\boldsymbol{c}(x)$ *with probability 1*
2. *If* $\Delta(\boldsymbol{c}, \boldsymbol{g}) \leq \delta$ *with* $\boldsymbol{g} \in H_m \subset \mathbb{F}_2^{2^m}$

$$\mathbb{P}(\text{Self-corr}_m^{\boldsymbol{c}}(x) = g(x)) \geq 1 - 2\delta$$

“Quadratic encoding”

$$\begin{array}{rrcl} H_{m\times m}: & \mathbb{F}_2^m & \to & \mathbb{F}_2^{2^{m\times m}} \\ & \boldsymbol{w} \in \mathbb{F}_2^m & \mapsto & \langle \boldsymbol{w} \otimes \boldsymbol{w}, \boldsymbol{q} \rangle_{q\in\mathbb{F}_2^m\times\mathbb{F}_2^m} \end{array}$$

Explicitely, with

$$\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m)$$
$$\boldsymbol{q} = (\boldsymbol{q}_{ij})_{(i,j)\in[m]\times[m]}$$

Then

$$H_{m\times m}(\boldsymbol{w})(\boldsymbol{q}) = \sum_{i=1}^{m}\sum_{j=1}^{m} \boldsymbol{q}_{ij}\boldsymbol{w}_i\boldsymbol{w}_j$$

Coding the witness and the values of the gates

- C has km inputs witness $\boldsymbol{w}$ is written $\boldsymbol{w}_1, \dots, \boldsymbol{w}_k$ with $\boldsymbol{w}_i \in \mathbb{F}_2^m$
- Let $\boldsymbol{z}_{\boldsymbol{w}} \in \mathbb{F}_2^n$ the values of all the gates on input $\boldsymbol{w}$

Verifier $V_2^{O_1,\dots,O_k,\pi_1,\pi_2}(C)$

Input

C has size n

C has km inputs

Oracles

$O_1, \dots, O_k : \mathbb{F}_2^m \to \mathbb{F}_2$ *(input oracles $O_i = H_m(\boldsymbol{w}_i) \in \mathbb{F}_2^{2^m}$)*

$\pi_1 : \mathbb{F}_2^n \to \mathbb{F}_2$ *(auxiliary oracle $\pi_1 = H_n(\boldsymbol{z}_{\boldsymbol{w}}) \in \mathbb{F}_2^{2^n}$)*

$\pi_2 : \mathbb{F}_2^{n\times n} \to \mathbb{F}_2$ *(auxiliary oracle $\pi_2 = H_{n\times n}(\boldsymbol{z}_{\boldsymbol{w}}) \in \mathbb{F}_2^{2^{n\times n}}$)*

Nota bene: V_2 will not read the witness entirely

Quadratic-linear test

Test-quad$^{\pi_1,\pi_2}$

Oracles $\pi_1 \in \mathbb{F}_2^{2^n}$ *and* $\pi_2 \in \mathbb{F}_2^{2^{n\times n}}$

1. $x, y \in_R \mathbb{F}_2^n$
2. Output $\pi_1(x)\pi_1(y) \overset{?}{=} \pi_2(x \otimes y)$

Proposition

- Perfect completeness when there exists $\boldsymbol{z} \in \mathbb{F}_2^n$ s.t. $\pi_1 = H_n(\boldsymbol{z})$ and $\pi_2 = H_{n\times n}(\boldsymbol{z})$
- If

$$\mathbb{P}(\text{Test-quad}^{\pi_1,\pi_2} \text{ accepts}) \geq \frac{3}{4}$$

- then there exists $\boldsymbol{z} \in F_2^n$ such that

$$\pi_1 = H_n(\boldsymbol{z}) \text{ and } \pi_2 = H_{n\times n}(\boldsymbol{z})$$

Quadratic-linear test with self correction

Test-quad$^{\pi_1,\pi_2}$

Oracles $\pi_1 \in \mathbb{F}_2^{2^n}$ *and* $\pi_2 \in \mathbb{F}_2^{2^{n\times n}}$

1. $x, y \in_R \mathbb{F}_2^n$
2. $v = \text{Self-corr}^{\pi_2}(x \otimes y)$ *(self-correction)*
3. Output $\pi_1(x)\pi_1(y) \overset{?}{=} v$

Proposition

- Suppose there exists $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ such that

$$\Delta(H_n(\boldsymbol{z}_1), \pi_1) \leq \delta \text{ and } \Delta(H_{n\times n}(\boldsymbol{z}_2), \pi_2) \leq \delta$$

  and that

$$\mathbb{P}(\text{Test-quad}^{c_1,c_2} \text{ accepts}) \geq \frac{3}{4} + 4\delta$$

- then $\boldsymbol{z}_1 = \boldsymbol{z}_2$

Consistency of π_1 with the oracles $O_1, \ldots, O_m$

- Witness view: $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k, \overline{\boldsymbol{z}}) \in \mathbb{F}_2^n \to \mathbb{F}_2$ and $\boldsymbol{z}_i \in \mathbb{F}_2^m \to \mathbb{F}_2$ then

$$\text{Proj}(\boldsymbol{z}, i) = \boldsymbol{z}_i, \qquad i = 1, \ldots, k$$

- Oracle view $\pi_1 : \mathbb{F}_2^{2^n} \to \mathbb{F}_2$ and $O_1, \ldots, O_k : \mathbb{F}_2^{2^m} \to \mathbb{F}_2$ then

$$\text{Proj}(\pi_1, i) = O_i, \qquad i = 1, \ldots, k$$

Consistency

Given $O_1, \ldots, O_k$ and π_1
Does there exists $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k, \overline{\boldsymbol{z}}) \in \mathbb{F}_2^n \to \mathbb{F}_2$ such that

$$\begin{aligned} \pi_1 &= H_n(\boldsymbol{z}) \\ O_i &= H_m(\boldsymbol{z}_i), \quad i = 1 \ldots k \end{aligned}$$

Test-proj$^{\pi_1, O_i}(i)$

Input i

Oracles $\pi_1 : \mathbb{F}_2^n \to \mathbb{F}_2,\ O_i : \mathbb{F}_2^m \to \mathbb{F}_2$

1. $x_i \in_R \mathbb{F}_2^m$
2. $\overline{x}_i = (0 \dots 0, x_i, 0 \dots 0) \in \mathbb{F}_2^n$
3. $v = \text{Self-corr}^{\pi_1}(\overline{x}_i)$
4. Output $v \overset{?}{=} O_i(x_i)$

Proposition

- *If* $(\Delta(\pi_1, g_1) \le \delta$ *with* $g_1 \in H_n)$ *and* $(\Delta(O_i, g_i) \le \delta$ *with* $g_i \in H_m)$ *and*
$$\mathbb{P}(\text{Test-proj}^{\pi_1, O_i}(i) \textit{ accepts}) \ge \frac{1}{2} + 3\delta$$
- *then* $\text{Proj}(g_1, i) = g_i$

Circuit algebraization

- $|C| = n$, C has $k \times m$ inputs
- To gate j with input gates j_1 and j_2 define $P_j \in \mathbb{F}_2[Z_1, \ldots, Z_n]$
  - for a $+$ gate
    $$P_j = Z_j - (Z_{j_1} + Z_{j_2})$$
  - for a $\times$ gate
    $$P_j = Z_j - (Z_{j_1} \times Z_{j_2})$$
  - for the output gate
    $$P_j = 1 - Z_j$$

C accepts on inputs $x_1, \ldots, x_{km}$ if there exists

$$z = (z_0, \ldots, z_{km-1}, z_{km}, \ldots, z_{n-1})\mathbb{F}_2^n$$

such that

$$z_j = x_j, \quad j = 0, \ldots, km - 1$$
$$P_j(z) = 0, \quad j = km, \ldots, n - 1$$

Circuit test

Naïve test

Accepts if $\pi_2(P_i) = 0$ for $i = k, \dots, n-1$

Test-circuit$^{\pi_2}((P_i)_i)$

Input: P_i, $i = km, \dots, n-1$

Oracle: $\pi_2 \in \mathbb{F}_2^{2^{n\times n}}$

1. $r_i \in_R \mathbb{F}_2$, $i = km, \dots, n-1$
2. $P = \sum r_i P_i$
3. Accepts if Self-corr$(\pi_2, P) = 0$

Numbers

1. If all Test-lin pass with proba $1 - \delta_0$
   then everyone is $\delta = 2\delta_0$-close to the code of its concern H_m, H_n, $H_{n\times n}$
2. Then Test-quad has soundness $\frac{3}{4} + 4\delta$
3. Then Test-proj has soundness $\frac{1}{2} + 3\delta$

Proposition

- *If 1. 2. and 3. above pass with high enough proba and*

$$\mathbb{P}(\text{Test-circuit } \textit{accepts}) \geq \frac{1}{2} + 4\delta$$

then $\pi_1 = H_n((\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k, \overline{\boldsymbol{z}}))$ *where* $\boldsymbol{z}_i = \mathrm{Enc}^{-1}(O_i)$, $i = 1 \ldots, k$ *are such that*

$$C(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k) = 1$$

Use δ_0 which minimizes $s(\delta_0) = \max(1 - \delta_0, \frac{3}{4} + 8\delta_0, \frac{1}{2} + 6\delta_0)$

Verifier $V_2^{O_1,\dots,O_k,\pi_1,\pi_2}(C)$

1. $6k + 12$ queries

	O_i's	π_1	π_2
Test-lin	3k	3	3
Test-proj	k	2k	
Test-quad		2	2
Test-circuit			2

2. randomness $3km + n(k+5) + 4n^2$

	O_i's	π_1	π_2	aux.
Test-lin	$k \cdot 2m$	$2n$	$2n^2$	
Test-proj	$k \cdot m$	$k \cdot n$		
Test-quad		$2n$	n^2	
Test-circuit			n^2	n

3. circuit size 2^{6k+12}
4. answer size 1

3SAT

- A 3CNF is conjunction of *clauses*
- each clause is a disjunction of three *litterals*

$$(A \vee \neg B \vee \neg C) \wedge (\neg D \vee E \vee F)$$
$$(A \vee B) \wedge (C)(A \vee B) \wedge (C)$$

3SAT

- Input: a 3CNF formula ϕ
- Question: does ϕ has a satisfying assignement

3SAT is NP-complete.
We note $n = |\phi|$ the number of variables in ϕ
(which we take as the number of clauses)

Arora-Safra framework (I)

- an assignement is a map $[n] \hookrightarrow \{0,1\}$
  Write $n = h^m$ for some h and m
- an assignement is a map $[h]^m \hookrightarrow \{0,1\}$
  Consider $H \subset \mathbb{F}_q$, with $|H| = h$
- an assignement is a map $H^m \hookrightarrow \{0,1\} \subset \{0,1\}$

$$([n] \to \{0,1\}) \hookrightarrow ([h]^m \to \{0,1\}) \hookrightarrow (H^m \to \{0,1\})$$

Arora-Safra framework (II)

$$(H^m \to \{0,1\}) \hookrightarrow \bigotimes_{i=1}^{m} \mathrm{RS}[\mathbb{F}_q, h]$$

- Let C_m be

$$C_m = \bigotimes_{i=1}^{m} \mathrm{RS}[\mathbb{F}_q, h]$$

  C_m is a $[q^m, |H|^m]_q$ code
- H^m is an information set of C_m
- To an assignement $H^m \to \{0,1\}$ associate

$$A = A(X_1, \ldots, X_m) \in \mathbb{F}_1[X_1, \ldots, X_m]_{h-1}$$

  and a codeword

$$c_A = (A(P))_{P \in \mathbb{F}_q^m}$$

Verifier $V_1^{\pi_1,\pi_2}(\phi)$

Input

ϕ has n clauses and n variables

Oracles

$\pi_1 : \mathbb{F}_q^m \to \mathbb{F}_q$ *a codeword* $c_A \in \bigotimes_{i=1}^m RS[\mathbb{F}_q, h]$

$\pi_2 : [m] \times \mathbb{F}_q^{m-1} \to \mathbb{F}_q[Z]_h$ *(auxiliary oracle* π_2*)*

1. Test-degree$^{\pi_1,\pi_2}$
2. Test-assign$^{\pi_1}(\phi)$

Test-degree$^{\pi_1,\pi_2}()$

Test-degree$^{\pi_1,\pi_2}()$

Input

ϕ has n clauses and n variables

Oracles

$\pi_1 : \mathbb{F}_q^m \to \mathbb{F}_q$ *a codeword* $c_A \in \bigotimes_{i=1}^m RS[\mathbb{F}_q, h]$

$\pi_2 : [m] \times \mathbb{F}_q^{m-1} \to \mathbb{F}_q[Z]_h$ *(auxiliary fancy oracle* π_2*)*

Completeness situation

$\pi_1 = c_A$ *is a codeword with an associated polynomial* $A \in \mathbb{F}[X_1, \ldots, X_m]_h$

$\pi_2(i, (a_1, \ldots, \hat{a}_i, \ldots, a_{m-1}) = A(a_1, \ldots, Z, \ldots, a_m)$

1. Do $\dfrac{4}{\delta}$ consistency checks between π_1 and π_2
2. Do $\dfrac{8m}{\delta}$ consistency checks of π_2 with itself

Arithmetization of formula ϕ

- Function $\chi_1 : H^m \times H^m \to \{0,1\}$

$$\chi_1(i,j) = 1 \text{ if and only if } V_j \text{ appears as first litteral in the } i\text{-th clause}$$

- Function $\iota_1 : H^m \to \{0,1\}$

$$\iota_i(i) = 1 \text{ if and only if the first litteral is not negated in the } i\text{-th clause}$$

- Similar functions χ_2, ι_2 and χ_3, ι_3 for the second and third litteral

Proposition

To polynomial $A \in \mathbb{F}_q[X_1, \dots, X_m]$ associate

$$A(V_1) = A(V_{11}, \dots, V_{1m}), \quad A(V_2) = A(V_{21}, \dots, V_{3m}), \quad A(V_3) = A(V_{31}, \dots, V_{3m})$$

Then A is a valid assignement polynomial at clause i if and only

$$0 = (\chi_1(i,j)(\iota_1(i)) - A(V_1)))$$

Arithmetization of formula ϕ

- Function $\chi_1 : H^m \times H^m \to \{0,1\}$

$$\chi_1(i,j) = 1 \text{ if and only if } V_j \text{ appears as first litteral in the } i\text{-th clause}$$

- Function $\iota_1 : H^m \to \{0,1\}$

$$\iota_i(i) = 1 \text{ if and only if the first litteral is not negated in the } i\text{-th clause}$$

- Similar functions χ_2, ι_2 and χ_3, ι_3 for the second and third litteral

Proposition

To polynomial $A \in \mathbb{F}_q[X_1, \ldots, X_m]$ associate

$$A(V_1) = A(V_{11}, \ldots, V_{1m}), \quad A(V_2) = A(V_{21}, \ldots, V_{3m}), \quad A(V_3) = A(V_{31}, \ldots, V_{3m})$$

Then A is a valid assignement polynomial at clause i if and only

$$0 = (\chi_1(i,j)(\iota_1(i)) - A(V_1))) \times (\chi_2(i,j)(\iota_2(i)) - A(V_2)))$$

Arithmetization of formula ϕ

- Function $\chi_1 : H^m \times H^m \to \{0,1\}$

$$\chi_1(i,j) = 1 \text{ if and only if } V_j \text{ appears as first litteral in the } i\text{-th clause}$$

- Function $\iota_1 : H^m \to \{0,1\}$

$$\iota_i(i) = 1 \text{ if and only if the first litteral is not negated in the } i\text{-th clause}$$

- Similar functions χ_2, ι_2 and χ_3, ι_3 for the second and third litteral

Proposition

To polynomial $A \in \mathbb{F}_q[X_1, \ldots, X_m]$ associate

$$A(V_1) = A(V_{11}, \ldots, V_{1m}), \quad A(V_2) = A(V_{21}, \ldots, V_{3m}), \quad A(V_3) = A(V_{31}, \ldots, V_{3m})$$

Then A is a valid assignement polynomial at clause i if and only

$$0 = (\chi_1(i,j)(\iota_1(i)) - A(V_1))) \times (\chi_2(i,j)(\iota_2(i)) - A(V_2))) \times (\chi_3(i,j)(\iota_3(i)) - A(V_3)))$$

Polynomial view

- To the function $\chi_1, \chi_2, \chi_3 \in H^m \times H^m \to \{0,1\}$ associate polynomials

$$P_{\chi_1} \in \mathbb{F}_q[C_1, \dots, C_m, V_1]$$
$$P_{\chi_2} \in \mathbb{F}_q[C_1, \dots, C_m, V_2]$$
$$P_{\chi_3} \in \mathbb{F}_q[C_1, \dots, C_m, V_3]$$

  and to $\iota_1, \iota_2, \iota_3 \in \times H^m \to \{0,1\}$ associate

$$P_{\iota_1} \in \mathbb{F}_q[C_1, \dots, C_m],\ P_{\iota_2} \in \mathbb{F}_q[C_1, \dots, C_m],\ P_{\iota_3} \in \mathbb{F}_q[C_1, \dots, C_m]$$

- A is valid assignment polynomial if the polynomial BIGPOL

$$P_{\chi_1}(C, V_1)(P_{\iota_1}(C) - A(V_1)) \times P_{\chi_2}(C, V_2)(P_{\iota_2}(C) - A(V_2)) \times P_{\chi_3}(C, V_3)(P_{\iota_3}(C)) - A(V_3))$$

  vanishes at $H^m \times H^m \times H^m \times H^m$

Test-assign$^{\pi_1}(\phi)$

Input

ϕ has n clauses and n variables

Oracles

$\pi_1 : \mathbb{F}_q^m \to \mathbb{F}_q$ *a codeword* $c_A \in \bigotimes_{i=1}^m RS[\mathbb{F}_q, h]$

1. Build $P\chi_1, P_{\chi_2}, P_{\chi_3}$ and $P_{\iota_1}, P_{\iota_2}, P_{\iota_3}$

   $\Longrightarrow$ Virtual oracle access to BIGPOL from

2. Sum-check$^{\text{BIGPOL}}$ *(Reduces "vanishes at all H^m to a test at a single point $\in \mathbb{F}_q^m$)*

Parameters juggling

- Query
  1. Test-degree has $O_\delta(m)$ queries
  2. Sum-check has $4m + 3$ queries
- Randomness
  1. Test-degree has randomness $O_\delta(m^2 \log q)$
  2. Sum-check has randomness $4m \log q$

Choosing h and m:

- With $|\phi| = n = h^m$, and $q = \text{poly}(h) = h^\kappa$, $\kappa > 1$:
- $\log n = m \log h$ and $\log q = \kappa \log h$

  $\implies \log q = O(\kappa) \log n$
- $m = \log n / logh = O(\log n)$

V_1 has randomness $O(\log n^3)$ and query complexity $O(\log n)$

More parameters of a verifier of type V_1

A $(r_1(n), q_1(n), c_1(n), a_1(n))$-verifier is a PPT $V^{\pi_1}(x)$ such that:

1. uses $r_1(n)$ random bits: $r \in_R \{0,1\}^{r_1(n)}$
2. computes a circuit $C_1(x, r)$ of size $c_1(n)$
3. computes queries $l_i = l_i(x, r)$, $i \in [q_1(n)]$
4. gets $\pi_1(l_1), \ldots, \pi_1(l_q) \in \mathbb{F}_2^{a_1(n)}$
5. accepts if $C_1(\pi(l_1), \ldots, \pi_1(l_q)) = 1$

Proposition

$$RPCP(r_1(n),\ q_1(n),\ c_1(n),\ a_1(n)) \subset PCP(r_1(n),\ q_1(n)a_1(n))$$

V_1 is a

$$r_1(n) = \log^3(n),\ q_1(n) = O(\log n),\ c_1(n) = O(\log^2 n),\ a_1(n) = O(\log^2 n)$$

verifier

More parameters of a verifier of type V_1

A $(r_1(n), q_1(n), c_1(n), a_1(n))$-verifier is a PPT $V^{\pi_1}(x)$ such that:

1. uses $r_1(n)$ random bits: $r \in_R \{0,1\}^{r_1(n)}$
2. computes a circuit $C_1(x,r)$ of size $c_1(n)$
3. computes queries $l_i = l_i(x,r)$, $i \in [q_1(n)]$
4. gets $\pi_1(l_1), \ldots, \pi_1(l_q) \in \mathbb{F}_2^{a_1(n)}$
5. accepts if $C_1(\pi(l_1), \ldots, \pi_1(l_q)) = 1$

Proposition (Arora-Safra 98)

$$RPCP(r_1(n),\ q_1(n),\ c_1(n),\ a_1(n)) \subset PCP(r_1(n),\ q_1(n)a_1(n))$$

V_1 is a

$$r_1(n) = O(\log(n)),\ q_1(n) = O(\log^{e_1} n),\ c_1(n) = O(\log^{e_1} n),\ a_1(n) = O(\log^2 n)$$

verifier

More parameters of CKT-SAT Verifier

A $(k, r(n), q(n), c(n), a(n))$ CKT-SAT verifier $(V, \mathrm{Enc}, \mathrm{Enc}^{-1})$
inputs a circuit C with km entries
queries oracles $O_1, \dots, O_k, \pi$

1. generates $r(n)$ random bits
2. builds a circuit C_0 of size $c(n)$
3. generates queries $l_1, \dots, l_q$, where $q = q(n)$
4. receives answers $a_1, \dots, a_q \in \{0, 1\}^{a(n)}$
5. accepts if $C_0(a_1, \dots, a_q) = 1$

Definition (Soundness s)

If for some $O_1, \dots, O_k, \pi$

$$\mathbb{P}(V^{1 O_1, \dots, O_k, \pi}(C) = 1) \geq s$$

then $\mathrm{Enc}^{-1}(O_1), \dots, \mathrm{Enc}^{-1}(O_k)$ is a satisying assignment to C

Verifier $V_2^{O_1,\ldots,O_k,\pi_1,\pi_2}(C)$ is an CKT-SAT verifier

Input

C has size n, with C has km inputs

Oracles

$O_1,\ldots,O_k : \mathbb{F}_2^m \to \mathbb{F}_2$ *(input oracles $O_i = H_m(\boldsymbol{w}_i) \in \mathbb{F}_2^{2^m}$)*

$\pi_1 : \mathbb{F}_2^n \to \mathbb{F}_2$ *(0iliary oracle $\pi_1 = H_n(\boldsymbol{z}_{\boldsymbol{w}}) \in \mathbb{F}_2^{2^n}$)*

$\pi_2 : \mathbb{F}_2^{n\times n} \to \mathbb{F}_2$ *(auxiliary oracle $\pi_2 = H_{n\times n}(\boldsymbol{z}_{\boldsymbol{w}}) \in \mathbb{F}_2^{2^{n\times n}}$)*

V_2 with $(H_m, H_n, H_{n\times n})$ is a

$$r(n) = 4n^2,\ q(n) = 6k+12,\ c(n) = 2^{6k+12},\ a(n) = 1$$

CKT-SAT verifier

Two verifiers

$V_1^{\pi_1}(x)$ a verifier

1. $r_1(n)$
2. $q_1(n)$
3. $c_1(n)$
4. $a_1(n)$

Two verifiers

$V_1^{\pi_1}(x)$ a verifier

1. $r_1(n)$
2. $q_1(n)$
3. $c_1(n)$
4. $a_1(n)$

$V_2^{O_1,\ldots,O_k,\pi_2}(C)$ an CKT-SAT verifier

Circuit C has km inputs

1. $r_2(n)$
2. $q_2(n)$
3. $c_2(n)$
4. $a_2(n)$

Two verifiers

$V_1^{\pi_1}(x)$ a *outer* verifier

1. $r_1(n)$
2. $q_1(n)$
3. $c_1(n)$
4. $a_1(n)$

$V_2^{O_1,\ldots,O_k,\pi_2}(C)$ is a *inner* verifier

Circuit C has km inputs

1. $r_2(n)$
2. $q_2(n)$
3. $c_2(n)$
4. $a_2(n)$

Two verifiers

$V_1^{\pi_1}(x)$ a *outer* verifier

1. $r_1(n)$
2. $q_1(n)$
3. $c_1(n)$ size of circuit C_1 with $q_1(n)$ inputs
4. $a_1(n)$ size of the inputs of C_1

$V_2^{O_1,\ldots,O_{q_1(n)},\pi_2}(C_1)$ is a *inner* verifier

Circuit C_1 has $q_1(n) \times a_1(n)$ inputs

1. $r_2(c_1(n))$
2. $q_2(c_1(n))$
3. $c_2(c_1(n))$
4. $a_2(c_1(n))$

Composed oracle for input $x \in L$

There exists π_1 s.t. $V_1^{\pi_1}(x)$ accepts. For each $r_1 \in \{0,1\}^{r_1(n)}$:

1. generate V_1 queries: $i_1(r_1), \ldots, i_{q_1}(r_1)$
2. get entries from π_1: $a_1(r_1), \ldots, a_{q_1}(r_1) \in \mathbb{F}_2^{a_1(n)}$
3. build circuit $C_1(r_1)$
4. determine oracles for V_2:
    - encode: $c_1(r_1) = \mathrm{Enc}(a_1(r_1)), \ldots, c_{q_1}(r_1) = \mathrm{Enc}(a_{q_1}(r_1))$
    - determine $\pi_2(r_1)$ such that

$$V_2^{c_1(r_1),\ldots,c_{q_1}(r_1),\pi_2(r_1)}(C_1(r_1)) \text{ accepts}$$

Composed proof is $\pi'(\pi'_1, \pi'_2)$ with $\pi'_1 = \pi_1$ and

$$\pi'_2 = (c_1(r_1), \ldots, c_{q_1}(r_1), \pi_2(r_1))_{r_1 \in \{0,1\}^{r_1(n)}}$$

Composed verifier $V^{\pi'}(x)$ has parameters

$$r(n) = r_1(n) + r_2(c_1(n)), \quad q(n) = q_2(c_1(n)), \quad c(n) = c_2(c_1(n)), \quad a(n) = a_2(c_1(n))$$

Towards crypto,introduce a Prover, who maybe dishonnest

Kilian 92

1. Prover instantiates the oracle π with commitments
   For $|\pi| = N$, then the root of Merkle of all π_i, $i \in [N]$ is published by the Prover
2. Verifier invokes $V^{\pi}(x)$
3. Verifier gGenerates randomness r
4. Verifier asks for "opening" π at indices $i_1, \ldots, i_q$
5. Prover reveals $\pi_1 \ldots, \pi_q$ together with their Merkle proof
6. Verifer executes circuit $C(r)$ on inputs $\pi_1 \ldots, \pi_q$

Security holds only on collision resistance

Micali 95

Non interactive proof (argument) using a Fiat-Shamir like transformation

Very subtle (*Building Cryptographic Proofs from Hash Functions*. Chiesa and Yogev, 440 p.)

2025, 2026

- *Proving the PCP Theorem with 1.5 proof compositions (or yet another PCP construction)*, Oded Golreich, ECCC 2026
- *Ideals, Gröbner Bases, and PCPs*, Madhu Sudan, ECCC2025

PCP not efficient in practice $\Longrightarrow$ Interactive Oracle Proofs (IOPs)
Polynomial commitments seems a good abstraction
Sumcheck seems to be very relevant (next lecture, along with another arithmetization and *Spartan*)