Communication Complexity

Jake Kinsella and Max von Hippel

Northeastern University

April 10, 2021

Communication Complexity

If Alice knows x, and Bob knows y, how many bits of information must they communicate, in order for both Alice and Bob to know $f(x, y)$?

Introduction (Max)

Consider a two-party communication problem, in which the participants

(a) Alice

and

(b) Bob

participate to compute a function:

$$f : \underbrace{\mathbb{B}^n}_{\text{Alice's input}} \times \underbrace{\mathbb{B}^n}_{\text{Bob's input}} \to \underbrace{\mathbb{B}}_{\text{global output}}$$

Introduction (Max)

The players can come up with a *protocol* $\Pi = (p_1, \ldots, p_t)$, namely, for some natural $t \in \mathbb{N}$, a sequence of t-many functions $p_i : \mathbb{B}^* \to \mathbb{B}^*$ such that the communication between the players looks like this ...

Introduction (Max)

Alice is given input x.

Introduction (Max)

Alice is given input x.

Hello Bob. I can't reveal x, but $p_1(x)$ is `p1`.

Introduction (Max)

Alice is given input x.

> Hello Bob. I can't reveal x, but $p_1(x)$ is `p1`.

Bob is given input y.

Introduction (Max)

Alice is given input x.

Hello Bob. I can't reveal x, but $p_1(x)$ is $\mathtt{p1}$.

Bob is given input y.

Thanks Alice. I can't reveal y, but $p_2(y, \mathtt{p1})$ is $\mathtt{p2}$.

Introduction (Max)

Alice is given input x.

Hello Bob. I can't reveal x, but $p_1(x)$ is $\mathtt{p1}$.

Bob is given input y.

Thanks Alice. I can't reveal y, but $p_2(y, \mathtt{p1})$ is $\mathtt{p2}$.

... yada yada yada ...

Introduction (Max)

Alice is given input x.

Hello Bob. I can't reveal x, but $p_1(x)$ is $\mathtt{p1}$.

Bob is given input y.

Thanks Alice. I can't reveal y, but $p_2(y, \mathtt{p1})$ is $\mathtt{p2}$.

... yada yada yada ...

Pleasure doing business with you Bob. My final clue for you is that $p_{n-1}(x, \mathtt{p1}, \ldots, \mathtt{pn{-}2})$ is $\mathtt{pn{-}1}$.

Introduction (Max)

Alice is given input x.

Hello Bob. I can't reveal x, but $p_1(x)$ is $\texttt{p1}$.

Bob is given input y.

Thanks Alice. I can't reveal y, but $p_2(y, \texttt{p1})$ is $\texttt{p2}$.

... yada yada yada ...

Pleasure doing business with you Bob. My final clue for you is that $p_{n-1}(x, \texttt{p1}, ..., \texttt{pn-2})$ is $\texttt{pn-1}$.

Rad. Then $p_n(y, \texttt{p1}, ..., \texttt{pn-1})$ is $\texttt{pn}$. TTFN!

Introduction (Max)

- The functions p_i can be *anything* so long as they are well-defined. E.g., could solve the Halting Problem.
- After the final message, *both parties* must know $f(x, y)$.

Introduction (Max)

- The functions p_i can be *anything* so long as they are well-defined. E.g., could solve the Halting Problem.
- After the final message, *both parties* must know $f(x, y)$.

Definition (Communication Complexity)

Suppose Π is a protocol for f in which at most t bits are communicated between Alice and Bob. Then the *communication complexity* of Π, denoted $C(\Pi)$, is t.

Introduction (Max)

- The functions p_i can be *anything* so long as they are well-defined. E.g., could solve the Halting Problem.
- After the final message, *both parties* must know $f(x, y)$.

Definition (Communication Complexity)

Suppose Π is a protocol for f in which at most t bits are communicated between Alice and Bob. Then the *communication complexity* of Π, denoted $C(\Pi)$, is t.

Definition ($C(f)$)

The communication complexity of f, denoted $C(f)$, is the minimum communication complexity achieved by any protocol for f.

Parity (Jake)

Example (Are the number of 1s in xy even (0), or odd (1)?)

$f : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $(x, y) \mapsto \bigoplus xy$.

Parity (Jake)

Example (Are the number of 1s in xy even (0), or odd (1)?)

$f : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $(x, y) \mapsto \bigoplus xy$.

Example protocol Π:

$\texttt{P1} = \text{parity}(x)$.

Parity (Jake)

Example (Are the number of 1s in xy even (0), or odd (1)?)

$f : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $(x, y) \mapsto \bigoplus xy$.

Example protocol Π:

$\text{P1} = \text{parity}(x)$.

$\text{P2} = \text{parity}(y) \bigoplus \text{P1}$

Parity (Jake)

Example (Are the number of 1s in xy even (0), or odd (1)?)

$f : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $(x, y) \mapsto \bigoplus xy$.

Example protocol Π:

$\mathtt{P1} = \text{parity}(x)$.

$\mathtt{P2} = \text{parity}(y) \bigoplus \mathtt{P1}$

Now both Alice and Bob know $f(x, y) = \mathtt{P2}$. $C(f) \leq 2$ because $C(\Pi) = 2$ and Π implements f. But $C(f) \geq 2$ because f depends on x and y. Hence $C(f) = 2$.

Halting (Jake)

Example (A_{TM})

$H : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $\langle M, 1^n \rangle \mapsto 1$ if M halts on 1^n else 0.

Halting (Jake)

Example (A_{TM})

$H : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $\langle M, 1^n \rangle \mapsto 1$ if M halts on 1^n else 0.

Example protocol Π:

P1 = 1^n.

Halting (Jake)

Example (A_{TM})

$H : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $\langle M, 1^n \rangle \mapsto 1$ if M halts on 1^n else 0.

Example protocol Π:

P1 $= 1^n$.

P2$=$ (M does/doesn't accept 1^n).

Halting (Jake)

Example (A_{TM})

$H : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $\langle M, 1^n \rangle \mapsto 1$ if M halts on 1^n else 0.

Example protocol Π:

P1 $= 1^n$.

P2$=$ (M does/doesn't accept 1^n).

Both players have unlimited computation power. We are only interest in communication complexity.

Methods (Max)

If we find a protocol Π, then we know $C(f)$ is at most $C(\Pi)$.

Methods (Max)

If we find a protocol Π, then we know $C(f)$ is at most $C(\Pi)$. What if we don't know any protocol Π?

Methods (Max)

If we find a protocol Π, then we know $C(f)$ is at most $C(\Pi)$.
What if we don't know any protocol Π?

- Could we upper-bound $C(f)$ without knowing Π?

Methods (Max)

If we find a protocol Π, then we know $C(f)$ is at most $C(\Pi)$.
What if we don't know any protocol Π?

- Could we upper-bound $C(f)$ without knowing Π?

What if the only protocols we find seem really lousy?

Methods (Max)

If we find a protocol Π, then we know $C(f)$ is at most $C(\Pi)$.
What if we don't know any protocol Π?

- Could we upper-bound $C(f)$ without knowing Π?

What if the only protocols we find seem really lousy?

- Could we lower-bound $C(f)$ without finding a better protocol?

Methods (Max)

If we find a protocol Π, then we know $C(f)$ is at most $C(\Pi)$.
What if we don't know any protocol Π?

- Could we upper-bound $C(f)$ without knowing Π?

What if the only protocols we find seem really lousy?

- Could we lower-bound $C(f)$ without finding a better protocol?

TL;DR: yup.

Fooling Set Method (Jake)

Consider a two-party protocol for determining whether two inputs are equal:

Fooling Set Method (Jake)

Consider a two-party protocol for determining whether two inputs are equal:

Example (*Equality*)

$EQ : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $\langle x, y \rangle \mapsto 1$ if $x = y$ else 0.

Fooling Set Method (Jake)

We begin with a motivating observation.

Fooling Set Method (Jake)

We begin with a motivating observation.

Lemma (Communication Equality is Image Equality)

If Alice and Bob exchange the same sequence of messages when Alice gets x *and Bob gets* y *as they do when Alice gets* x' *and Bob gets* y'*, then* $f(x, y) = f(x', y')$*.*

Fooling Set Method (Jake)

We begin with a motivating observation.

Lemma (Communication Equality is Image Equality)

If Alice and Bob exchange the same sequence of messages when Alice gets x *and Bob gets* y *as they do when Alice gets* x' *and Bob gets* y'*, then* $f(\mathrm{x}, \mathrm{y}) = f(\mathrm{x}', \mathrm{y}')$.

Proof.

Π is deterministic and f is a function. □

Fooling Set Method (Jake)

We begin with a motivating observation.

Lemma (Communication Equality is Image Equality)

If Alice and Bob exchange the same sequence of messages when Alice gets x *and Bob gets* y *as they do when Alice gets* x' *and Bob gets* y'*, then* $f(\mathrm{x}, \mathrm{y}) = f(\mathrm{x}', \mathrm{y}')$*.*

Proof.

Π is deterministic and f is a function. □

Idea: an efficient protocol will efficiently group together inputs that go to the same output.

Fooling Set Method (Jake)

Idea: an efficient protocol will efficiently **group together inputs that go to the same output.**

Definition (Fooling Set)

If $f : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is a function, a *fooling set* for f is a set $S \subseteq \mathbb{B}^n \times \mathbb{B}^n$ such that for some choice $b \in \mathbb{B}$ $f(S) = \{b\}$ but, for all distinct $(x,y),(x',y') \in S$, $(\neg b) \in f(\{x,x'\} \times \{y,y'\})$.

Basically, a fooling set is a group of inputs that go to the same output, but which is *brittle* to argument-swapping. In some sense these *brittle* sets lower-bound the difficulty in grouping like inputs.

Fooling Set Method (Jake)

Idea: an efficient protocol will efficiently **group together inputs that go to the same output.**

Definition (Fooling Set)

If $f : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is a function, a *fooling set* for f is a set $S \subseteq \mathbb{B}^n \times \mathbb{B}^n$ such that for some choice $b \in \mathbb{B}$ $f(S) = \{b\}$ but, for all distinct $(x, y), (x', y') \in S$, $(\neg b) \in f(\{x, x'\} \times \{y, y'\})$.

Basically, a fooling set is a group of inputs that go to the same output, but which is *brittle* to argument-swapping. In some sense these *brittle* sets lower-bound the difficulty in grouping like inputs.

Lemma (Fooling Set Method)

If f has a size-M fooling set, then $C(f) \geq \log_2(M)$.

Fooling Set Method (Jake)

Example (Set-Disjointness)

$\textsc{Disj} : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is the function that maps (A, B) to 1 if $A \cap B = \varnothing$ else 0.

How many fooling sets does $\textsc{Disj}$ have?

Fooling Set Method (Jake)

Example (Set-Disjointness)

$\textsc{Disj} : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is the function that maps (A, B) to 1 if $A \cap B = \varnothing$ else 0.

How many fooling sets does $\textsc{Disj}$ have? Notice $A \cap B = \varnothing$ if $B = \overline{A}$.

Fooling Set Method (Jake)

Example (Set-Disjointness)

$\textsc{Disj} : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is the function that maps (A, B) to 1 if $A \cap B = \varnothing$ else 0.

How many fooling sets does $\textsc{Disj}$ have? Notice $A \cap B = \varnothing$ if $B = \overline{A}$. There are 2^n possible values A.

Fooling Set Method (Jake)

Example (Set-Disjointness)

$\textsc{Disj} : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is the function that maps (A, B) to 1 if $A \cap B = \varnothing$ else 0.

How many fooling sets does $\textsc{Disj}$ have? Notice $A \cap B = \varnothing$ if $B = \overline{A}$. There are 2^n possible values A. None of these distinct $(A, \overline{A}), (A', \overline{A'})$ satisfy $A \cap \overline{A} = A \cap \overline{A'}$ or $A \cap \overline{A} = A' \cap \overline{A}$ else they wouldn't be distinct.

Fooling Set Method (Jake)

Example (Set-Disjointness)

$\textsc{Disj} : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is the function that maps (A, B) to 1 if $A \cap B = \varnothing$ else 0.

How many fooling sets does $\textsc{Disj}$ have? Notice $A \cap B = \varnothing$ if $B = \overline{A}$. There are 2^n possible values A. None of these distinct $(A, \overline{A}), (A', \overline{A'})$ satisfy $A \cap \overline{A} = A \cap \overline{A'}$ or $A \cap \overline{A} = A' \cap \overline{A}$ else they wouldn't be distinct. So we get a 2^n-size fooling set.

Fooling Set Method (Jake)

Example (Set-Disjointness)

$\textsc{Disj} : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is the function that maps (A, B) to 1 if $A \cap B = \emptyset$ else 0.

How many fooling sets does $\textsc{Disj}$ have? Notice $A \cap B = \emptyset$ if $B = \overline{A}$. There are 2^n possible values A. None of these distinct $(A, \overline{A}), (A', \overline{A'})$ satisfy $A \cap \overline{A} = A \cap \overline{A'}$ or $A \cap \overline{A} = A' \cap \overline{A}$ else they wouldn't be distinct. So we get a 2^n-size fooling set.

$$\therefore C(\textsc{Disj}) \geq \log_2(2^n) = n$$

Fooling Set Method (Jake)

NTS: If f has a size-M fooling set then $C(f) \geq \log_2(M)$.

Proof.

For a contradiction suppose a protocol Π exists for f s.t. $C(\Pi) < \log_2(M)$.

Fooling Set Method (Jake)

NTS: If f has a size-M fooling set then $C(f) \geq \log_2(M)$.

Proof.

For a contradiction suppose a protocol Π exists for f s.t. $C(\Pi) < \log_2(M)$. Then Π yields at most $2^{C(\Pi)} < 2^{\log_2(M)} = M$ distinct communication patterns.

Fooling Set Method (Jake)

NTS: If f has a size-M fooling set then $C(f) \geq \log_2(M)$.

Proof.

For a contradiction suppose a protocol Π exists for f s.t. $C(\Pi) < \log_2(M)$. Then Π yields at most $2^{C(\Pi)} < 2^{\log_2(M)} = M$ distinct communication patterns. But there are $M * (M-1)$ disjoint choices of $(x, y), (x', y') \in S$.

Fooling Set Method (Jake)

NTS: If f has a size-M fooling set then $C(f) \geq \log_2(M)$.

Proof.

For a contradiction suppose a protocol Π exists for f s.t. $C(\Pi) < \log_2(M)$. Then Π yields at most $2^{C(\Pi)} < 2^{\log_2(M)} = M$ distinct communication patterns. But there are $M * (M-1)$ disjoint choices of $(x, y), (x', y') \in S$. Since $M * (M-1) > M$ there must be some $(x, y), (x', y')$ on which Π yields the same communication pattern.

Fooling Set Method (Jake)

NTS: If f has a size-M fooling set then $C(f) \geq \log_2(M)$.

Proof.

For a contradiction suppose a protocol Π exists for f s.t. $C(\Pi) < \log_2(M)$. Then Π yields at most $2^{C(\Pi)} < 2^{\log_2(M)} = M$ distinct communication patterns. But there are $M * (M-1)$ disjoint choices of $(x, y), (x', y') \in S$. Since $M * (M-1) > M$ there must be some $(x, y), (x', y')$ on which Π yields the same communication pattern.

Then (x, y') must yield the same communication pattern as (x, y) as Bob cannot possibly tell the difference. The argument is symmetric for (x', y) and Alice. One of the two must yield a contradiction and we are done. $\square$

Tiling Method (Max)

With the *fooling set* method, we lower-bounded $C(f)$. Now we'll introduce a new method that both lower- and upper-bounds $C(f)$.

Tiling Method (Max)

With the *fooling set* method, we lower-bounded $C(f)$. Now we'll introduce a new method that both lower- and upper-bounds $C(f)$.

Definition ($M(f)$)

The *matrix of f*, denoted $M(f)$, is the $2^n \times 2^n$ matrix whose (x, y)th entry is the value $f(x, y)$.

Tiling Method (Max)

With the *fooling set* method, we lower-bounded $C(f)$. Now we'll introduce a new method that both lower- and upper-bounds $C(f)$.

Definition ($M(f)$)

The *matrix of f*, denoted $M(f)$, is the $2^n \times 2^n$ matrix whose (x, y)th entry is the value $f(x, y)$.

Example ($M(\vee)$)

	00	01	10	11
00	00	01	10	11
01	01	01	11	11
10	10	11	10	11
11	11	11	11	11

- The green cells are Alice's possible inputs x.
- The blue cells are Bob's possible inputs y.
- The uncolored cells are the matrix $M(f)$.

Tiling Method (Max)

Definition (Combinatorial Rectangle)

A *combinatorial rectangle* in $M(f)$ is any submatrix of M. We say a rectangle $A \times B$ in $M(f)$ is *monochromatic* if for all x, x' in A and y, y' in B, $M_{x,y} = M_{x',y'}$.

Tiling Method (Max)

Definition (Combinatorial Rectangle)

A *combinatorial rectangle* in $M(f)$ is any submatrix of M. We say a rectangle $A \times B$ in $M(f)$ is *monochromatic* if for all x, x' in A and y, y' in B, $M_{x,y} = M_{x',y'}$.

Idea: Each event in a protocol Π splits $M(f)$ into two or more combinatorial rectangles of still-possible values for $f(x, y)$.

Tiling Method (Max)

Definition (Combinatorial Rectangle)

A *combinatorial rectangle* in $M(f)$ is any submatrix of M. We say a rectangle $A \times B$ in $M(f)$ is *monochromatic* if for all x, x' in A and y, y' in B, $M_{x,y} = M_{x',y'}$.

Idea: Each event in a protocol Π splits $M(f)$ into two or more combinatorial rectangles of still-possible values for $f(x, y)$.

Intuition: Much like splitting a circuit C into "C where the first bit is 0" and "C where the first bit is 1".

Tiling Method (Max)

Definition (Combinatorial Rectangle)

A *combinatorial rectangle* in $M(f)$ is any submatrix of M. We say a rectangle $A \times B$ in $M(f)$ is *monochromatic* if for all x, x' in A and y, y' in B, $M_{x,y} = M_{x',y'}$.

Idea: Each event in a protocol Π splits $M(f)$ into two or more combinatorial rectangles of still-possible values for $f(x, y)$.

Intuition: Much like splitting a circuit C into "C where the first bit is 0" and "C where the first bit is 1".

Let's see an example ...

Tiling Method (Max)

Example (Π = LeastSignificantBit, $f =<$)

- $f : \mathbb{B}^3 \times \mathbb{B}^3 \to \mathbb{B}$ is the function that maps (x, y) to 1 if $x < y$ else 0.
- Π = LeastSignificantBit is the naïve protocol where Alice and Bob read off their bits from right to left.

	000	001	010	011	100
000	0	1	1	1	1
001	0	0	1	1	1
010	0	0	0	1	1
011	0	0	0	0	1
100	0	0	0	0	0

Alice: "x = __0"

	000	001	010	011	100
000	0	1	1	1	1
010	0	0	0	1	1
100	0	0	0	0	0

Alice: "x = __1"

	000	001	010	011	100
001	0	0	1	1	1
011	0	0	0	0	1

Tiling Method (Max)

Now we get to the punchline.

Tiling Method (Max)

Now we get to the punchline.

Definition (Monochromatic Tiling)

A *monochromatic tiling* of $M(f)$ is a partition of $M(f)$ into disjoint monochromatic rectangles.

Tiling Method (Max)

Now we get to the punchline.

Definition (Monochromatic Tiling)

A *monochromatic tiling* of $M(f)$ is a partition of $M(f)$ into disjoint monochromatic rectangles.

It's thinking time.

Tiling Method (Max)

Now we get to the punchline.

Definition (Monochromatic Tiling)

A *monochromatic tiling* of $M(f)$ is a partition of $M(f)$ into disjoint monochromatic rectangles.

It's thinking time.

- Then the leaves of the tree induced by Π and rooted at $M(f)$ clearly form a monochromatic tiling of $M(f)$.

Tiling Method (Max)

Now we get to the punchline.

Definition (Monochromatic Tiling)

A *monochromatic tiling* of $M(f)$ is a partition of $M(f)$ into disjoint monochromatic rectangles.

It's thinking time.

- Then the leaves of the tree induced by Π and rooted at $M(f)$ clearly form a monochromatic tiling of $M(f)$.
- The number of leaves in a binary tree can be used to upper-bound its depth.

Tiling Method (Max)

Now we get to the punchline.

Definition (Monochromatic Tiling)

A *monochromatic tiling* of $M(f)$ is a partition of $M(f)$ into disjoint monochromatic rectangles.

It's thinking time.

- Then the leaves of the tree induced by Π and rooted at $M(f)$ clearly form a monochromatic tiling of $M(f)$.
- The number of leaves in a binary tree can be used to upper-bound its depth.
- The depth of the binary tree induced by Π is exactly $C(\Pi)$.

Tiling Method (Max)

It's thinking time.

- Then the leaves of the tree induced by Π and rooted at $M(f)$ clearly form a monochromatic tiling of $M(f)$.
- The number of leaves in a binary tree can be used to upper-bound its depth.
- The depth of the binary tree induced by Π is exactly $C(\Pi)$.

Tiling Method (Max)

It's thinking time.

- Then the leaves of the tree induced by Π and rooted at $M(f)$ clearly form a monochromatic tiling of $M(f)$.
- The number of leaves in a binary tree can be used to upper-bound its depth.
- The depth of the binary tree induced by Π is exactly $C(\Pi)$.

Theorem (The Punchline)

Let $\chi(f)$ denote the minimum number of rectangles in any monochromatic tiling of $M(f)$.

$$log_2 \chi(f) \leq C(f) \leq 16\left(log_2 \chi(f)\right)^2$$

Tiling Method (Max)

NTS: $\log_2 \chi(f) \leq C(f)$.

Proof.

Assume $C(f)$.

Tiling Method (Max)

NTS: $\log_2 \chi(f) \leq C(f)$.

Proof.

Assume $C(f)$. Then $\exists$ a protocol Π in which $\leq C(f)$ bits are communicated between the 2 participants.

Tiling Method (Max)

NTS: $\log_2 \chi(f) \leq C(f)$.

Proof.

Assume $C(f)$. Then $\exists$ a protocol Π in which $\leq C(f)$ bits are communicated between the 2 participants. For simplicity suppose each bit is communicated individually.

Tiling Method (Max)

NTS: $\log_2 \chi(f) \leq C(f)$.

Proof.

Assume $C(f)$. Then $\exists$ a protocol Π in which $\leq C(f)$ bits are communicated between the 2 participants. For simplicity suppose each bit is communicated individually. Then Π induces a tree whose max depth is $C(f)$, whose leaves form a monochromatic partition of $M(f)$.

Tiling Method (Max)

NTS: $\log_2 \chi(f) \leq C(f)$.

Proof.

Assume $C(f)$. Then $\exists$ a protocol Π in which $\leq C(f)$ bits are communicated between the 2 participants. For simplicity suppose each bit is communicated individually. Then Π induces a tree whose max depth is $C(f)$, whose leaves form a monochromatic partition of $M(f)$. Every m.c. partition / $M(f)$ requires $\geq \chi(f)$ rectangles, so the tree induced by Π has $\geq \chi(f)$ leaves.

Tiling Method (Max)

NTS: $\log_2 \chi(f) \leq C(f)$.

Proof.

Assume $C(f)$. Then $\exists$ a protocol Π in which $\leq C(f)$ bits are communicated between the 2 participants. For simplicity suppose each bit is communicated individually. Then Π induces a tree whose max depth is $C(f)$, whose leaves form a monochromatic partition of $M(f)$. Every m.c. partition / $M(f)$ requires $\geq \chi(f)$ rectangles, so the tree induced by Π has $\geq \chi(f)$ leaves. But it's a binary tree so its depth is at least $\log_2 \chi(f)$. $\square$

Tiling Method (Max)

NTS: $C(f) \leq 16\left(\log_2 \chi(f)\right)^2$. [?aho1983notions]

Proof.

- Let $M_1, \ldots, M_{\chi(f)}$ be a monochromatic partitioning of $M(f)$ known ahead of time to both Alice (on the "left") and Bob (on the "right"). Each rectangle M_i can alternatively be written $X_i \times Y_i$.
- Let G_L, G_R be graphs whose nodes are $\{1, \ldots, \chi(f)\}$. There is an edge $i \to j$ in G_L (G_R resp.) if M_i and M_j have a row (column resp.) in common.
- Let $\deg_L(u)$ (resp. $\deg_R(u)$) denote the degree of the node u in the graph G_L (resp. G_R.)
- Let x be Alice's input and y Bob's input.

Tiling Method (Max)

NTS: $C(f) \leq 16\left(\log_2 \chi(f)\right)^2$. [?aho1983notions]

Proof.

- We'll describe the protocol in "rounds". During the rounds, Alice keeps track of Y (a set containing y) and Bob keeps track of X (a set containing x), both of which are initially $\mathbb{B}^n$.
- Both sides know the graphs G_L, G_R and the rectangles M_i ahead of time.

Tiling Method (Max)

NTS: $C(f) \leq 16\big(\log_2 \chi(f)\big)^2$. [?aho1983notions]

Proof.

Each stage proceeds as follows.

1. Alice looks for a rectangle $M_i = X_i \times Y_i$ s.t. $x \in X_i$ and $\deg_L(i) \leq 3\chi(f)/4$.
   1. If she finds some such rectangle then she sends i to Bob.
      1. Bob replies to indicate if $y \in M_i$.
      2. If so then the protocol ends because $f(x, y)$ is the color of M_i.
      3. Otherwise $X := X \cap X_i$, and each rectangle $M_\alpha = X_\alpha \cap Y_\alpha$ is replaced with $(X_i \cap X_\alpha) \times Y_\alpha$.
   2. Otherwise she replies that she found no such rectangle. In this case Bob does what Alice just attempted, symmetrically, with a small caveat ...

Tiling Method (Max)

NTS: $C(f) \le 16\big(\log_2 \chi(f)\big)^2$. [?aho1983notions]

Proof.

- If neither Alice nor Bob could find any M_i with low-enough degree, then they both know that every node i in $G := G_L \cap G_R$ for which $(x,y) \in M_i$ has degree $\ge (3\chi(f)/4)^2 = 9\chi(f)/16 > \chi(f)/2$.
- Let i,j both have degree $\ge \chi(f)/2$ in G. Then some node z is adjacent to i and j in G, by the Pigeonhole Principle. Hence $M_i \cap M_z = \varnothing$ and $M_j \cap M_z = \varnothing$. But the rectangles are monochromatic, hence, M_i and M_j are the same color. So Alice needs to find an M_i containing x and some $y \in Y$ whose degree in G is at least $\chi(f)/2$; and Bob's procedure is symmetric.

Tiling Method (Max)

NTS: $C(f) \leq 16\big(\log_2 \chi(f)\big)^2$. [?aho1983notions]

Proof.

- In the worst case for each stage, the first participant sends "nothing found" (1 bit), the second participant sends some i ($\leq \log_2(\chi(f))$ bits), and the first participant replies with some j ($\leq \log_2(\chi(f))$ bits). So in the worst case each round requires $\leq 2 + 2\log_2(\chi(f))$ bits.
- The protocol ends after at most n rounds where $(3\chi(f)/4)^n \approx 1$, i.e., after $\log_{(4/3)}(\chi(f))$ rounds.
- So total communication complexity is $\leq \log_{(4/3)}(\chi(f)) * (1 + 2\log_2(\chi(f)))$.

Tiling Method (Max)

NTS: $C(f) \leq 16\big(\log_2 \chi(f)\big)^2$. [?aho1983notions]

Proof.

For $\chi(f) \geq 2$:

$$\begin{aligned}
C(f) &\leq \log_{(4/3)}(\chi(f)) * (1 + 2\log_2(\chi(f))) \\
&= \frac{\log_2(\chi(f))}{\log_2(4/3)} * (1 + 2\log_2(\chi(f))) \\
&< 2.5 * \log_2(\chi(f)) * (1 + 2\log_2(\chi(f))) \\
&\leq 2.5 * \log_2(\chi(f)) * 3\log_2(\chi(f))) \\
&= 7.5\log_2^2(\chi(f)) \\
&\leq 16\log_2^2(\chi(f))
\end{aligned}$$

I'm almost certainly missing a factor of 2, which would explain the choice of 16. -Max. $\square$

2-Party Discrepency Method (Max)

Recall that $\chi(f)$ induces both lower and upper bounds on $C(f)$. So if any bound on $\chi(f)$ induces a bound on $C(f)$. We are about to prove the following lower-bound on $\chi(f)$:

$$\text{Disc}(A \times B) = \frac{1}{2^n * 2^n} \Big| \sum_{a \in A, b \in B} (-1)^{M_{a,b}} \Big| \leq \chi(f) \quad \leq \chi(f)$$

2-Party Discrepency Method (Max)

When we partition $M(f)$ into some number of rectangles, the sizes of the rectangles must add up to the size of $M(f)$.

2-Party Discrepency Method (Max)

When we partition $M(f)$ into some number of rectangles, the sizes of the rectangles must add up to the size of $M(f)$.

Hence, if $\chi(f) \leq K$ for some integer K, then $M(f)$ must have a m.c. rectangle containing at least $2^n * 2^n / K$ entries.

2-Party Discrepency Method (Max)

When we partition $M(f)$ into some number of rectangles, the sizes of the rectangles must add up to the size of $M(f)$.

Hence, if $\chi(f) \leq K$ for some integer K, then $M(f)$ must have a m.c. rectangle containing at least $2^n * 2^n / K$ entries.

Proof.

Suppose $\chi(f) \leq K$ for some integer K.

2-Party Discrepency Method (Max)

When we partition $M(f)$ into some number of rectangles, the sizes of the rectangles must add up to the size of $M(f)$.

Hence, if $\chi(f) \leq K$ for some integer K, then $M(f)$ must have a m.c. rectangle containing at least $2^n * 2^n/K$ entries.

Proof.

Suppose $\chi(f) \leq K$ for some integer K. If $\chi(f) = K$ then $\exists$ a partioning of $M(f)$ into K m.c. rects, in which case at least 1 must have size $\geq |M(f)|/K$, i.e., $2^n * 2^n/K$.

2-Party Discrepency Method (Max)

When we partition $M(f)$ into some number of rectangles, the sizes of the rectangles must add up to the size of $M(f)$.

Hence, if $\chi(f) \leq K$ for some integer K, then $M(f)$ must have a m.c. rectangle containing at least $2^n * 2^n/K$ entries.

Proof.

Suppose $\chi(f) \leq K$ for some integer K. If $\chi(f) = K$ then $\exists$ a partioning of $M(f)$ into K m.c. rects, in which case at least 1 must have size $\geq |M(f)|/K$, i.e., $2^n * 2^n/K$. On the other hand if $\chi(f) < K$ then $\chi(f) = K'$ for some $K' < K$ and then $M(f)$ can be partitioned into K' monochromatic rectangles, at least 1 of which has size $\geq |M(f)|/K'$, which is strictly larger than $|M(f)|/K$.

2-Party Discrepency Method (Max)

When we partition $M(f)$ into some number of rectangles, the sizes of the rectangles must add up to the size of $M(f)$.

Hence, if $\chi(f) \leq K$ for some integer K, then $M(f)$ must have a m.c. rectangle containing at least $2^n * 2^n / K$ entries.

Proof.

Suppose $\chi(f) \leq K$ for some integer K. If $\chi(f) = K$ then $\exists$ a partioning of $M(f)$ into K m.c. rects, in which case at least 1 must have size $\geq |M(f)|/K$, i.e., $2^n * 2^n/K$. On the other hand if $\chi(f) < K$ then $\chi(f) = K'$ for some $K' < K$ and then $M(f)$ can be partitioned into K' monochromatic rectangles, at least 1 of which has size $\geq |M(f)|/K'$, which is strictly larger than $|M(f)|/K$. Either way the conjecture holds. $\square$

2-Party Discrepency Method (Max)

Suppose that $M(f)$ contains a monochromatic rectangle $A \times B$ having at least $2^n * 2^n / K$ entries.

2-Party Discrepency Method (Max)

Suppose that $M(f)$ contains a monochromatic rectangle $A \times B$ having at least $2^n * 2^n / K$ entries. Since $A \times B$ is monochromatic, this implies that:

$$\sum_{a \in A, b \in B} (-1)^{M_{a,b}} = \begin{cases} -1 * |A \times B| & \text{if it's colored 1} \\ +1 * |A \times B| & \text{if it's colored 0} \end{cases}$$

2-Party Discrepency Method (Max)

Suppose that $M(f)$ contains a monochromatic rectangle $A \times B$ having at least $2^n * 2^n/K$ entries. Since $A \times B$ is monochromatic, this implies that:

$$\sum_{a \in A, b \in B} (-1)^{M_{a,b}} = \begin{cases} -1 * |A \times B| & \text{if it's colored 1} \\ +1 * |A \times B| & \text{if it's colored 0} \end{cases}$$

So if we wrap an absolute value above our sum, we get:

$$\left| \sum_{a \in A, b \in B} (-1)^{M_{a,b}} \right| = \text{ the size of the rectangle } A \times B$$

2-Party Discrepency Method (Max)

Suppose that $M(f)$ contains a monochromatic rectangle $A \times B$ having at least $2^n * 2^n/K$ entries. Since $A \times B$ is monochromatic, this implies that:

$$\sum_{a \in A, b \in B} (-1)^{M_{a,b}} = \begin{cases} -1 * |A \times B| & \text{if it's colored 1} \\ +1 * |A \times B| & \text{if it's colored 0} \end{cases}$$

So if we wrap an absolute value above our sum, we get:

$$|\sum_{a \in A, b \in B} (-1)^{M_{a,b}}| = \text{ the size of the rectangle } A \times B$$

But we already assumed that $A \times B$ has at least $2^n * 2^n/K$ entries, hence:

$$|\sum_{a \in A, b \in B} (-1)^{M_{a,b}}| \geq 2^n * 2^n/K$$

2-Party Discrepency Method (Max)

Let's divide both size by $2^n * 2^n$, for fun and profit.

$$\frac{1}{2^n * 2^n} \Big| \sum_{a \in A, b \in B} (-1)^{M_{a,b}} \Big| \geq 1/K$$

2-Party Discrepency Method (Max)

Let's divide both size by $2^n * 2^n$, for fun and profit.

$$\frac{1}{2^n * 2^n} \Big| \sum_{a \in A, b \in B} (-1)^{M_{a,b}} \Big| \geq 1/K$$

Mathematicians like to name things.

Definition (Discrepency)

The *discrepency* of a rectangle $A \times B$ of $M(f)$ is exactly the following.

$$\mathrm{Disc}(A \times B) = \frac{1}{2^n * 2^n} \Big| \sum_{a \in A, b \in B} (-1)^{M_{a,b}} \Big|$$

The *discrepency of* $M(f)$ is the max disc among its rectangles.

2-Party Discrepency Method (Max)

Now that we've named this thing, let's re-write our inequality.

$$\mathrm{Disc}(A \times B) \geq 1/K$$

2-Party Discrepency Method (Max)

Now that we've named this thing, let's re-write our inequality.

$$\text{Disc}(A \times B) \geq 1/K$$

Taking inverses:

$$\frac{1}{\text{Disc}(A \times B)} \leq K$$

2-Party Discrepency Method (Max)

Now that we've named this thing, let's re-write our inequality.

$$\text{Disc}(A \times B) \geq 1/K$$

Taking inverses:

$$\frac{1}{\text{Disc}(A \times B)} \leq K$$

Certainly $\chi(f) \leq \chi(f)$, so supplanting $\chi(f)$ for K in the statement, we get:

Lemma (2-Party Discrepency Method)

$$\frac{1}{Disc(A \times B)} \leq \chi(f)$$

Multi-Party Problem (Jake)

- There are k of us.
- We all place a sticky note with some value $b \in \mathbb{B}^n$ on our heads.
- Without talking, we must compute some predetermined function via a predetermined protocol. All communication must be done through the whiteboard in front of us.
- The goal is for one player, after some amount of communication, to write the value $f(\text{sticky-note}_1, \ldots, \text{sticky-note}_k)$ on the whiteboard.

Multi-Party Problem (Jake)

Example (*MajorityParity*)

$MajPar : \mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ is precisely $\langle x_1, x_2, x_3 \rangle \mapsto 1$ if $\bigoplus_{i=1}^{n} maj(x_{1i}, x_{2i}, x_{3i})$ else 0.

Multi-Party Problem (Jake)

Example (*MajorityParity*)

$MajPar : \mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B}^n \rightarrow \mathbb{B}$ is precisely $\langle x_1, x_2, x_3 \rangle \mapsto 1$ if $\bigoplus_{i=1}^{n} maj(x_{1i}, x_{2i}, x_{3i})$ else 0.

For example: $f(1101, 1001, 1011) = \bigoplus 1001 = 0$

Multi-Party Problem (Jake)

Example (*MajorityParity*)

$MajPar : \mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B}^n \rightarrow \mathbb{B}$ is precisely $\langle x_1, x_2, x_3 \rangle \mapsto 1$ if $\bigoplus_{i=1}^n maj(x_{1i}, x_{2i}, x_{3i})$ else 0.

For example: $f(1101, 1001, 1011) = \bigoplus 1001 = 0$

Example protocol Π:

Player 1	Player 2	Player 3
$x_2 = 1001$	$x_1 = 1101$	$x_1 = 1101$
$x_3 = 1011$	$x_3 = 1011$	$x_2 = 1001$
$parity(10\_1)$	$parity(1\_\_1)$	$parity(1\_01)$
$p_1 = 0$	$p_2 = 0$	$p_3 = 0$
$parity(p_1 p_2 p_3) = parity(000) = 0$		

Multi-Party Problem (Jake)

Before we talked about *rectangles*. Now: *cylinders*.

Definition (Cylinder)

A *cylinder in dimension i* is a subset S of the inputs $(\mathbb{B}^n)^k$ such that if $(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k) \in S$, then for all $x_i' \in \mathbb{B}^n$, so is $(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_k)$.

Multi-Party Problem (Jake)

Before we talked about *rectangles*. Now: *cylinders*.

Definition (Cylinder)

A *cylinder in dimension* i is a subset S of the inputs $(\mathbb{B}^n)^k$ such that if $(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k) \in S$, then for all $x_i' \in \mathbb{B}^n$, so is $(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_k)$.

Definition (Cylinder Intersection)

A *cylinder intersection* is a set $C = \bigcap_{i=1}^{k} T_i$ where each T_i is a cylinder in dimension i.

Multi-Party Problem (Jake)

Before we talked about *rectangles*. Now: *cylinders*.

Definition (Cylinder)

A *cylinder in dimension i* is a subset S of the inputs $(\mathbb{B}^n)^k$ such that if $(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k) \in S$, then for all $x_i' \in \mathbb{B}^n$, so is $(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_k)$.

Definition (Cylinder Intersection)

A *cylinder intersection* is a set $C = \bigcap_{i=1}^{k} T_i$ where each T_i is a cylinder in dimension i.

Lemma (Generalize $\chi(f)$)

If every partition of $M(f)$ into m.c. cylinder intersections requires at least R of them, then $C(f) \geq \lceil log_2(R) \rceil$.

Multi-Party Discrepency Method (Max)

Definition (Multi-Party Discrepency)

Suppose

$$f : \underbrace{\mathbb{B}^n \times \ldots \times \mathbb{B}^n}_{k \text{ times}} \to \mathbb{B}$$

is a function. Then the *k-party discrepency of f* is defined as follows, where T ranges over all cylinder intersections of f.

$$\text{Disc}(f) = \frac{1}{(2^n)^k} \max_T \Big| \sum_{(x_1,\ldots,x_k)\in T} f(x_1, \ldots, x_k) \Big|$$

Multi-Party Discrepency Method (Max)

Definition (Multi-Party Discrepency)

Suppose

$$f : \underbrace{\mathbb{B}^n \times \ldots \times \mathbb{B}^n}_{k \text{ times}} \to \mathbb{B}$$

is a function. Then the *k-party discrepency of f* is defined as follows, where T ranges over all cylinder intersections of f.

$$\text{Disc}(f) = \frac{1}{(2^n)^k} \max_T \Big| \sum_{(x_1,\ldots,x_k)\in T} f(x_1,\ldots,x_k) \Big|$$

Man this is really complicated.

Multi-Party Discrepency Method (Max)

Definition (Multi-Party Discrepency)

Suppose

$$f : \underbrace{\mathbb{B}^n \times \ldots \times \mathbb{B}^n}_{k \text{ times}} \to \mathbb{B}$$

is a function. Then the *k-party discrepency of f* is defined as follows, where T ranges over all cylinder intersections of f.

$$\text{Disc}(f) = \frac{1}{(2^n)^k} \max_T \Big| \sum_{(x_1,\ldots,x_k)\in T} f(x_1,\ldots,x_k) \Big|$$

Man this is really complicated. Could we lower-bound it statistically?

Multi-Party Discrepency Method (Max)

First some extremely tedious definitions.

Multi-Party Discrepency Method (Max)

First some extremely tedious definitions.

Definition ((k, n)-Cube)

A (k, n)-cube is a set D of the form $D = \{a_1, a_1'\} \times \ldots \times \{a_k, a_k'\}$ where each $a_i, a_i' \in \mathbb{B}^n$. A point $\vec{d} \in D$ is a vector $(x_1, x_2, \ldots, x_k)$ s.t. each $x_i \in \{a_i, a_i'\}$.

Multi-Party Discrepency Method (Max)

First some extremely tedious definitions.

Definition ((k,n)-Cube)

A (k,n)-cube is a set D of the form $D = \{a_1, a_1'\} \times \ldots \times \{a_k, a_k'\}$ where each $a_i, a_i' \in \mathbb{B}^n$. A point $\vec{d} \in D$ is a vector $(x_1, x_2, \ldots, x_k)$ s.t. each $x_i \in \{a_i, a_i'\}$.

Definition ($\mathcal{E}$)

Let $f : (\mathbb{B}^n)^k \to \mathbb{B}$ be a function.

$$\mathcal{E}(f) = \mathop{\mathbb{E}}_{\substack{D \text{ is a} \\ (k,n)\text{-cube}}} \left[\prod_{\vec{d} \in D} f(\vec{d}) \right]$$

I.e., $\mathcal{E}(f) = E[$*given an arbitrary cube D, what is the product of the image of f over all the points $\vec{d} \in D$?*$]$

Multi-Party Discrepency Method (Max)

Although somewhat scary-looking, this definition pays dividends immediately.

Lemma (*k*-Party Discrepency Bound)

If $f : (\mathbb{B}^n)^k \to \mathbb{B}$ *is a function, then* $Disc(f) \leq (\mathcal{E}(f))^{1/2^k}$.

Multi-Party Discrepency Method (Max)

Although somewhat scary-looking, this definition pays dividends immediately.

Lemma (k-Party Discrepency Bound)

If $f : (\mathbb{B}^n)^k \to \mathbb{B}$ is a function, then $Disc(f) \leq (\mathcal{E}(f))^{1/2^k}$.

Proof sketch:

Proof.

Given any cylinder intersection and (n,k)-cube, what is the expectation / the image of the cube? What if we only consider points in the cylinder intersection? Derive a lower bound on $\mathcal{E}(f)$ like

$$\mathcal{E}(f) \geq E_{x_1,\ldots,x_k}[f(x_1,\ldots,x_k)(1 \text{ if } (x_1,\ldots,x_k) \in C \text{ else } 0)]^{2k}$$

given a cylinder intersection C. Argue from the def. of the k-party discrepency that this gives a natural lower-bound $\mathcal{E}(f) \geq \mathrm{Disc}(f)^{2k}$. But this implies $\mathrm{Disc}(f) \leq (\mathcal{E}(f))^{1/2^k}$, and we're done. $\square$

Non-Deterministic (Jake)

- Defined similarly to NP.

Non-Deterministic (Jake)

- Defined similarly to NP.
- Consider a two party problem (we can generalize to multi-party from here). Each player is given their input along with some nondeterministic guess z of length m that may depend on the given inputs.
  $C(f) = m +$ communication, and $f(x, y) = 1$ iff $\exists z$ that makes the players output 1.

Non-Deterministic (Jake)

- Defined similarly to NP.
- Consider a two party problem (we can generalize to multi-party from here). Each player is given their input along with some nondeterministic guess z of length m that may depend on the given inputs. $C(f) = m +$ communication, and $f(x, y) = 1$ iff $\exists z$ that makes the players output 1.
- NP^{CC} is the class of nondeterministic functions f s.t. $C(f) = n^k$. coNP^{CC} is defined similarly, i.e., $g(x, y) = 1 - f(x, y)$ for $f \in \mathrm{NP}^{CC}$.

Non-Deterministic (Jake)

- Defined similarly to NP.
- Consider a two party problem (we can generalize to multi-party from here). Each player is given their input along with some nondeterministic guess z of length m that may depend on the given inputs.
  $C(f) = m +$ communication, and $f(x,y) = 1$ iff $\exists z$ that makes the players output 1.
- NP^{CC} is the class of nondeterministic functions f s.t. $C(f) = n^k$. coNP^{CC} is defined similarly, i.e., $g(x,y) = 1 - f(x,y)$ for $f \in \mathrm{NP}^{CC}$.
- Claim: $\mathrm{NP}^{CC} \cap \mathrm{coNP}^{CC} = \mathrm{P}^{CC}$. This is shown by relating the communication complexities of $f \in \mathrm{NP}^{CC}$ and $\overline{f} \in \mathrm{coNP}^{CC}$.

Non-Deterministic (Jake)

- Defined similarly to NP.
- Consider a two party problem (we can generalize to multi-party from here). Each player is given their input along with some nondeterministic guess z of length m that may depend on the given inputs.
  $C(f) = m +$ communication, and $f(x, y) = 1$ iff $\exists z$ that makes the players output 1.
- NP^{CC} is the class of nondeterministic functions f s.t. $C(f) = n^k$. coNP^{CC} is defined similarly, i.e., $g(x, y) = 1 - f(x, y)$ for $f \in \mathrm{NP}^{CC}$.
- Claim: $\mathrm{NP}^{CC} \cap \mathrm{coNP}^{CC} = \mathrm{P}^{CC}$. This is shown by relating the communication complexities of $f \in \mathrm{NP}^{CC}$ and $\overline{f} \in \mathrm{coNP}^{CC}$.
- $C(f) = k$, and $C(\overline{f}) = 10kl$ for some complexity l.

Randomized (Max)

- All players share a random string r.
- $R(f) := E[|C(f)|]$ is the *expected* # bits communicated.
- A randomized communication protocol is allowed to output the wrong answer at most $1/3$ of the time.
- The "public coin" scenario is where you assume r is known ahead-of-time and does not need to be communicated.
- The "private coin" scenario is where someone computes r and shares it with the other players, so it contributes to the size of the protocol.

Randomized (Max)

It turns out randomized protocols can be much faster [?coins], e.g.,

$$C(=) \geq n$$

but

$$R(=) \in \mathcal{O}(\log n) \text{ with a private coin, or}$$
$$\in \mathcal{O}(1) \text{ with a public coin}$$

Randomized (Max)

The protocol with a private coin turns out to be $\textsc{ModCoin}$, where

1. Alice chooses $\textsc{Coin} \in \mathbb{Z}_{2n}$ randomly;
2. Alice sends $\langle x \bmod \textsc{Coin}, \textsc{Coin} \rangle$ to Bob;
3. Bob sends 1 iff $x \bmod \textsc{Coin} = y \bmod \textsc{Coin}$ else 0, back to Alice.

- If $x = y$ the protocol is correct with probability 1.
- If $x \neq y$ the protocol is correct with probability $P[x \bmod \textsc{Coin} \neq y \bmod \textsc{Coin}]$ which turns out to exceed $1/2$.

Randomized (Max)

The protocol with a public coin turns out to be CHECKSUM, which

- if $x = y$ is correct with probability 1, and
- if $x \neq y$ is correct with probability 1/2.

So if you just do this a second time to confirm $x \neq y$ you get 1/4 error rate and you are done.

References I