# Property Testing of Boolean Functions

Erik Waingarten (Penn)

**My goal:** Definitions and some important ideas.

- Linearity

  (most of the talk)

- Monotonicity

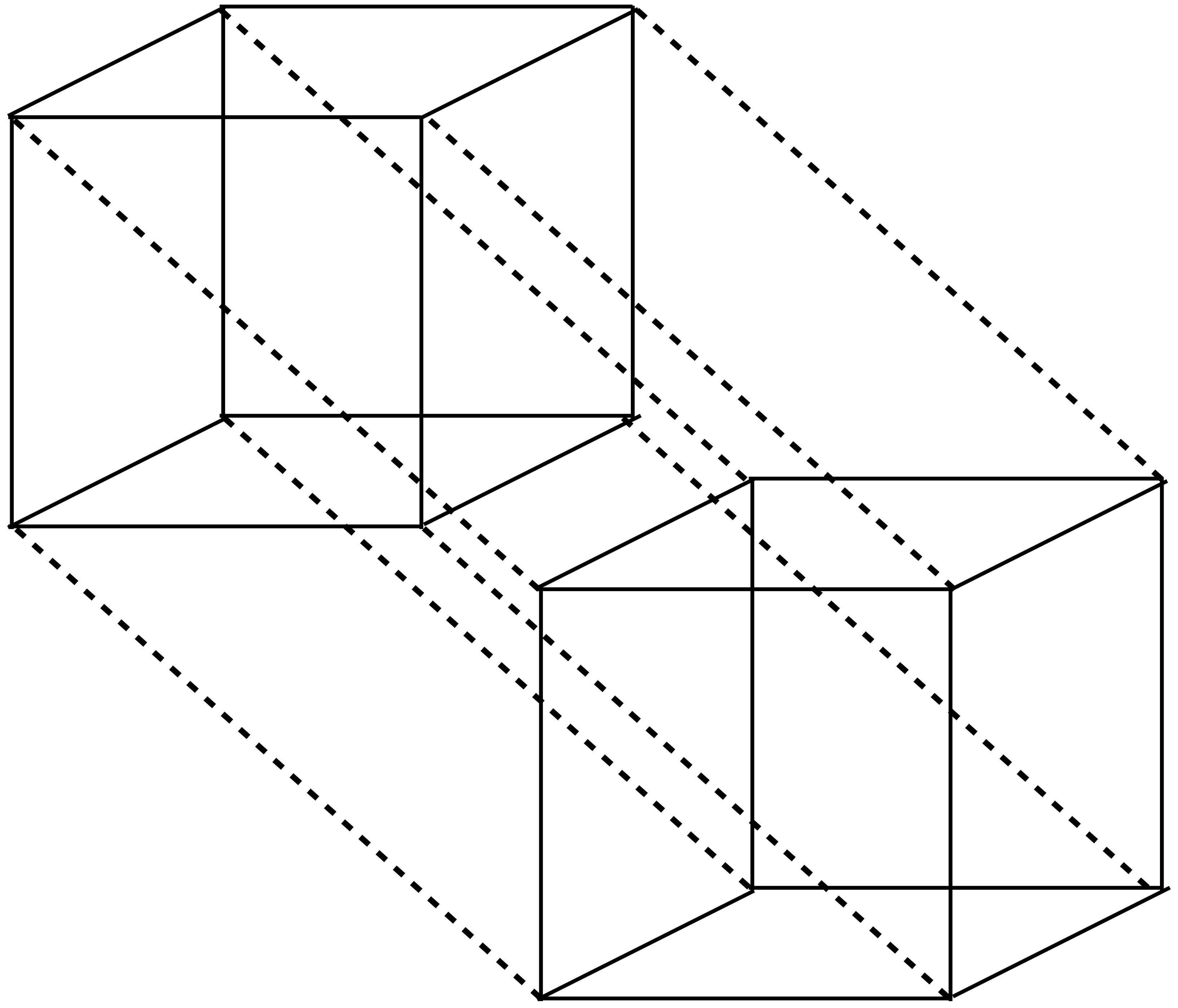
- Juntas

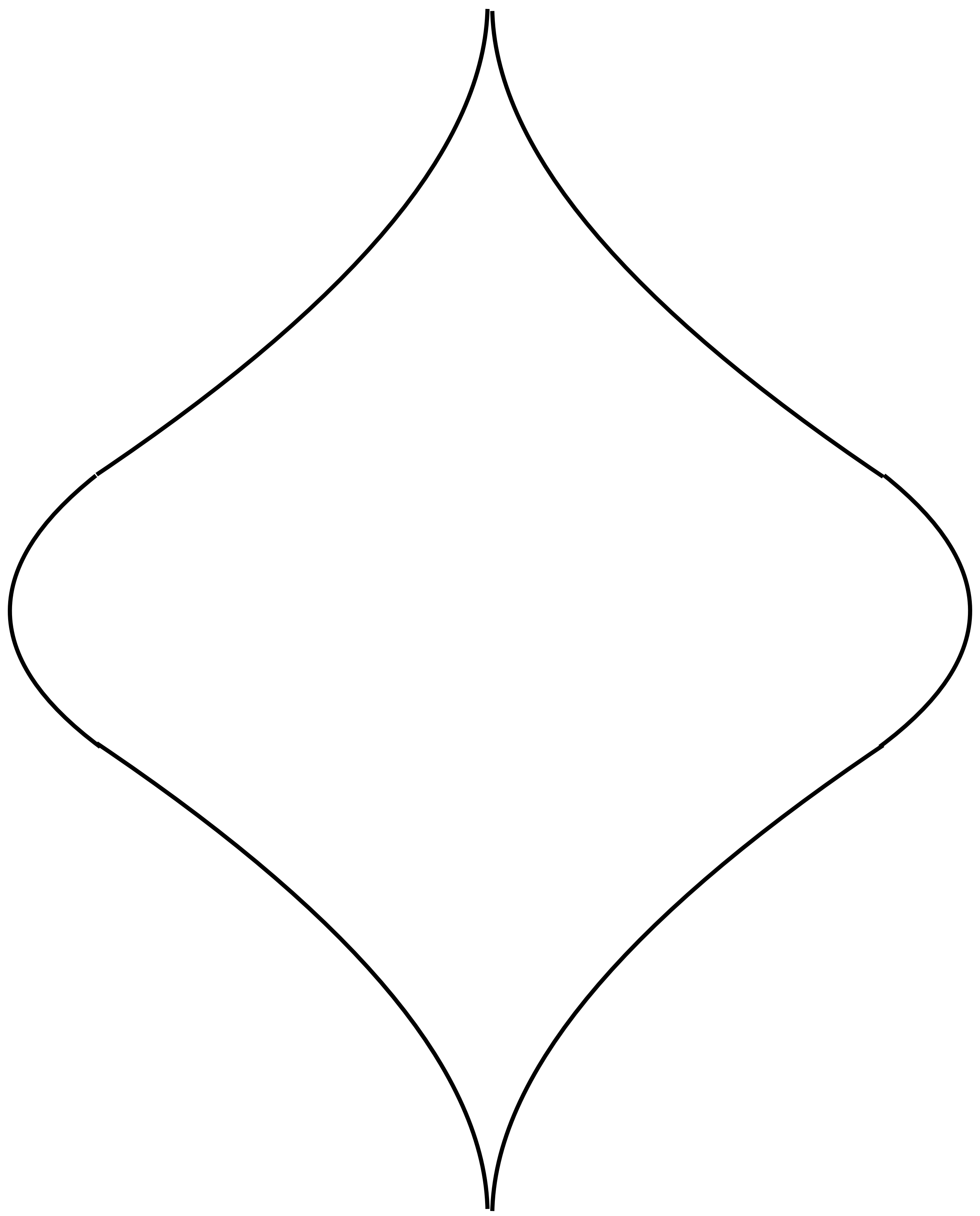
Reference: “Introduction to Property Testing” by Goldreich ‘17
“Algorithmic and Analysis Techniques in Property Testing” by Ron ‘10

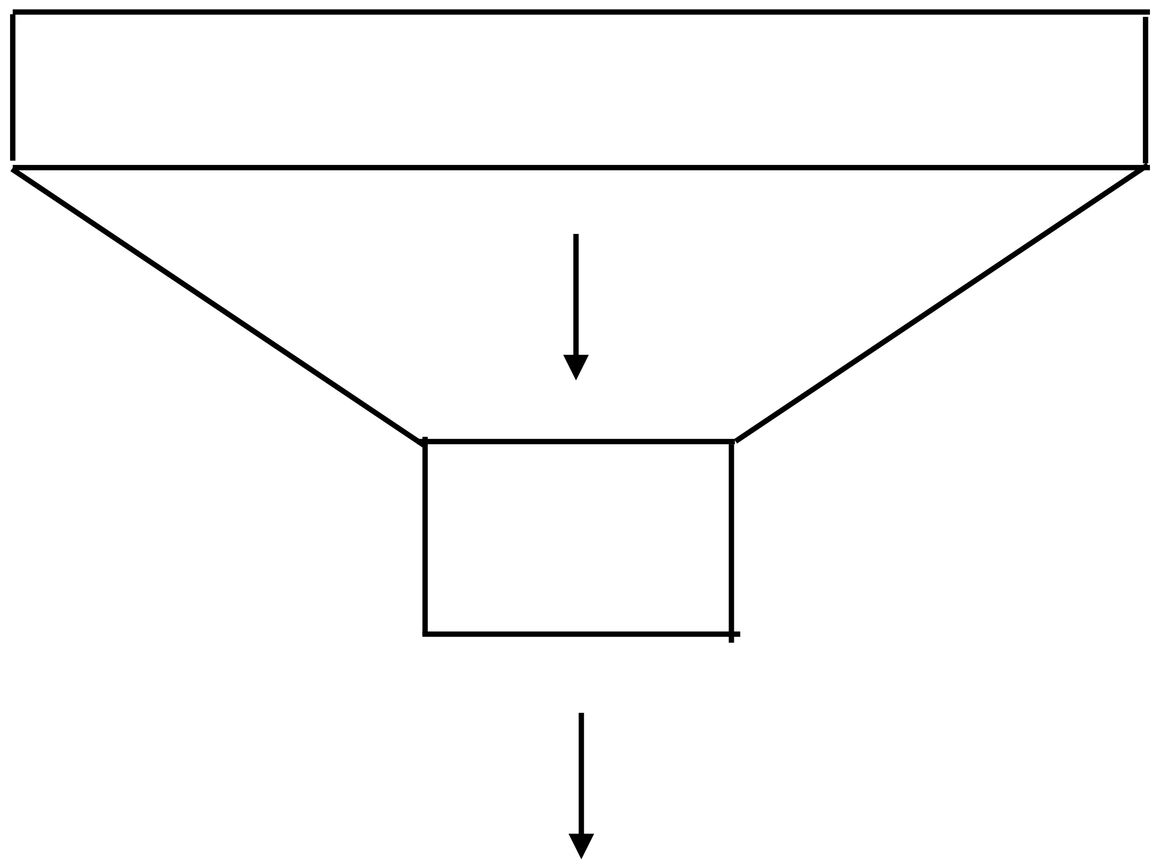
# A Boolean function encodes a set of its domain.

- This talk: domain is Boolean hypercube.

$$f\colon \{0,1\}^n \to \{0,1\}$$



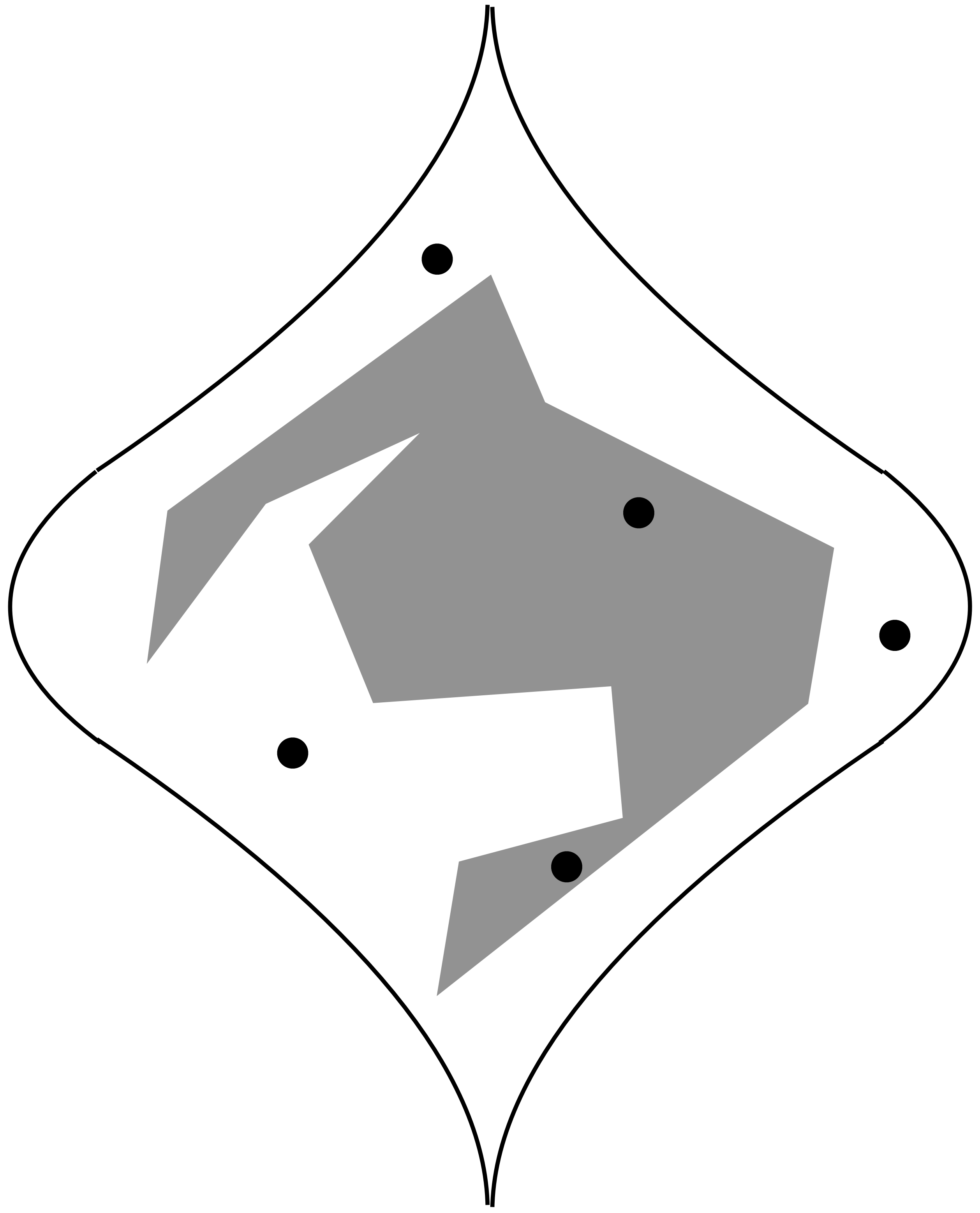




A property testing algorithm decides whether a function has a particular property (approximately).

Does the function have a particular property?

$$f : \{0,1\}^n \to \{0,1\}$$



$x_1 \in \{0,1\}^n \longrightarrow$

$\longleftarrow f(x_1)$

$\vdots$

$x_q \in \{0,1\}^n \longrightarrow$

$\longleftarrow f(x_q)$

YES or NO

A property testing algorithm decides whether a function has a particular property (approximately).

Does the function have a particular property?

YES or NO

$x_1 \in \{0,1\}^n \longrightarrow$

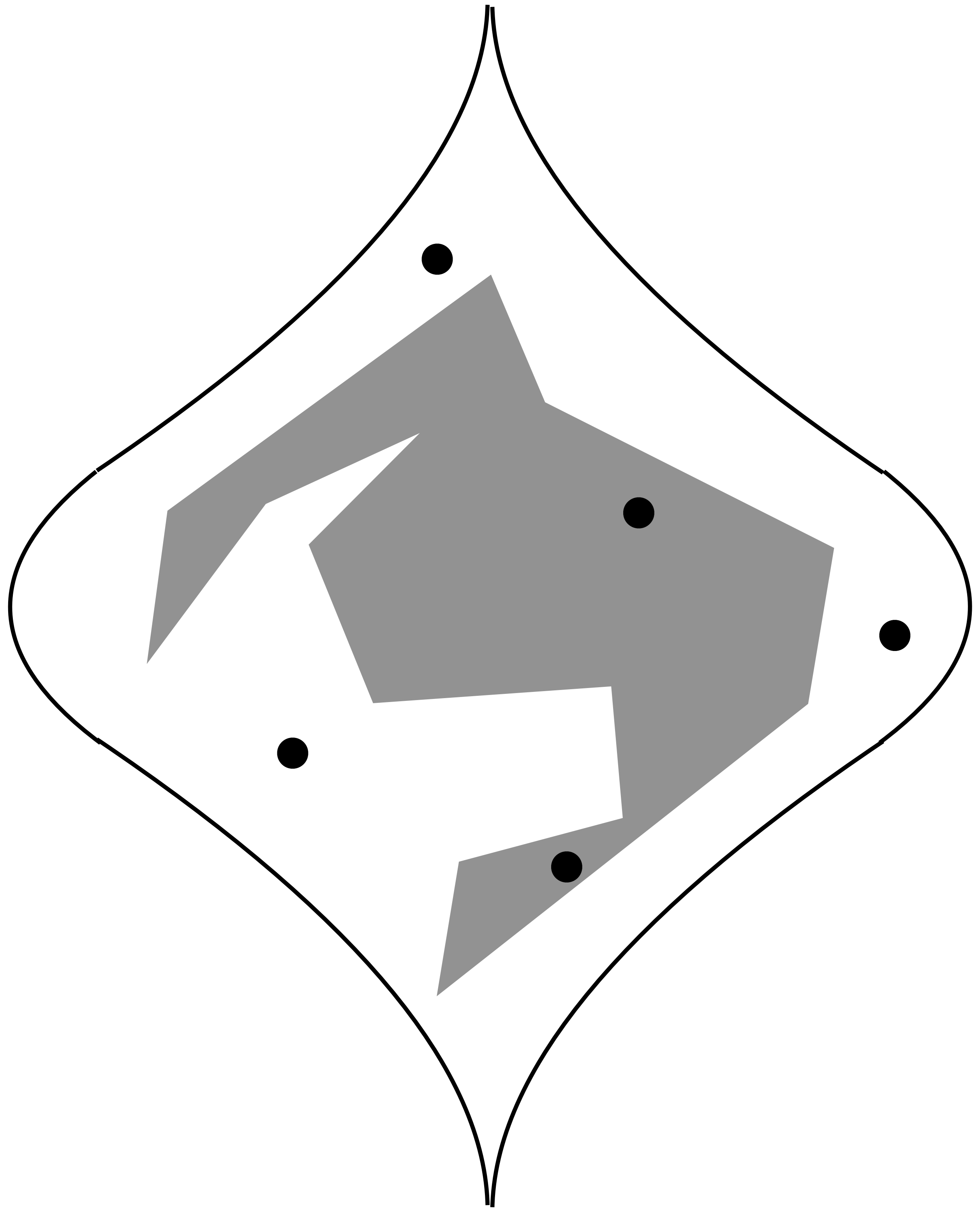
$\longleftarrow f(x_1)$

$\vdots$

$x_q \in \{0,1\}^n \longrightarrow$

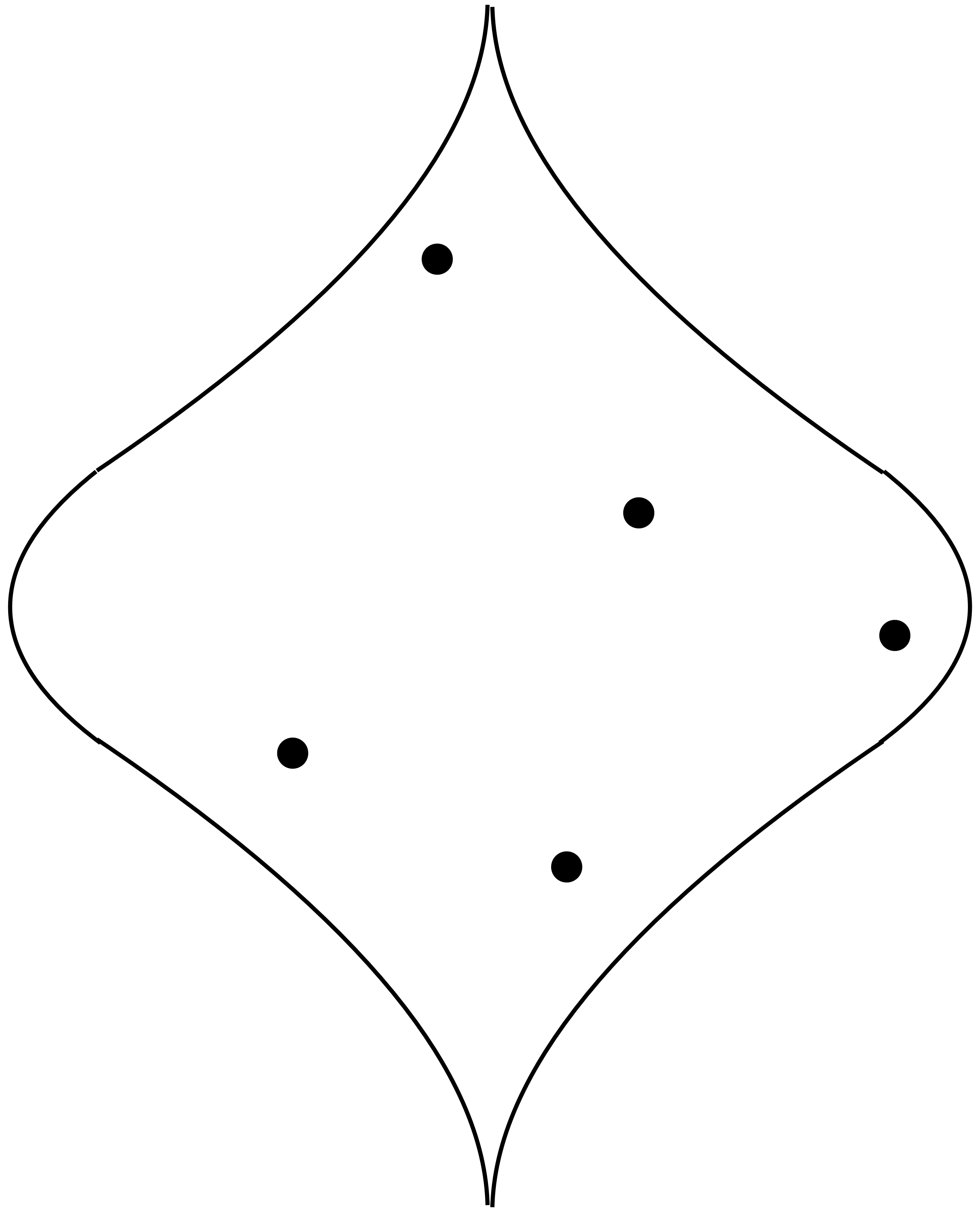
$\longleftarrow f(x_q)$

$$f : \{0,1\}^n \to \{0,1\}$$



# Is the function always 0?

$$f : \{0,1\}^n \to \{0,1\}$$



$x_1 \in \{0,1\}^n$ →

← 0

$\vdots$

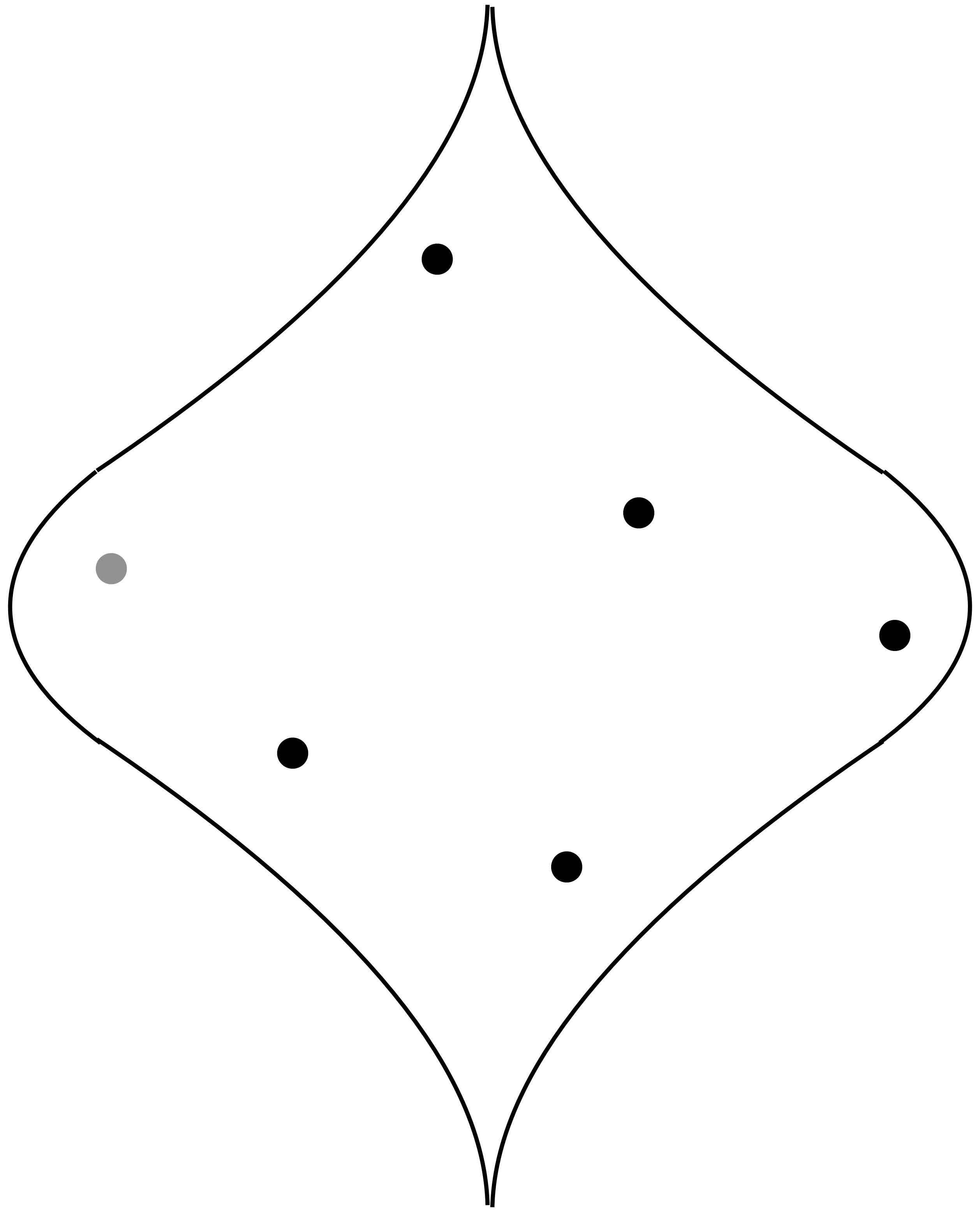
$x_q \in \{0,1\}^n$ →

← 0

YES or NO

# Is the function always 0?

$$\tilde{f} : \{0,1\}^n \to \{0,1\}$$



$x_1 \in \{0,1\}^n$ → 

← $0$

⋮

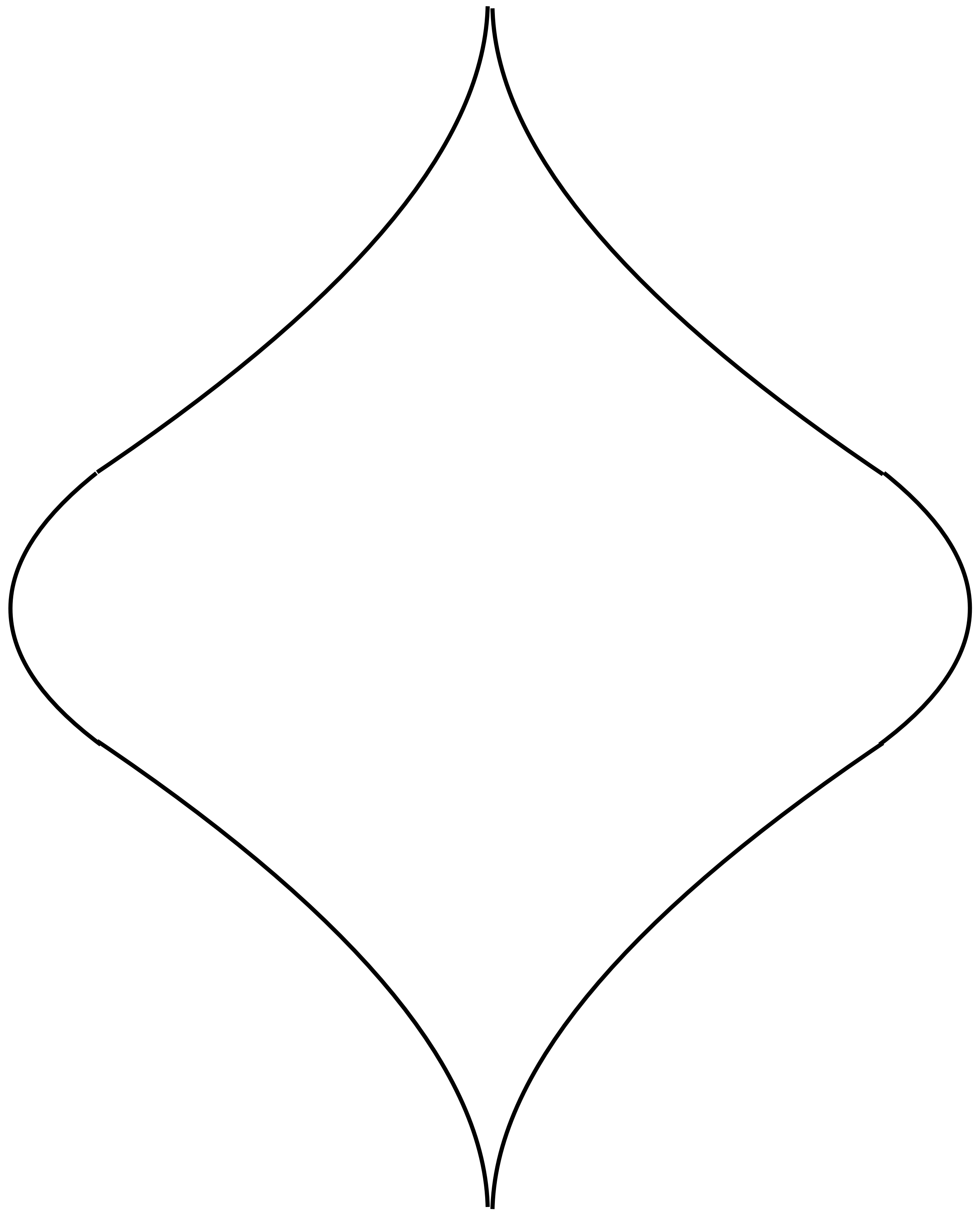
$x_q \in \{0,1\}^n$ →

← $0$

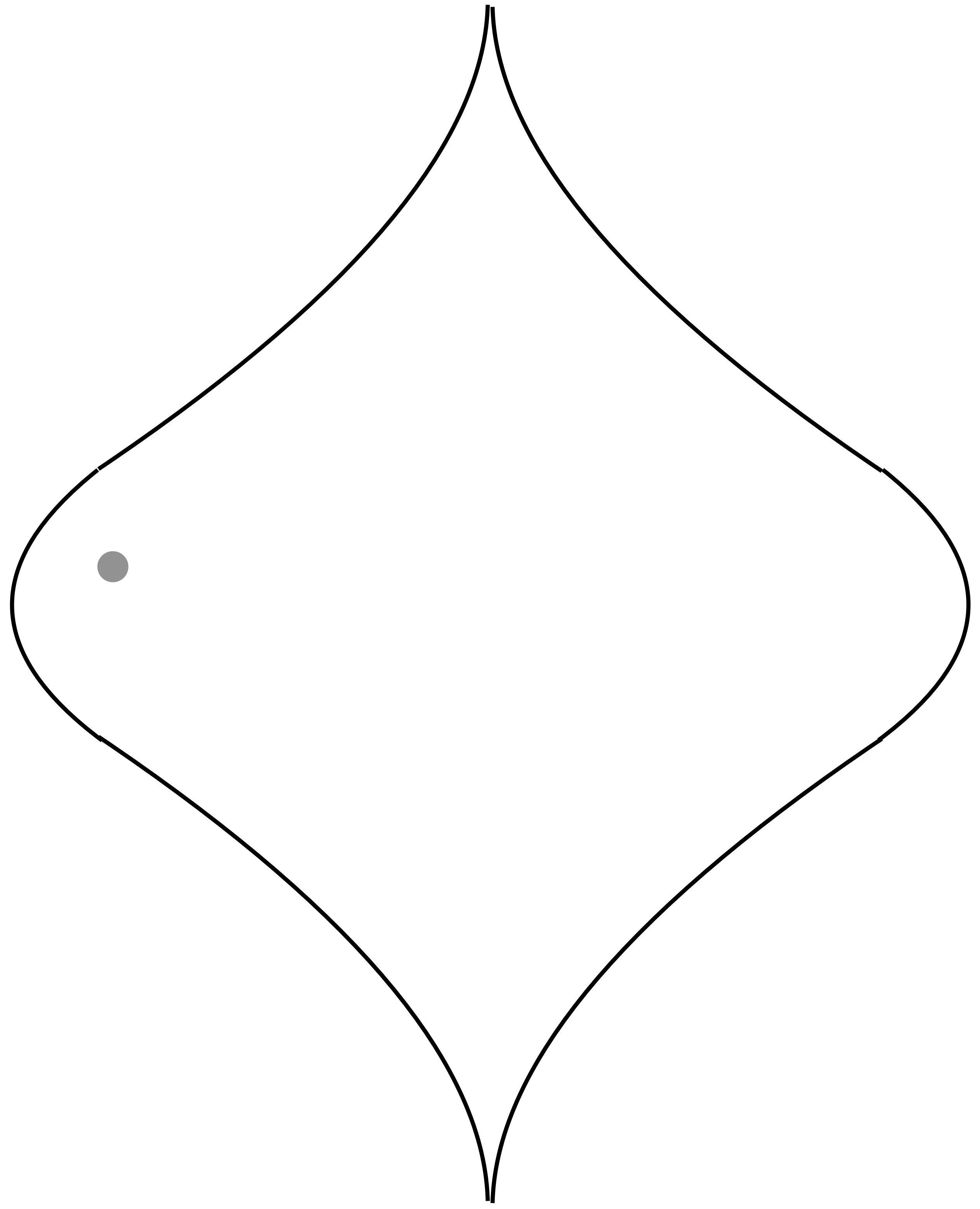
YES or NO

Unless Alice queries hidden point, looks like constant 0 function.

$$f : \{0,1\}^n \to \{0,1\}$$



$$\tilde{f} : \{0,1\}^n \to \{0,1\}$$



Query Complexity
Lower Bound

$$\Omega(2^n)$$

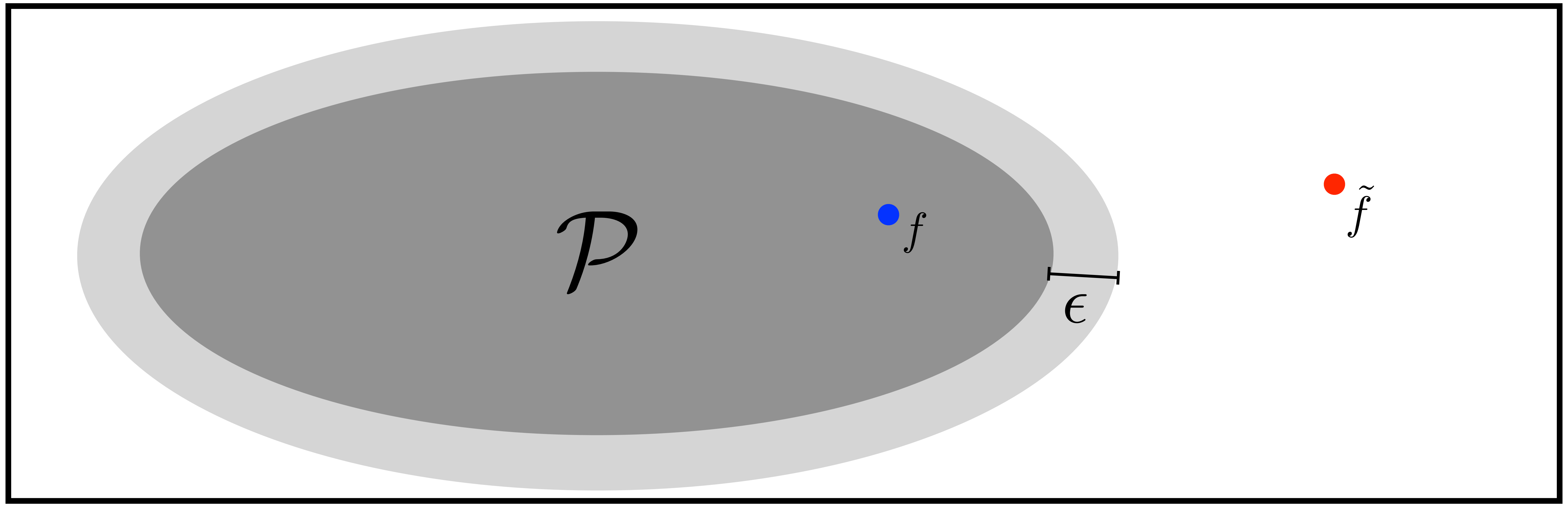
# Property Testing as Approximate Decision Making:

**Definition**: A property testing algorithm for a property $\mathcal{P}$ is a randomized algorithm which receives query access to an unknown function $f : \{0,1\}^n \to \{0,1\}$:

- If $f \in \mathcal{P}$, the algorithm output YES w.p at least $2/3$.
- If $f$ is $\epsilon$-far from $\mathcal{P}$, the algorithm outputs NO w.p at least $2/3$.

$f$ output YES w.p 2/3

$\tilde{f}$ output NO w.p 2/3



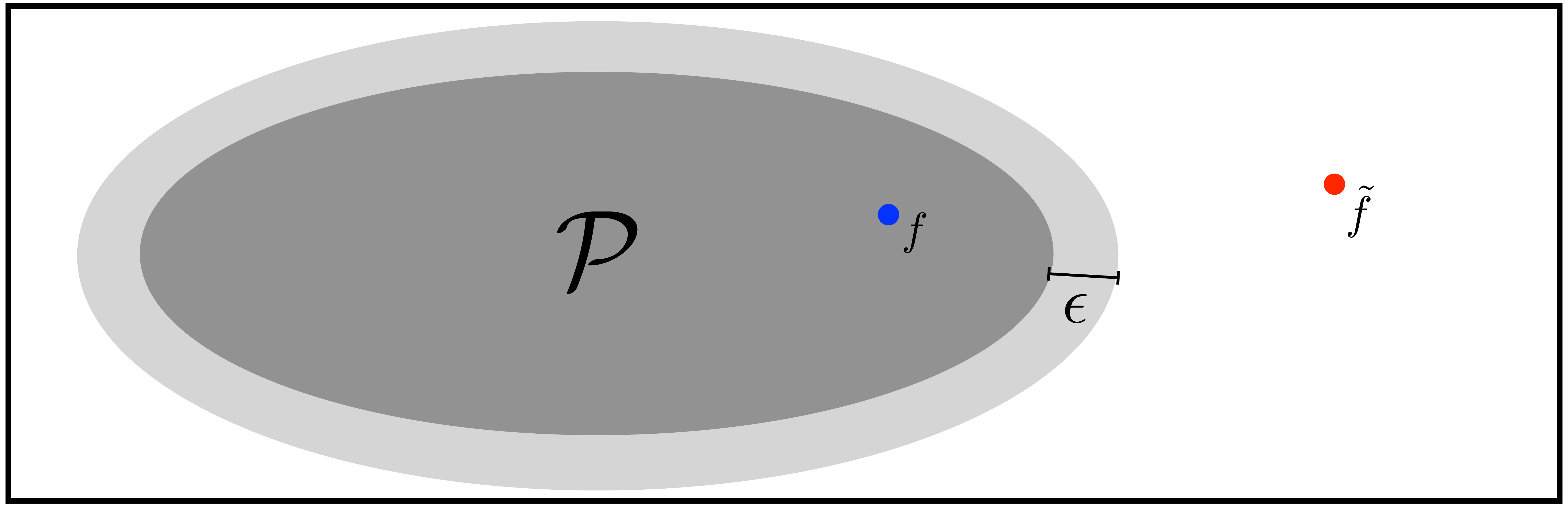
Property Testing as Approximate Decision Making:

**Definition**: A property testing algorithm for a property $\mathcal{P}$ is a randomized algorithm which receives query access to an unknown function $f\colon \{0,1\}^n \to \{0,1\}$:

- If $f \in \mathcal{P}$, the algorithm output YES w.p at least $2/3$.
- If algorithm outputs YES w.p at least 1/3, then $d(f, \mathcal{P}) \leq \epsilon$.

$f$ output YES w.p 2/3

$\tilde{f}$ output NO w.p 2/3

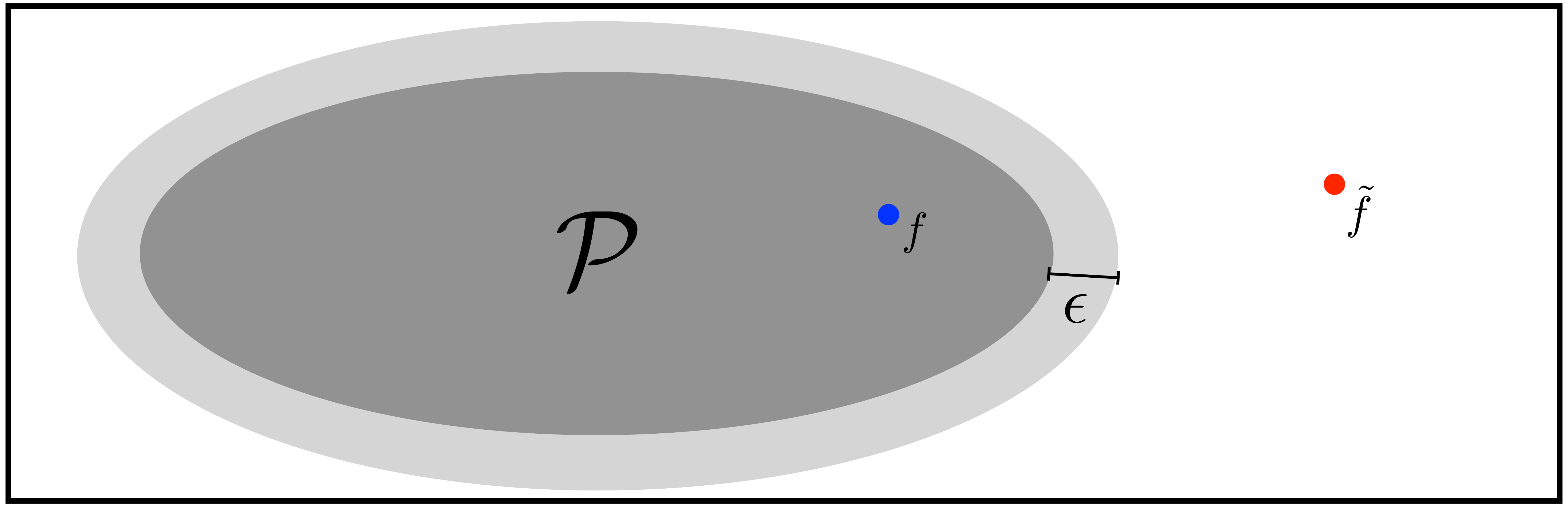


# Property Testing as Approximate Decision Making:

**Definition**: A property testing algorithm for a property $\mathcal{P}$ is a randomized algorithm which receives query access to an unknown function $f\colon \{0,1\}^n \to \{0,1\}$:

- If $f \in \mathcal{P}$, the algorithm output YES w.p at least $2/3$.
- If algorithm outputs YES w.p at least 1/3, then $d(f,\mathcal{P}) \leq \epsilon$.

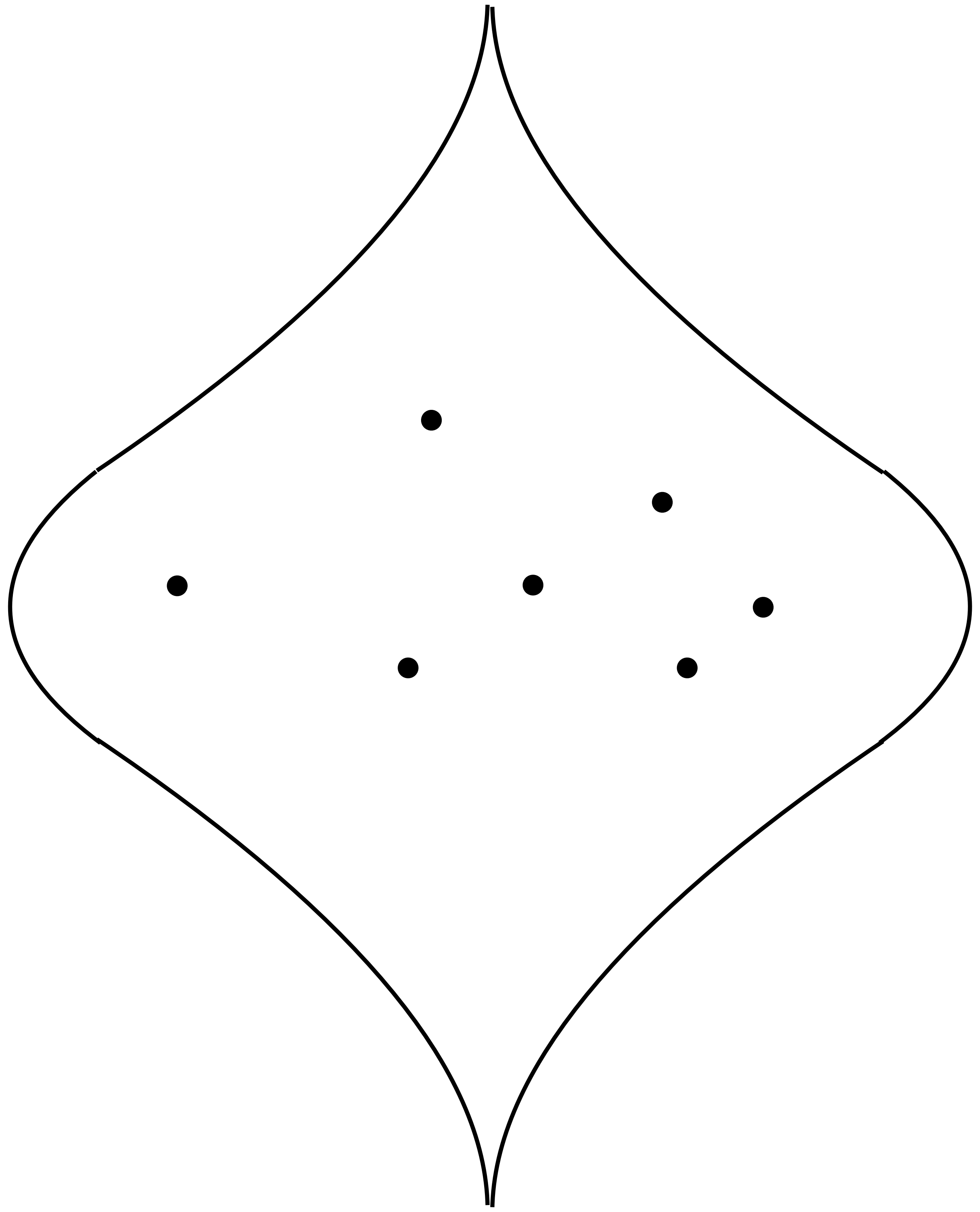
$$d(f,\mathcal{P}) = \min_{g\in\mathcal{P}} \Pr_{\boldsymbol{x}\sim\{0,1\}^n}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$$

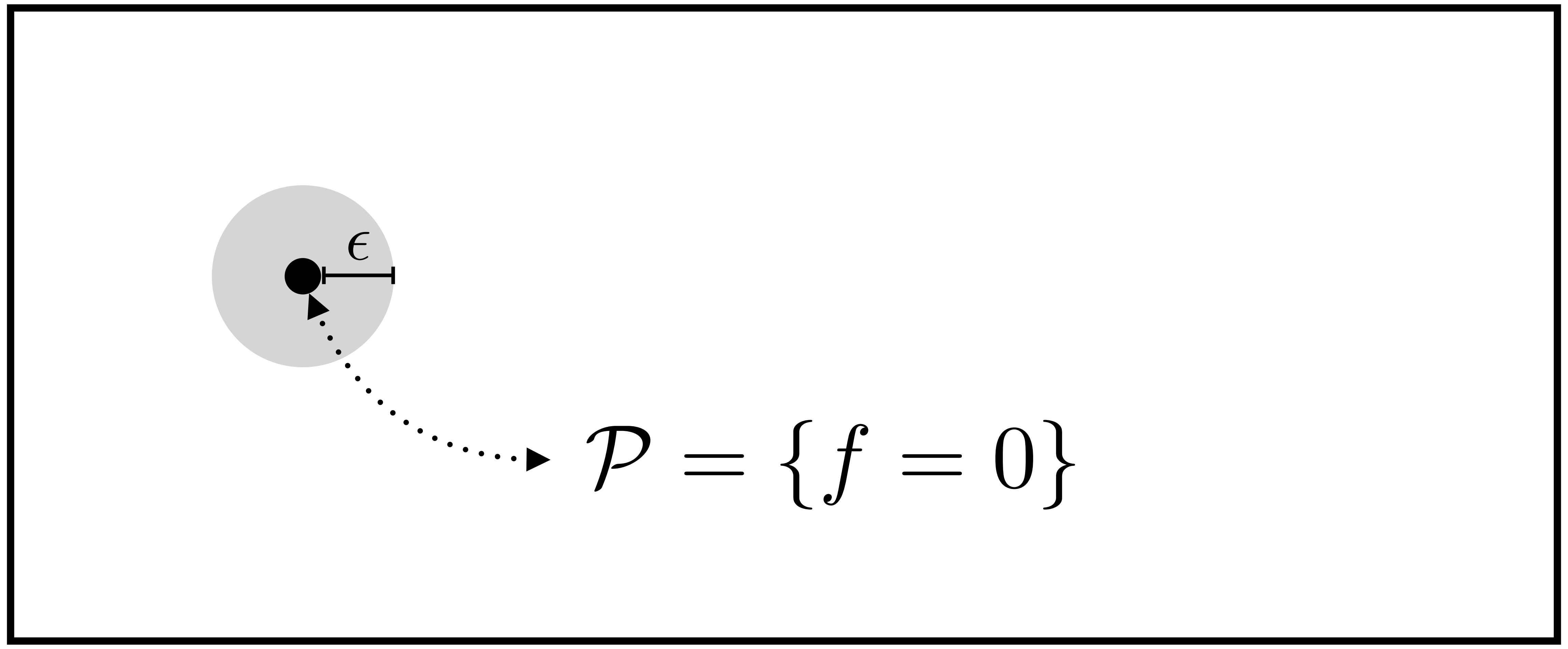


# Formal Analysis for $\mathcal{P} = \{f = 0\}$: Direction 1

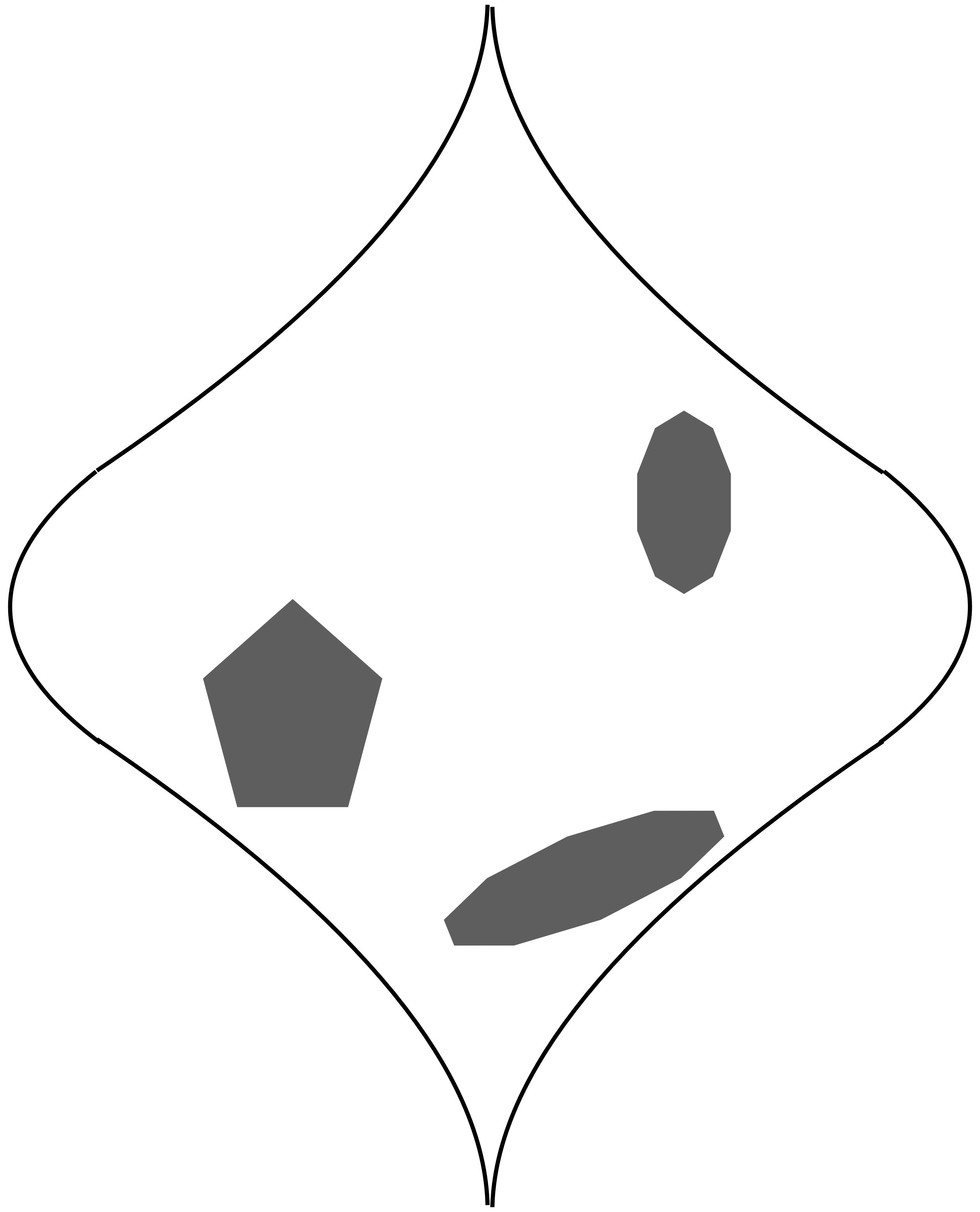
**Algorithm**

1. Sample $q$ inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_q \sim \{0,1\}^n$
2. Query all inputs.
3. Output YES iff always 0.





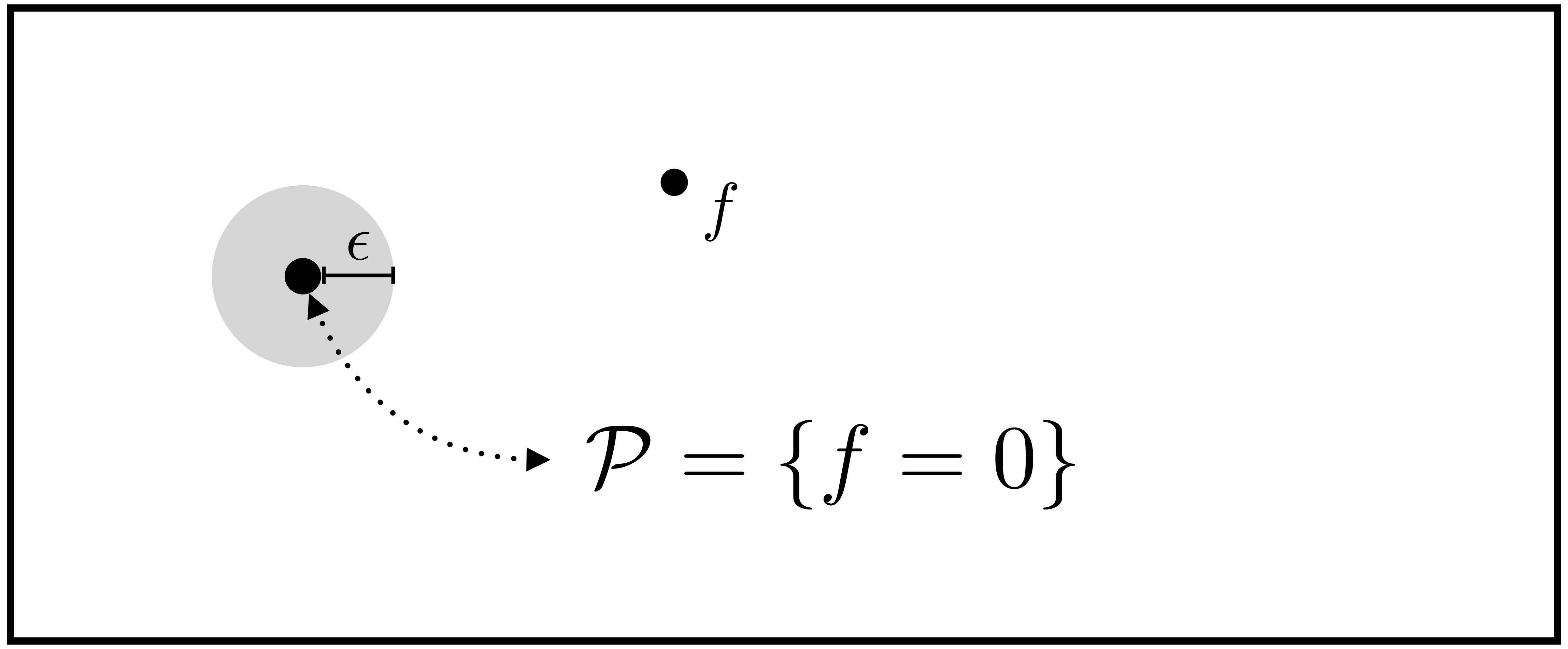
# Formal Analysis for $\mathcal{P} = \{f = 0\}$: Direction 2



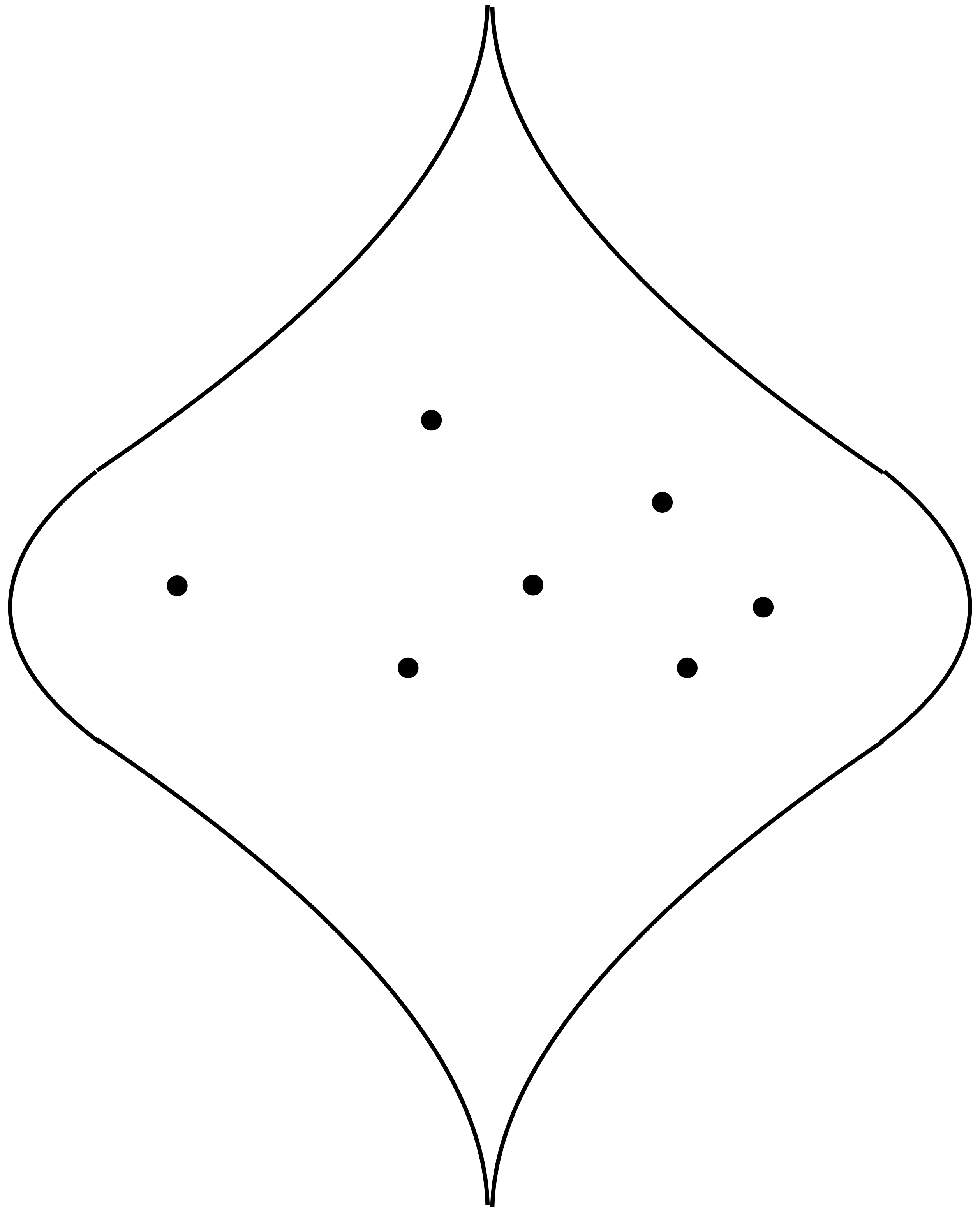
**Algorithm outputs YES:**

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}_1,\dots,\boldsymbol{x}_q}\left[\forall i \in [q] : f(\boldsymbol{x}_i) = 0\right] \leq \left(1 - \frac{\{x : f(x) = 1\}}{2^n}\right)^q$$

$$\frac{\{x : f(x) = 1\}}{2^n} = d(f, \mathcal{P}) \geq \varepsilon.$$

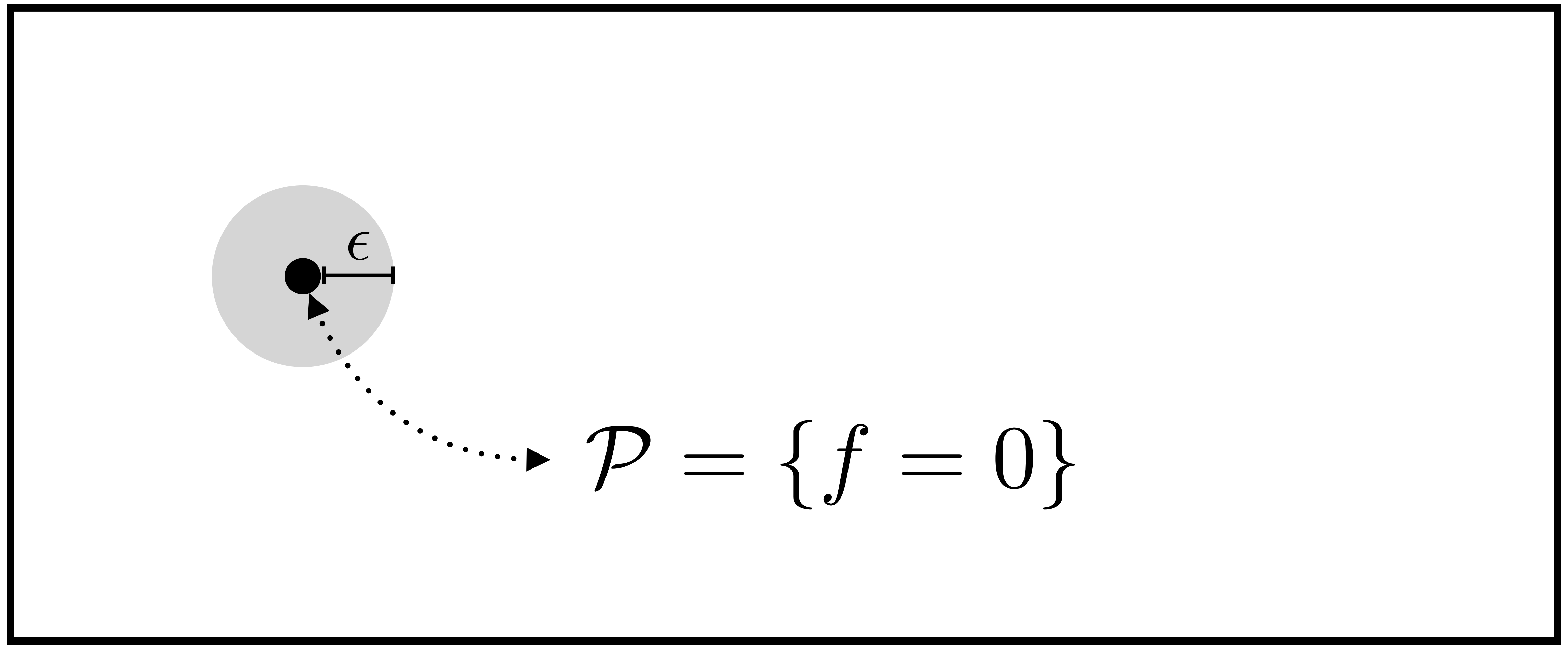


# Is the function always 0?



**Algorithm**

1. Sample $O(1/\varepsilon)$ inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_q \sim \{0,1\}^n$
2. Query all inputs.
3. Output YES iff always 0.




**Additional Properties:**

Non-adaptive and one-sided error

One can always test properties of few functions.

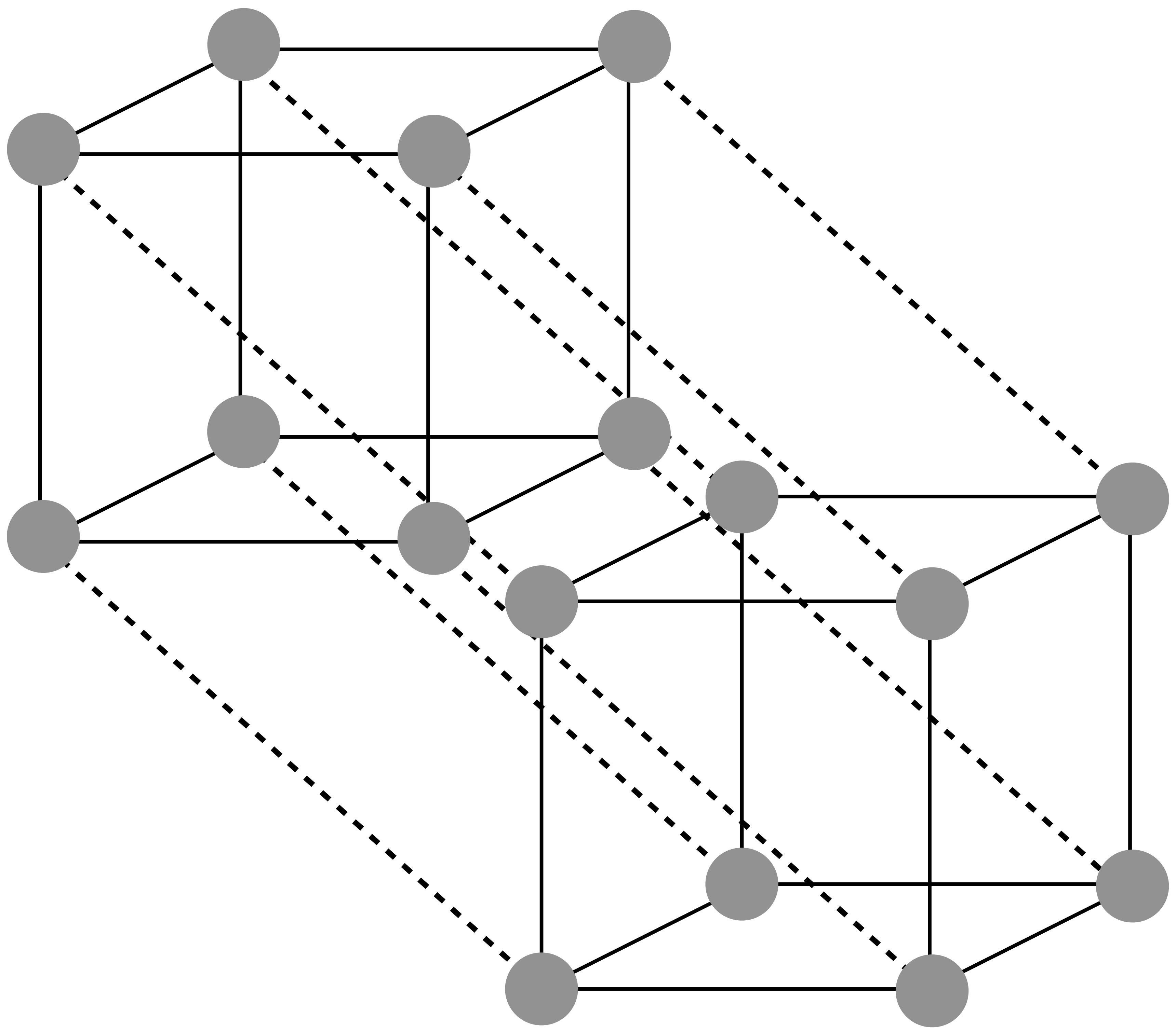

**Observation**: For any property $\mathcal{P}$, there exists an algorithm which tests that property of query complexity $O(\log|\mathcal{P}|/\varepsilon)$.

One can always test properties of few functions.


**Observation**: For any property $\mathcal{P}$, there exists an algorithm which tests that property of query complexity $O(\log|\mathcal{P}|/\varepsilon)$.

* Can be quite bad, number of functions is $2^{2^n}$.
* Actually, *learns* a close function in $\mathcal{P}$.
* Completely generic.

Blum-Luby-Rubinfeld '93: Is my function linear?

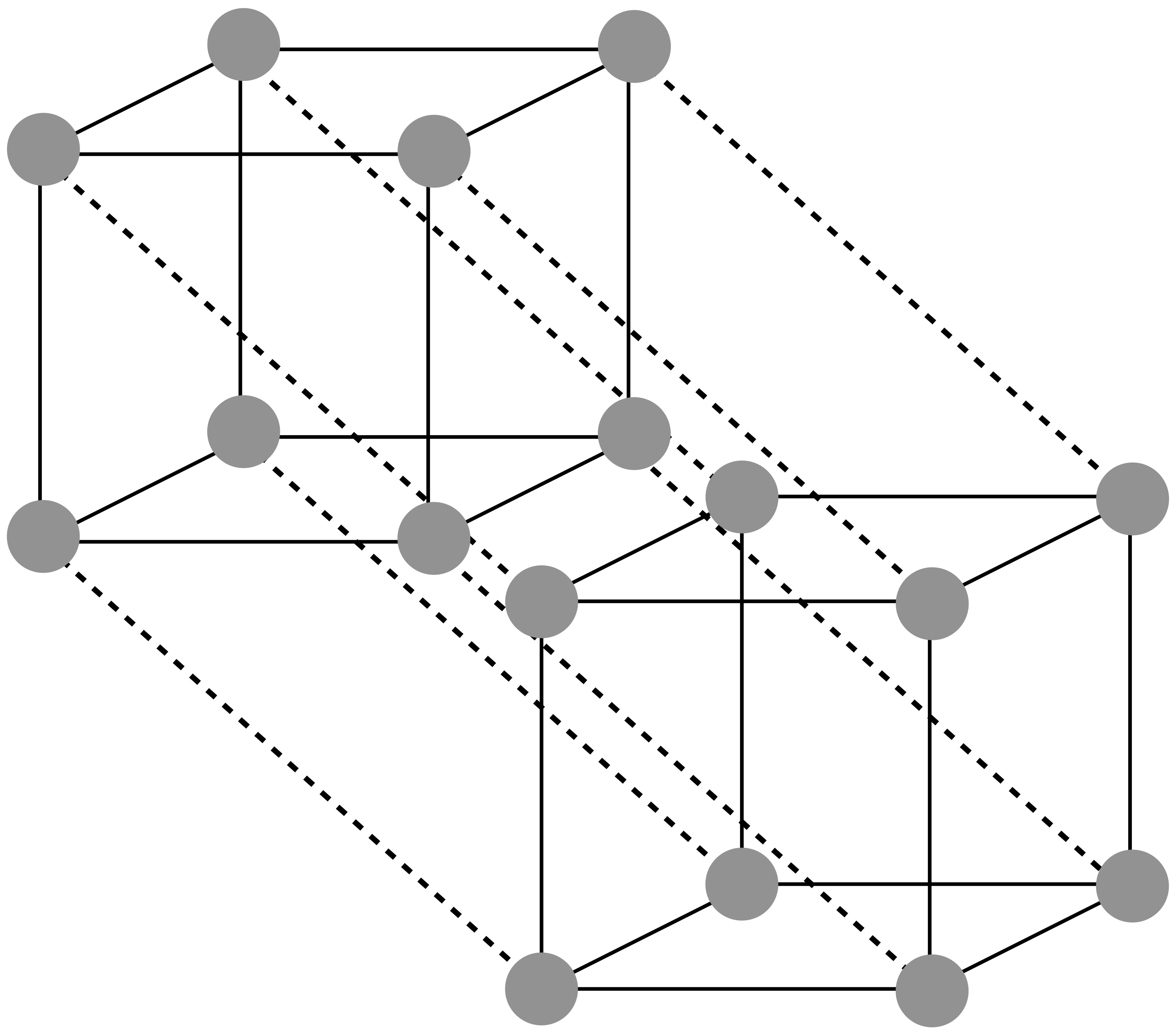


**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) if

$$f(x) = \sum_{i=1}^{n} a_i \cdot x_i \mod 2$$

for some $a \in \{0,1\}^n$.

# Blum-Luby-Rubinfeld '93: Is my function linear?



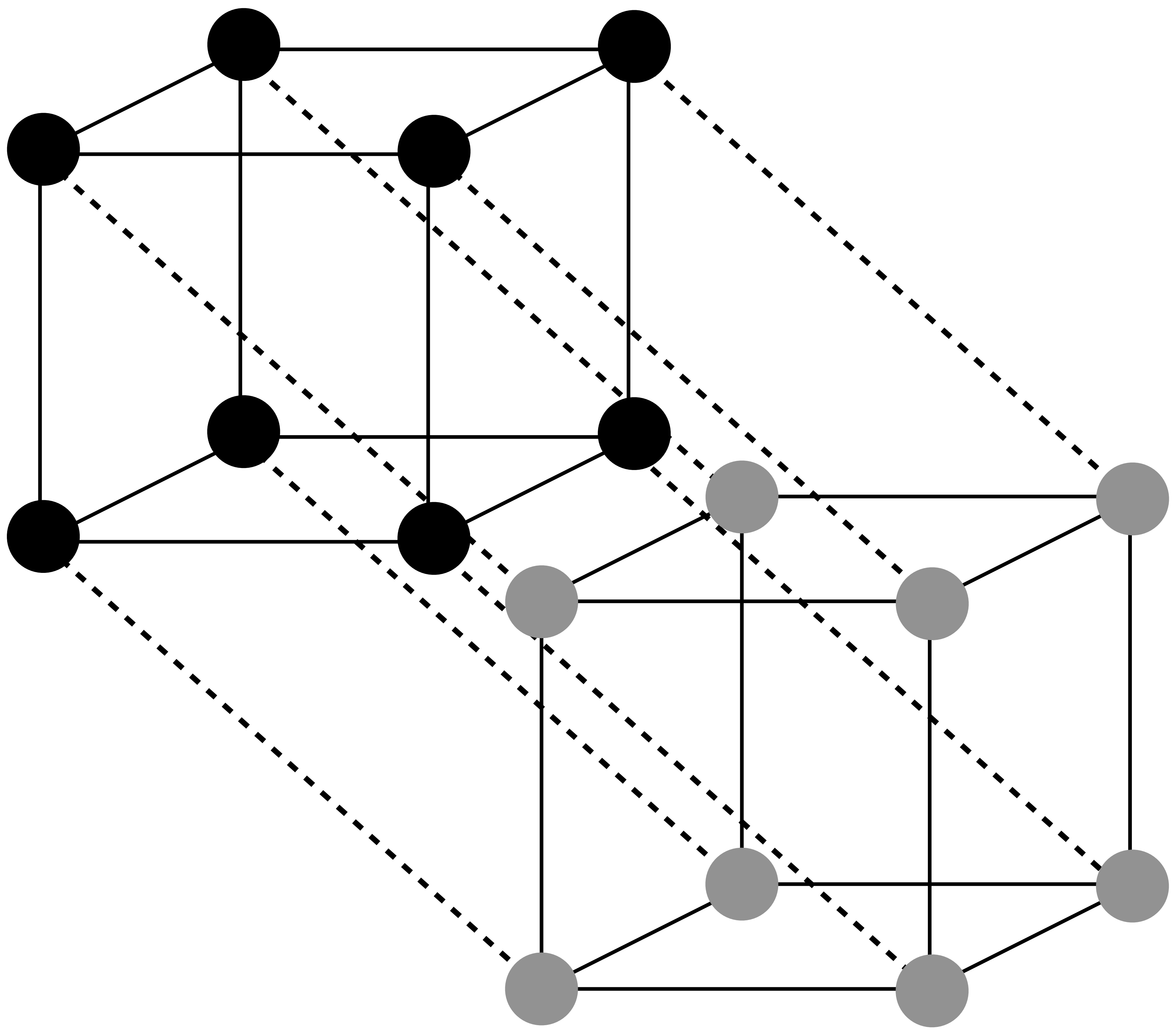
**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) if

$$f(x) = \sum_{i=1}^{n} a_i \cdot x_i \mod 2$$

for some $a \in \{0,1\}^n$.

Examples: $a = (0,0,0,0) \to f = 0$

# Blum-Luby-Rubinfeld '93: Is my function linear?



**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) if

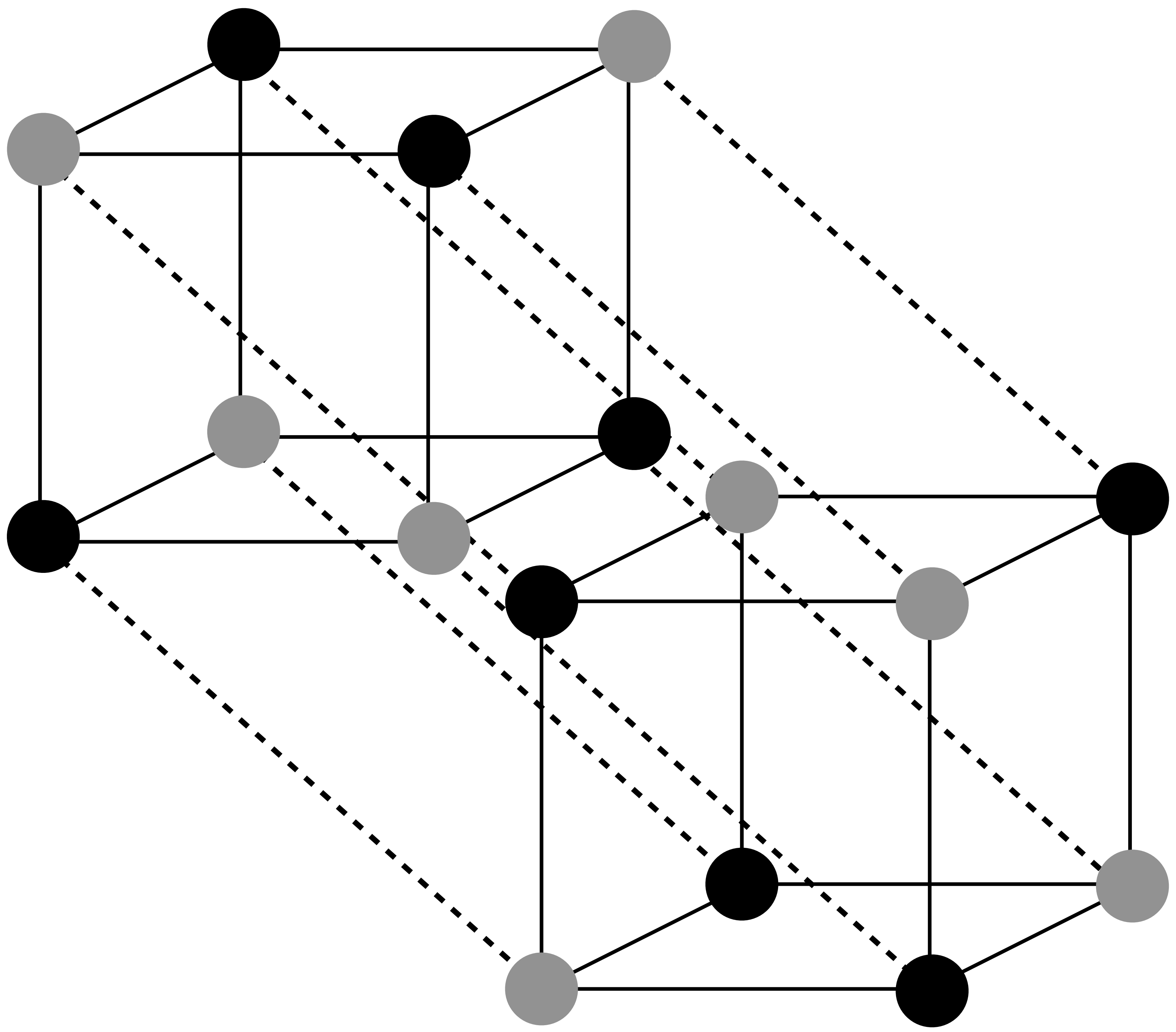
$$f(x) = \sum_{i=1}^{n} a_i \cdot x_i \mod 2$$

for some $a \in \{0,1\}^n$.

Examples: $a = (0,0,0,0) \to f = 0$

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

# Blum-Luby-Rubinfeld '93: Is my function linear?



**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) if

$$f(x) = \sum_{i=1}^{n} a_i \cdot x_i \mod 2$$

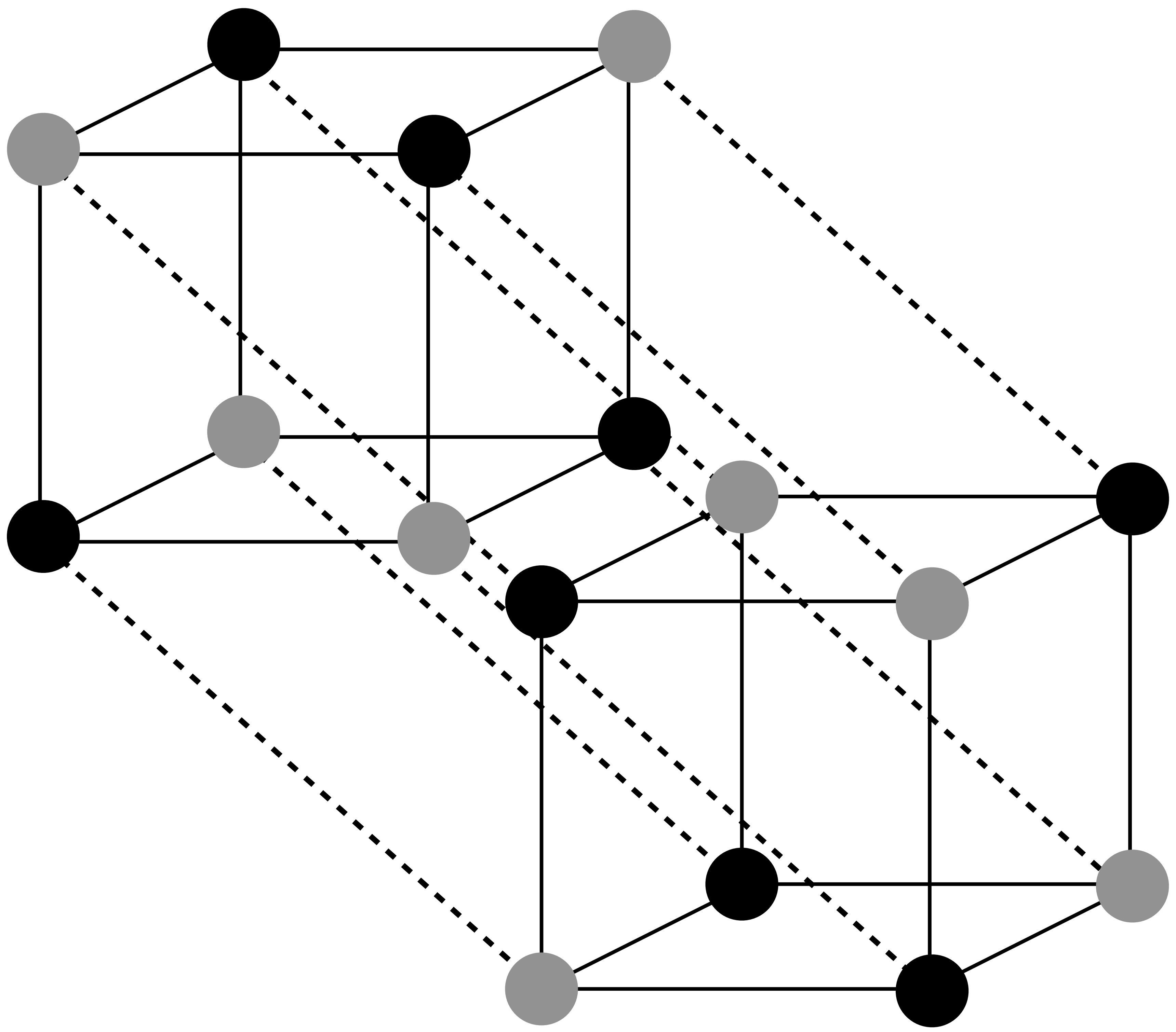
for some $a \in \{0,1\}^n$.

Examples: $a = (0,0,0,0) \to f = 0$

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

$a = (1,1,1,1)$

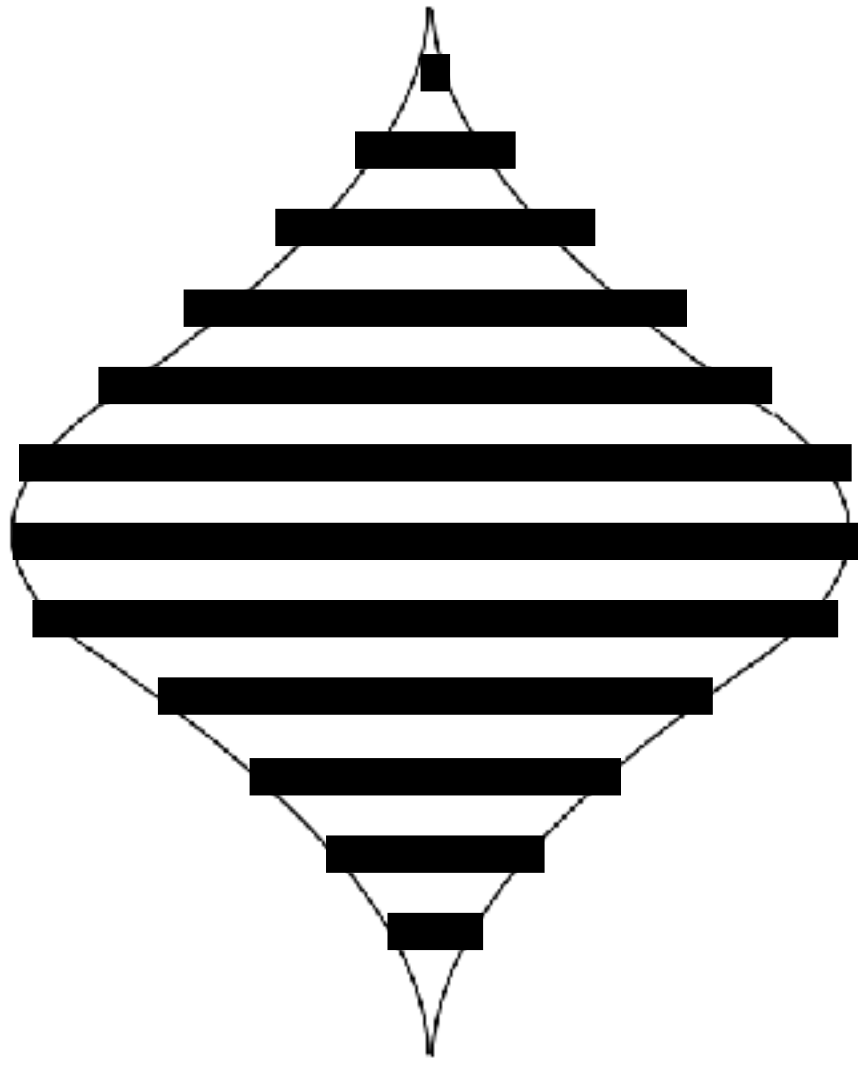
# Blum-Luby-Rubinfeld '93: Is my function linear?



**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) if

$$f(x) = \sum_{i=1}^{n} a_i \cdot x_i \mod 2$$

for some $a \in \{0,1\}^n$.

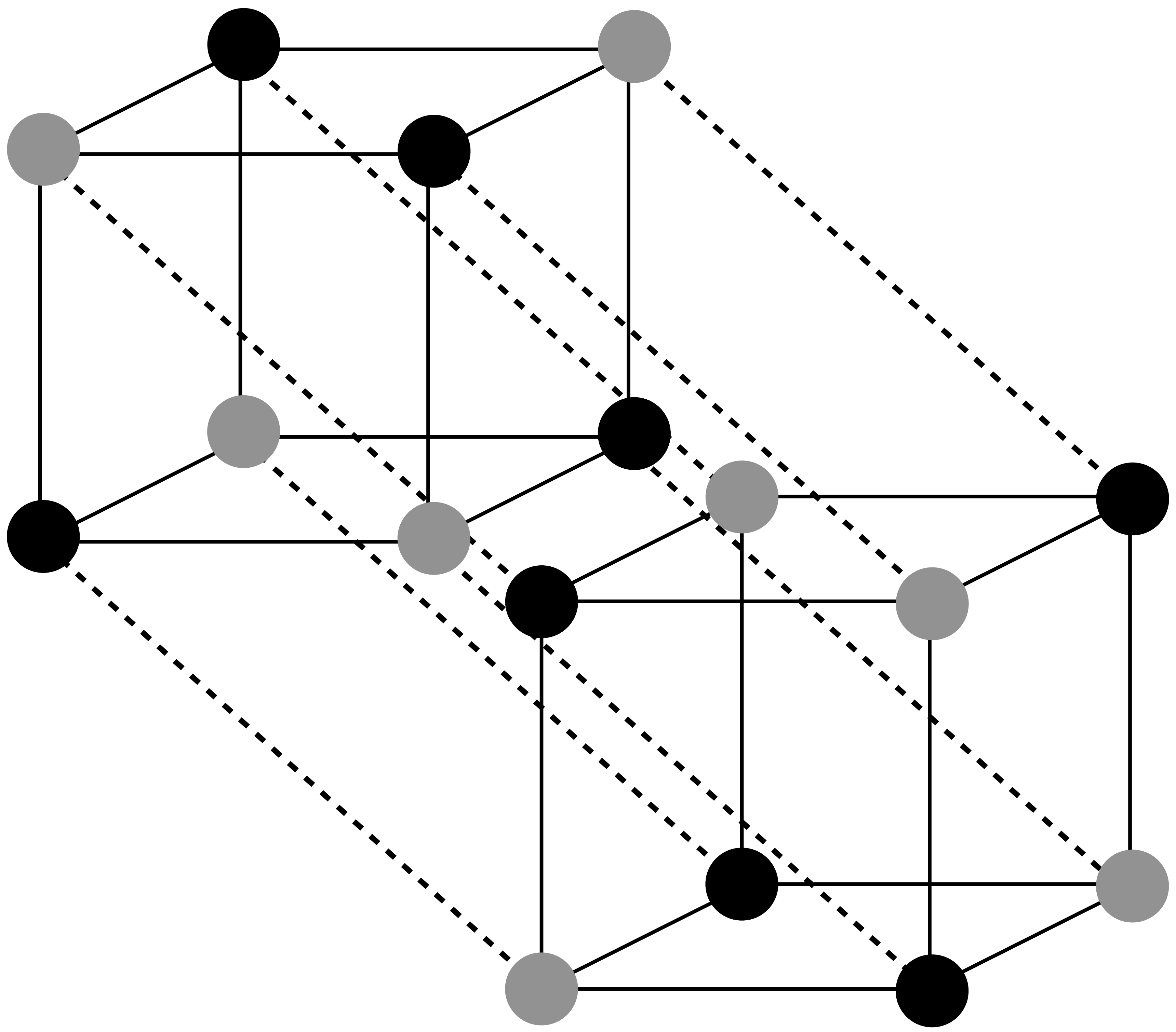


Examples: $a = (0,0,0,0) \to f = 0$

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

$a = (1,1,1,1)$

# Blum-Luby-Rubinfeld '93: Is my function linear?



**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) if

$$f(x) = \sum_{i=1}^{n} a_i \cdot x_i \mod 2$$

for some $a \in \{0,1\}^n$.

Examples: $a = (0,0,0,0) \to f = 0$
$a = (0,0,0,1)$
$a = (1,1,1,1)$

Naive Benchmark: $|\mathcal{P}| = 2^n$ $O(n/\varepsilon)$

# Blum-Luby-Rubinfeld '93: Is my function linear?

**Definition**: a function $f \colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) if

$$f(x) = \sum_{i=1}^{n} a_i \cdot x_i \mod 2$$

for some $a \in \{0,1\}^n$.

Slight improvement:

1. Query $f(e_i) = \hat{a}_i, \forall i$
2. Check if

$$f \stackrel{?}{=} \sum_{i=1}^{n} \hat{a}_i \cdot x_i$$

by querying $O(1/\epsilon)$ random inputs.

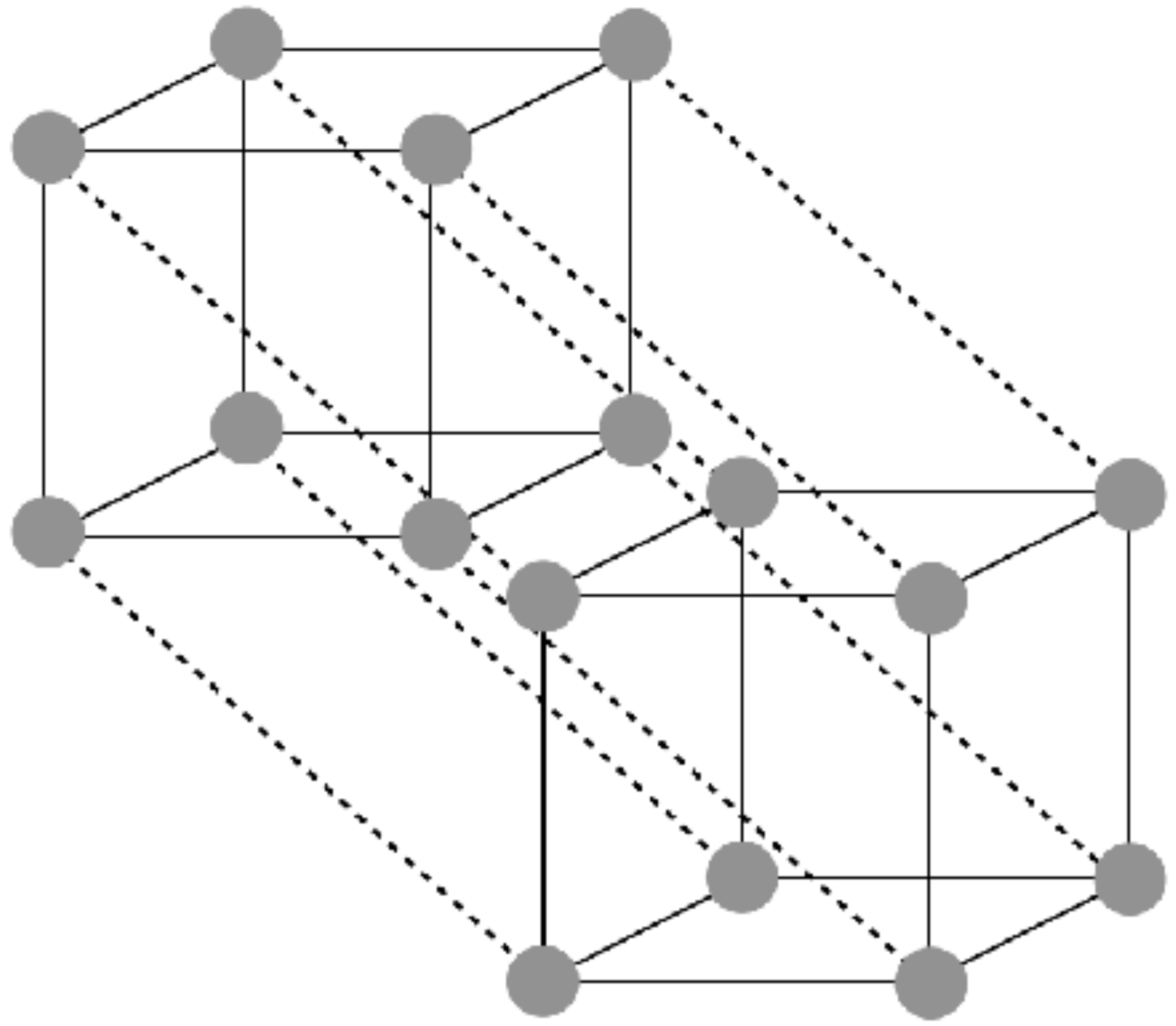
$n + O(1/\epsilon)$

Naive Benchmark: $|\mathcal{P}| = 2^n$ $O(n/\varepsilon)$

A different definition of linearity and a natural consistency test.

**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

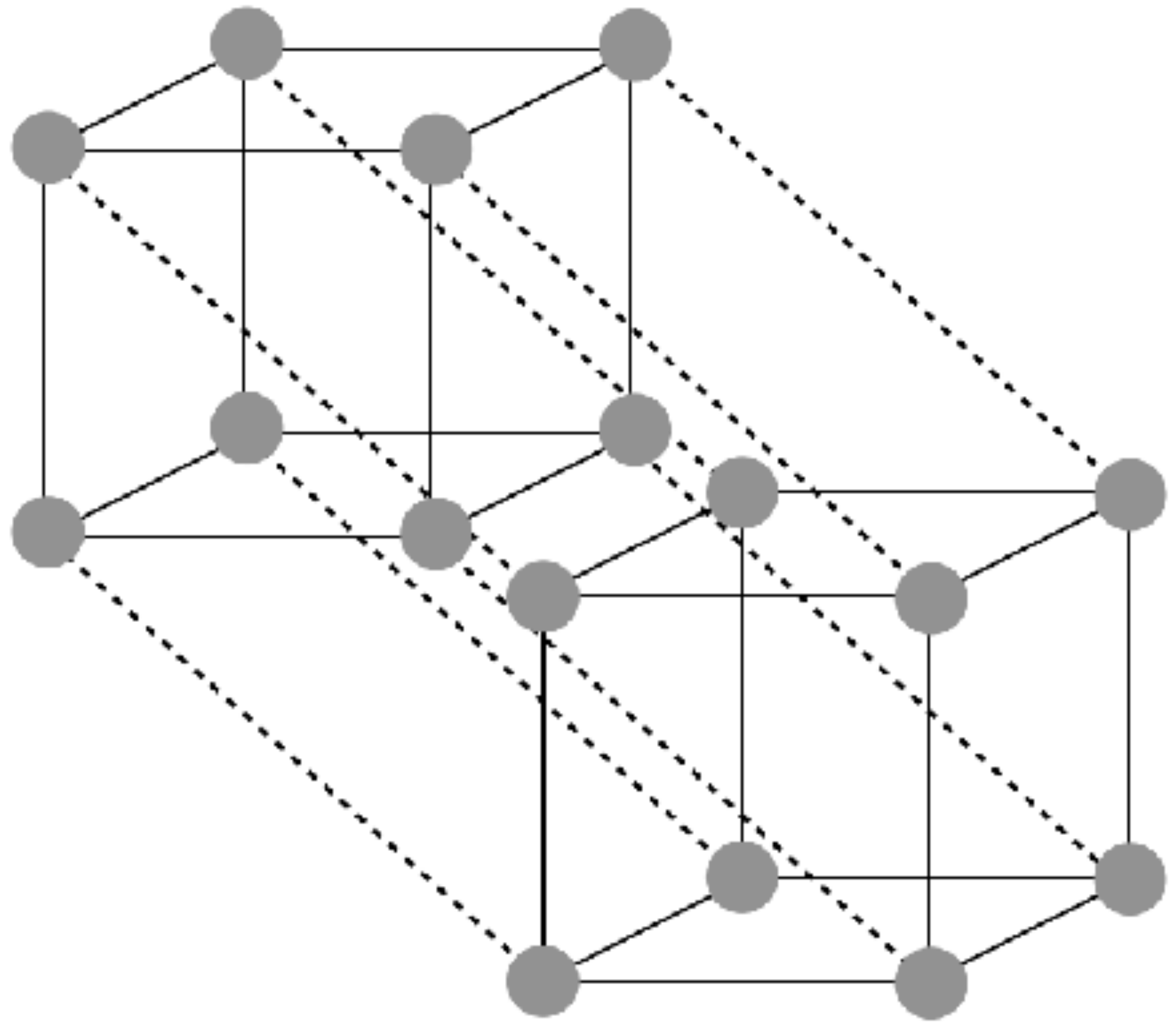
$$f(x) + f(y) = f(x+y) \qquad \forall x, y \in \mathbb{F}_2$$



# A different definition of linearity and a natural consistency test.

**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

$$f(x) + f(y) = f(x+y) \qquad \forall x, y \in \mathbb{F}_2$$



**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.

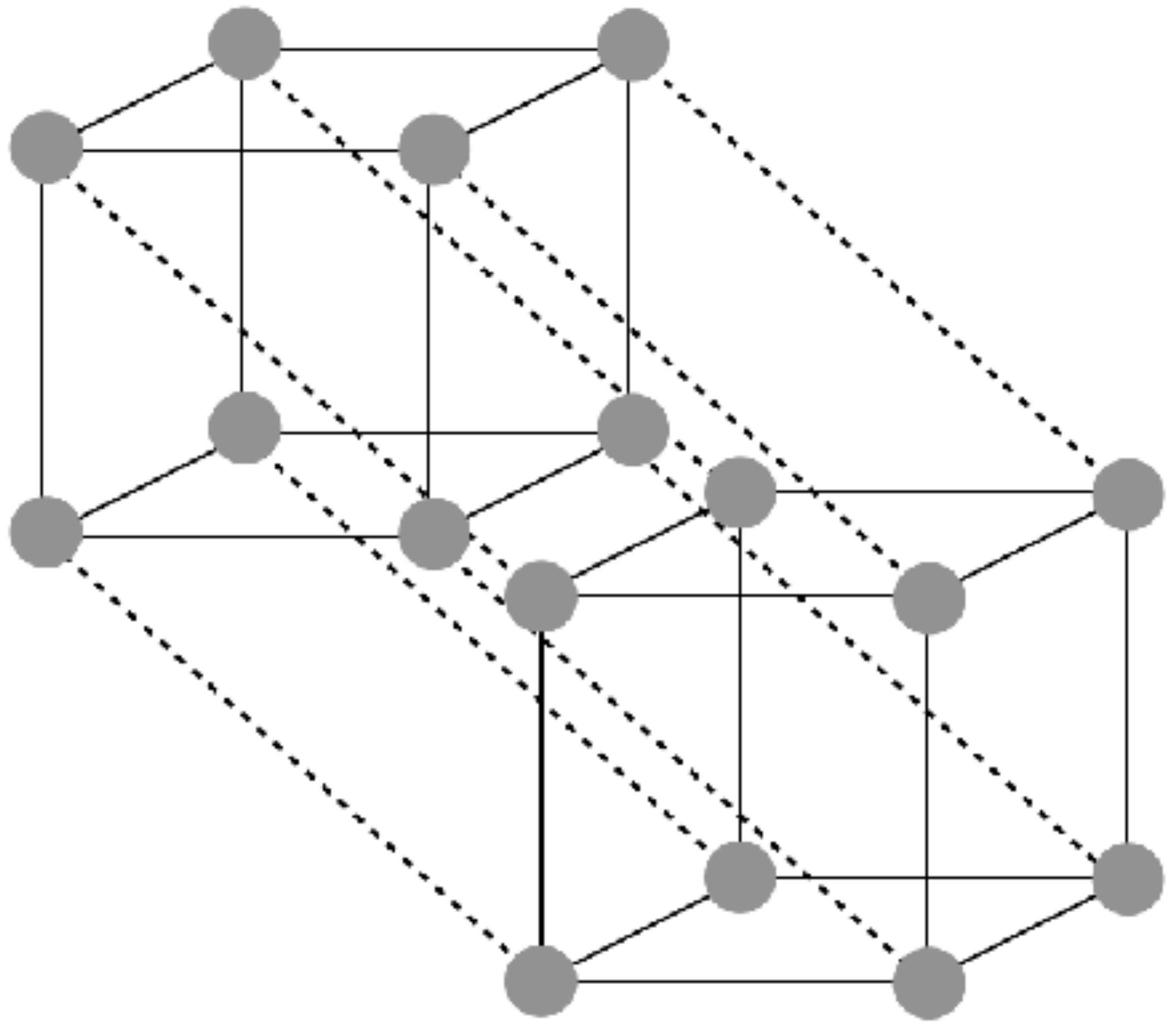
# A different definition of linearity and a natural consistency test.

**Definition:** a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

$$f(x) + f(y) = f(x+y) \qquad \forall x, y \in \mathbb{F}_2$$

**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.



A low-query algorithm cannot check $\forall x, y$. Can "closeness" be derived from probabilistic guarantee?

# What could possibly go wrong?

**Definition:** a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

$$f(x) + f(y) = f(x+y) \qquad \forall x, y \in \mathbb{F}_2$$

**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.

# What could possibly go wrong?

**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

$$f(x) + f(y) = f(x+y) \qquad \forall x, y \in \mathbb{F}_2$$

**Natural Algorithm (Check Consistency)**

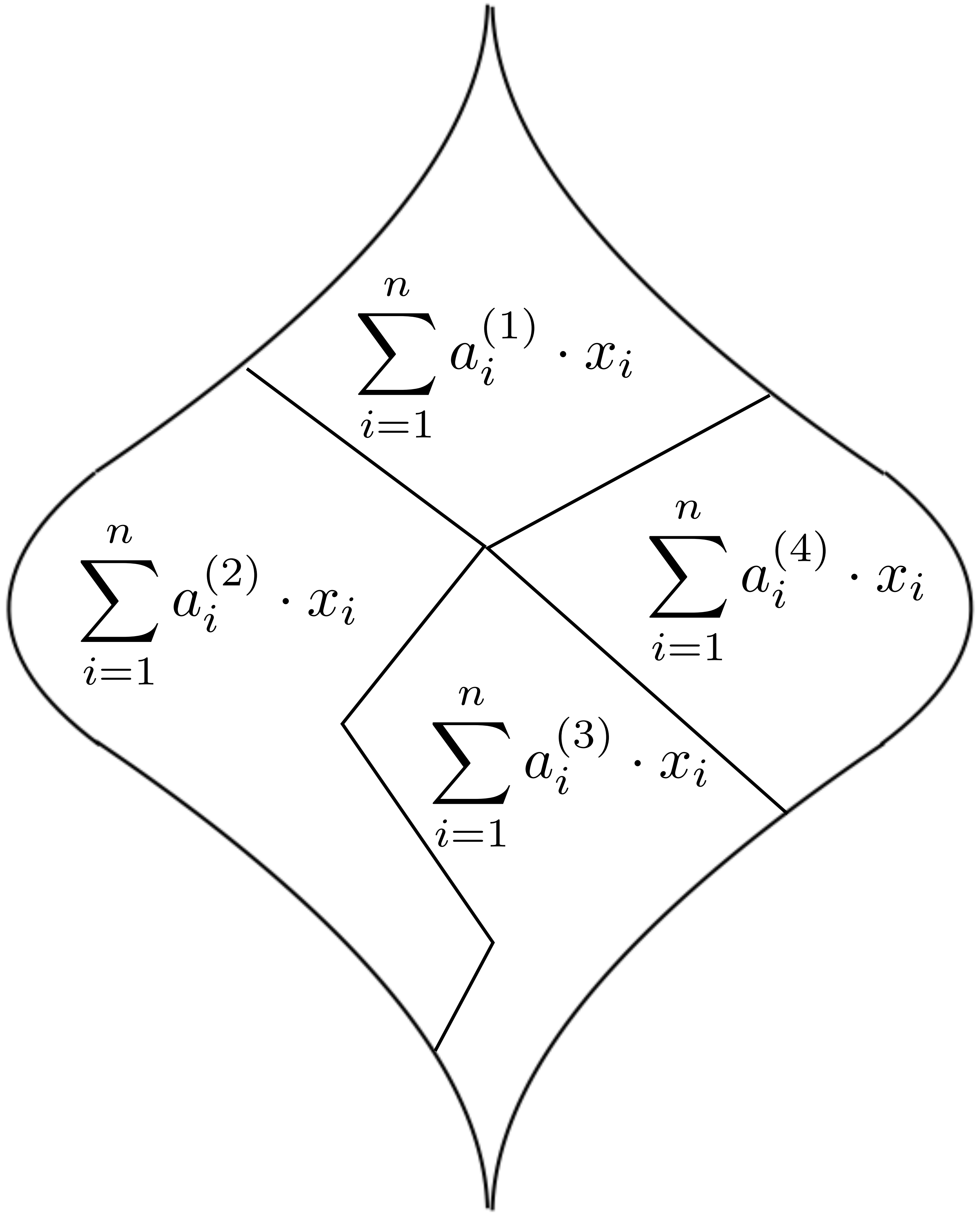
1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.

- Bad Case 1: *Portions* of function are linear for *different* reasons.
- Bad Case 2: Even though far, *evidence* is hard to find.

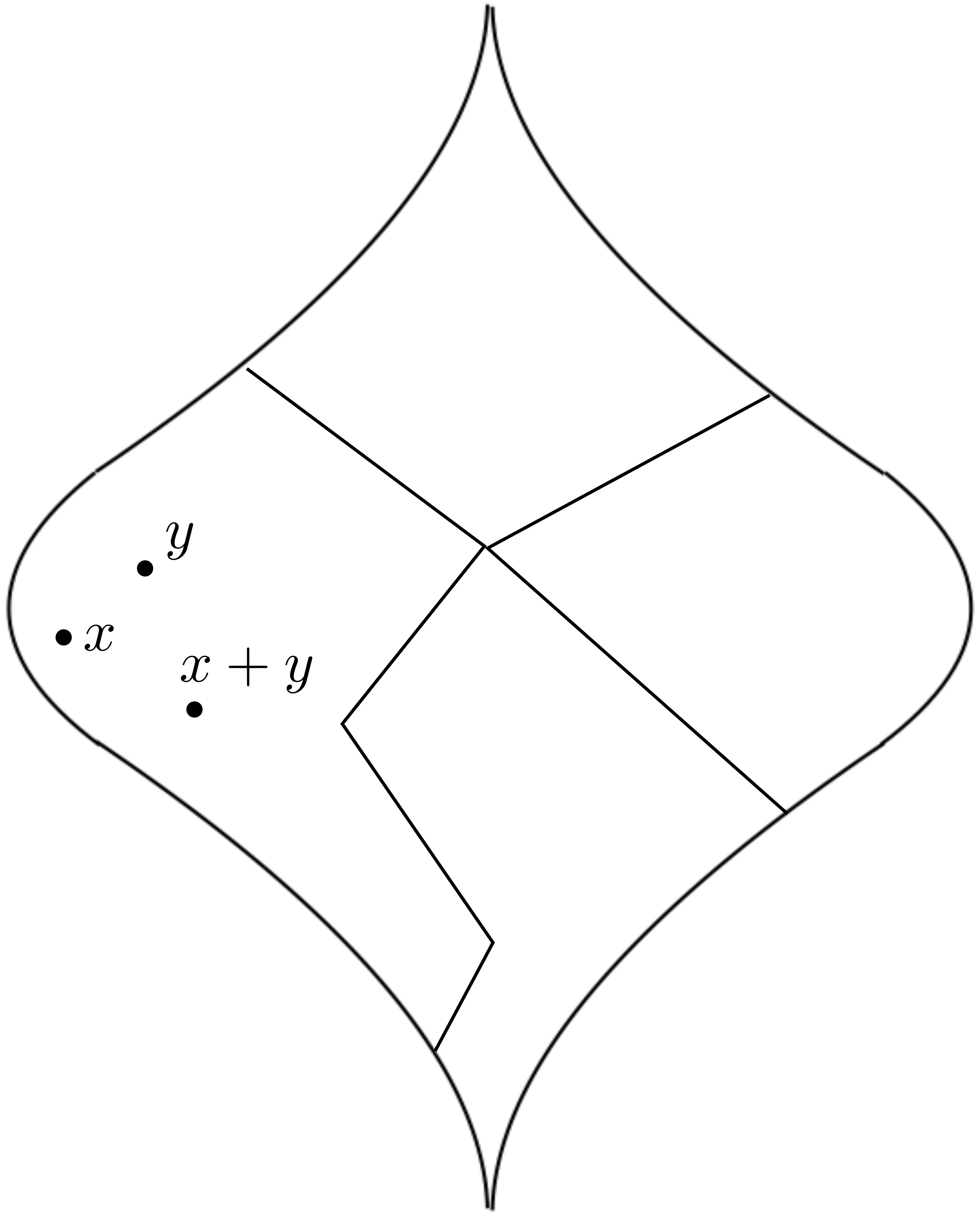
# Bad Case 1: *Portions* of function are linear for *different* reasons.

**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.



# Bad Case 1: *Portions* of function are linear for *different* reasons.



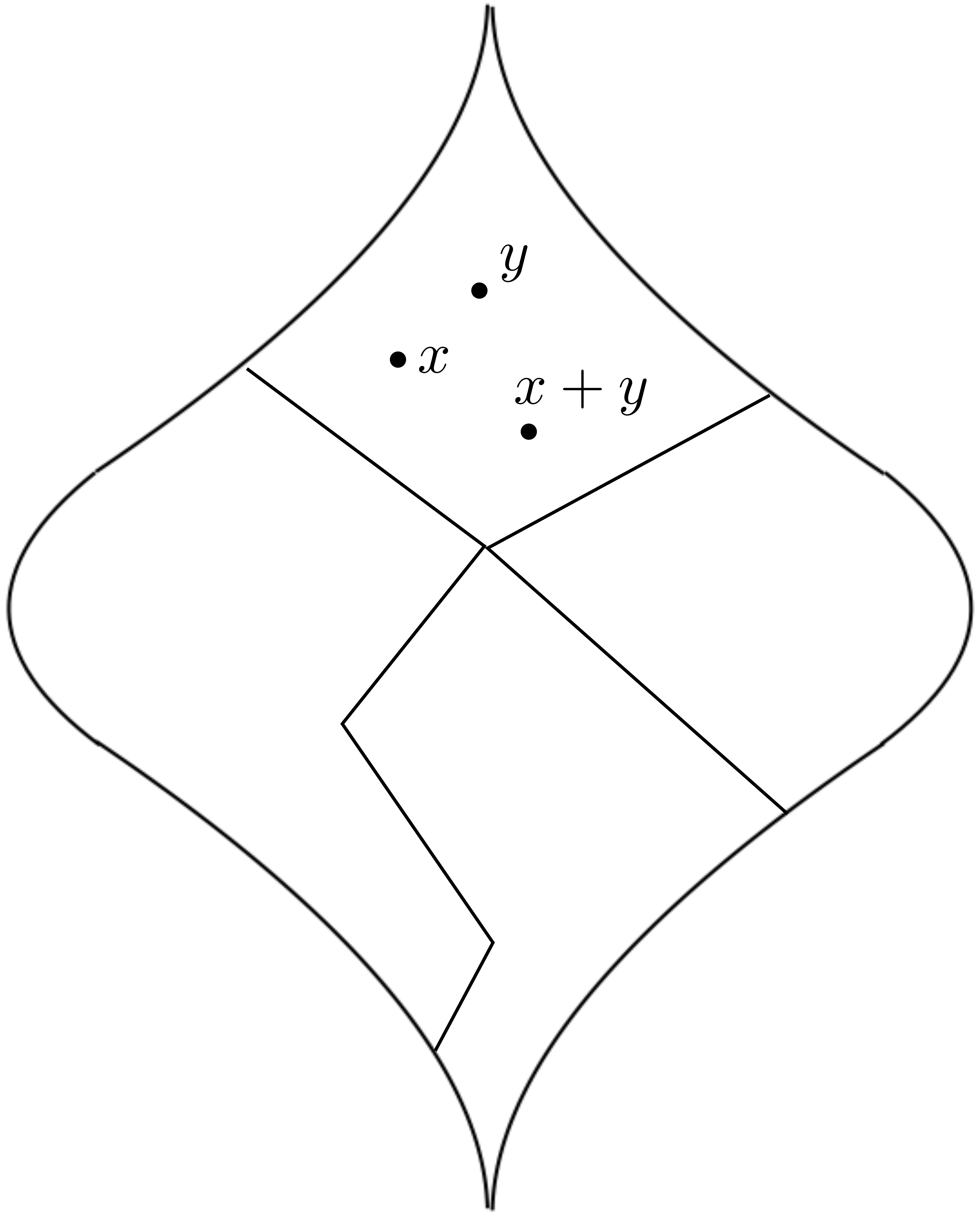

**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.

# Bad Case 1: *Portions* of function are linear for *different* reasons.

**Natural Algorithm (Check Consistency)**

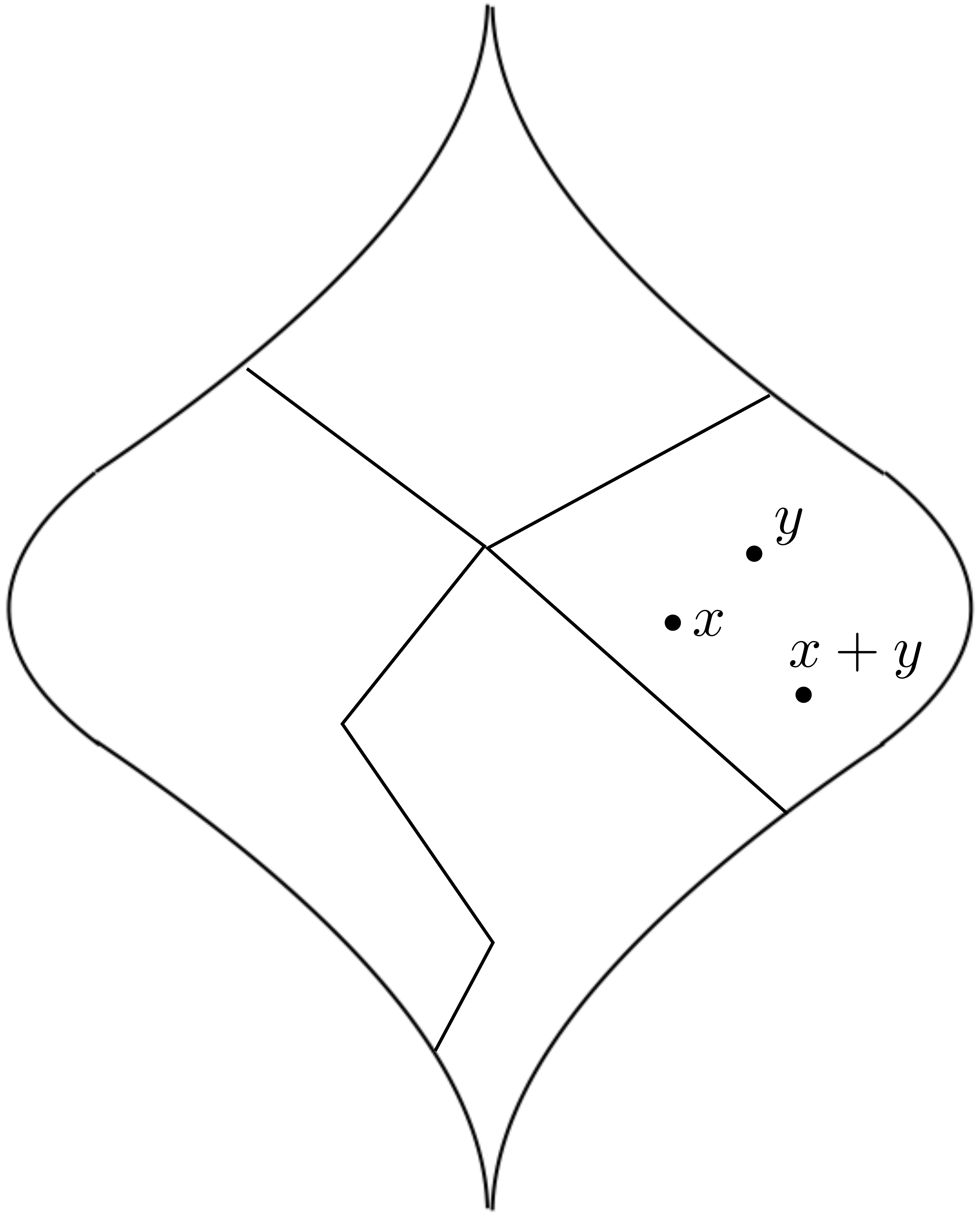
1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.



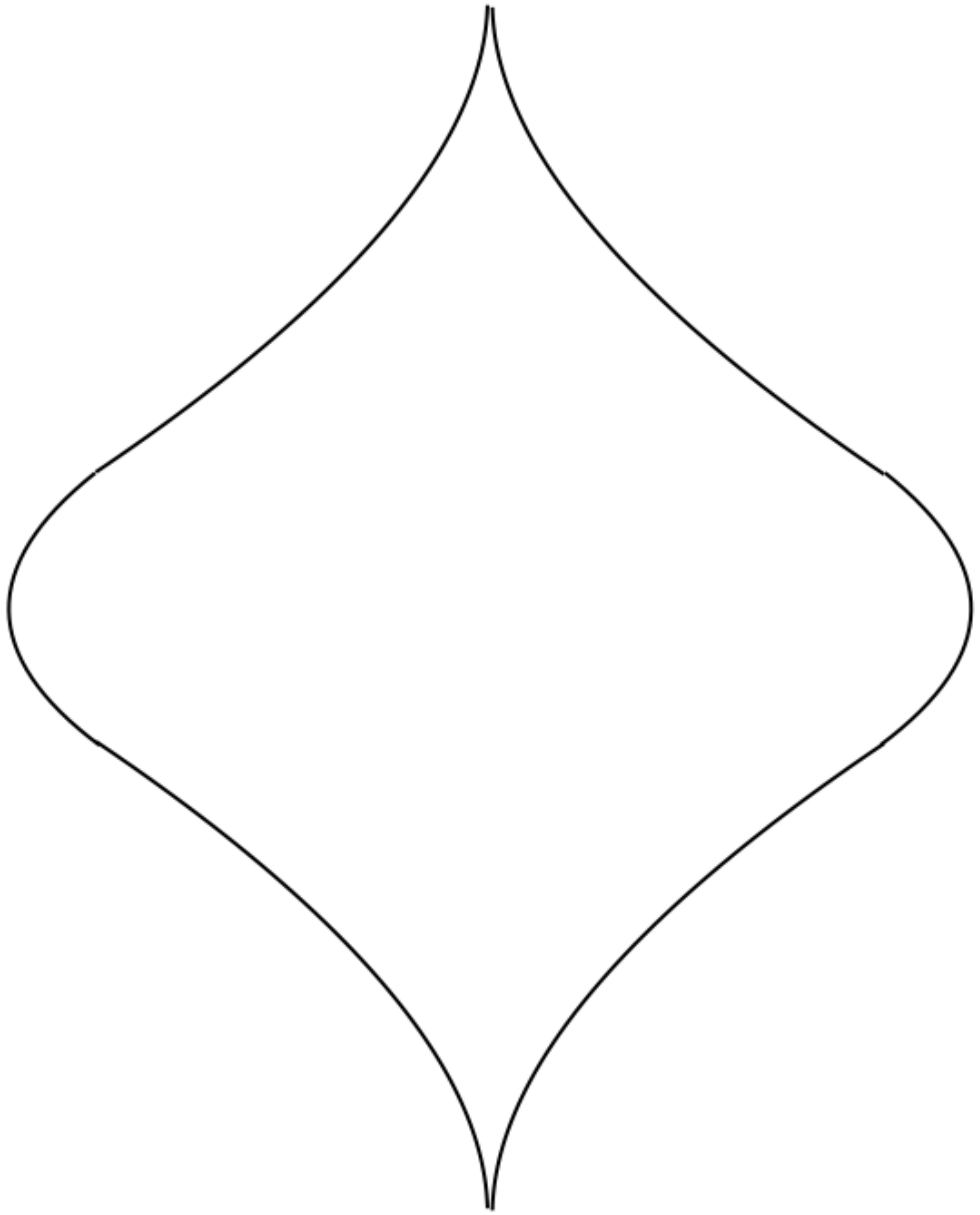
# Bad Case 1: *Portions* of function are linear for *different* reasons.

**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.



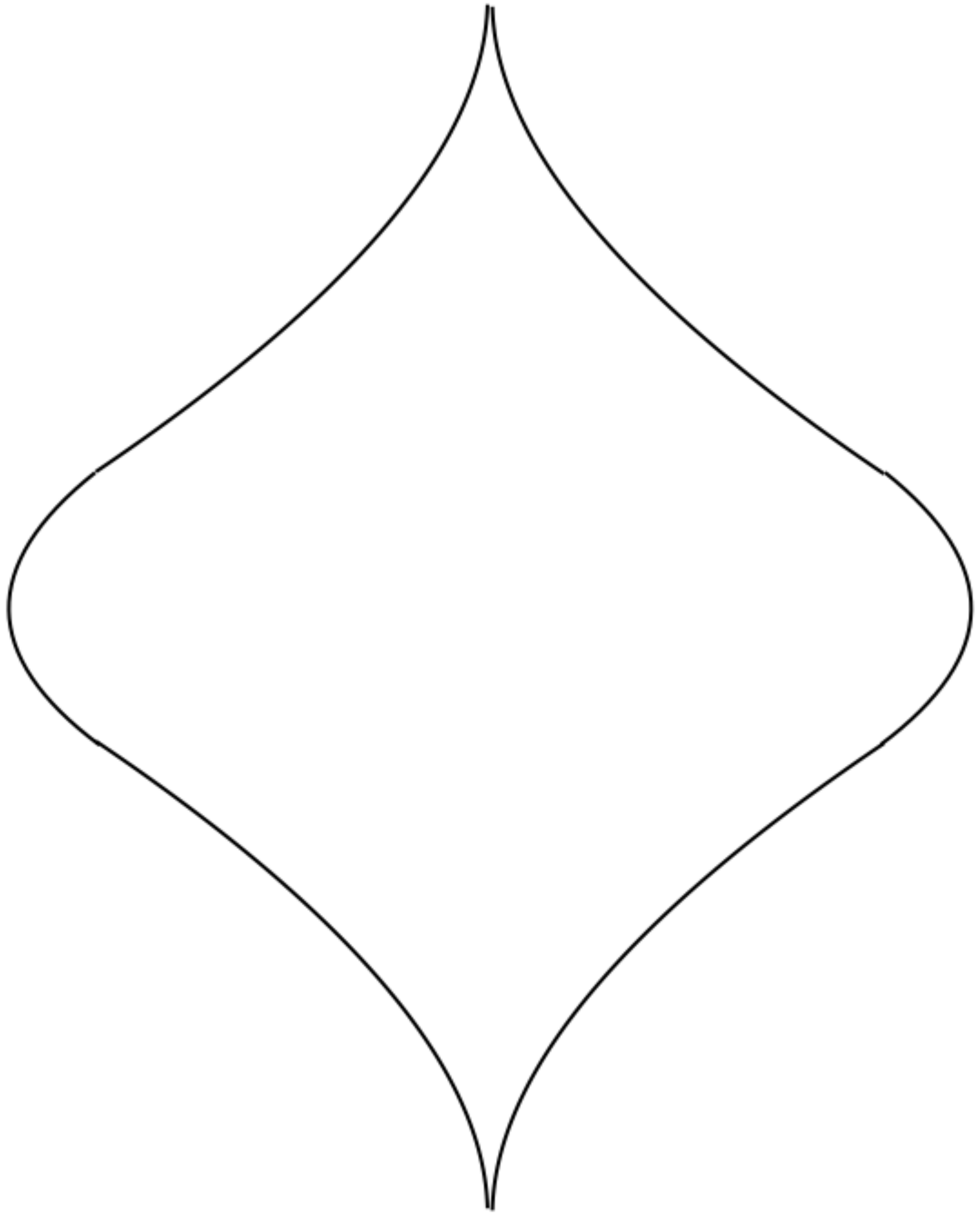
# Bad Case 2: Even though $\epsilon$-far from linear, hard to find evidence.



**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.

# Bad Case 2: Even though $\epsilon$-far from linear, hard to find evidence.



**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.

**Theorem [BLR'93]:** Test works with query complexity $O(1/\epsilon)$

# Bad Case 2: Even though $\epsilon$-far from linear, hard to find evidence.

**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

$$f(x + e_i) = f(x) + f(e_i) \qquad \forall x \in \{0,1\}^n, i \in [n]$$

# Bad Case 2: Even though $\epsilon$-far from linear, hard to find evidence.

**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

$$f(x+e_i) = f(x) + f(e_i) \qquad \forall x \in \{0,1\}^n, i \in [n]$$

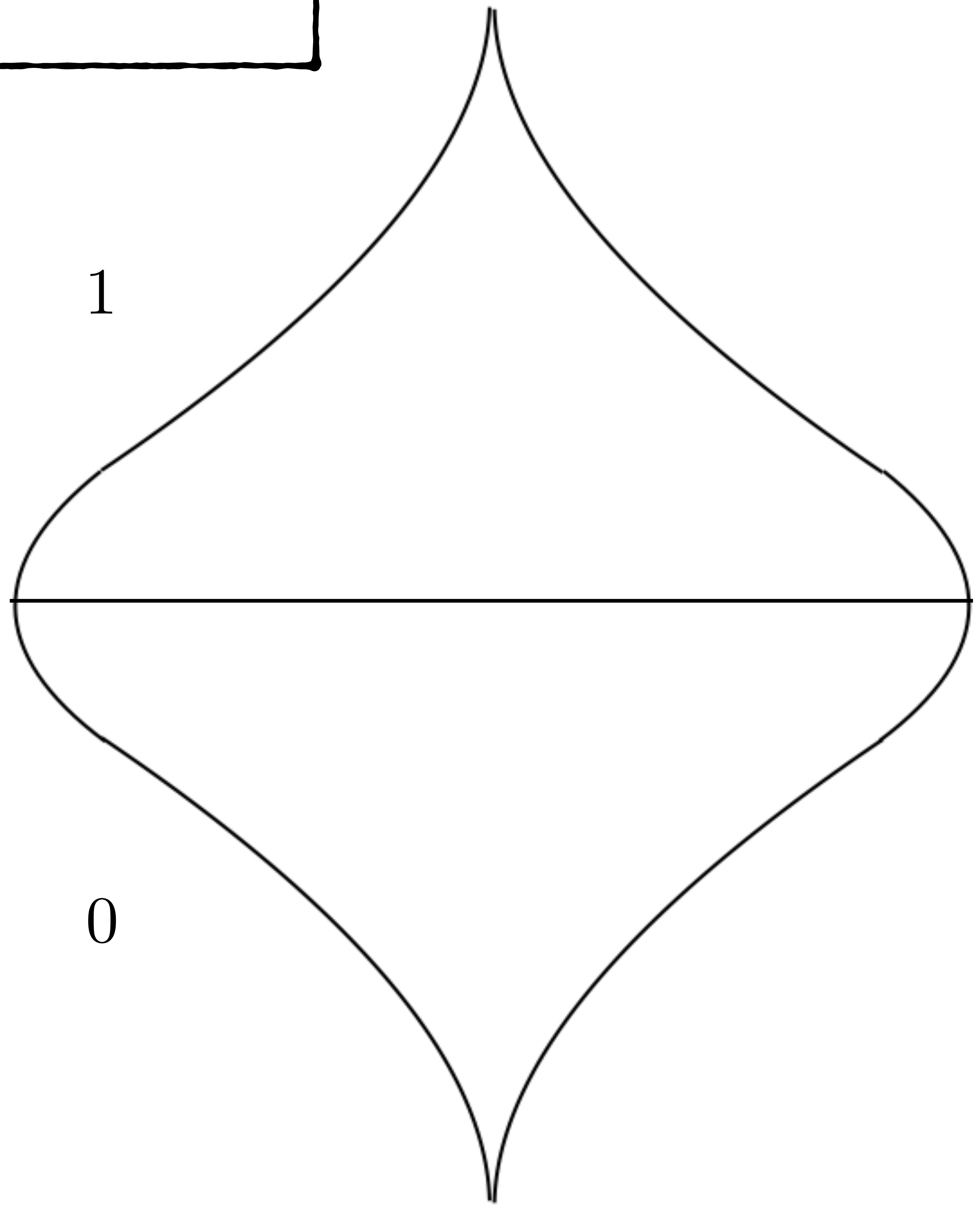
**Natural Algorithm (Check Consistency)**

1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}) + f(e_{\boldsymbol{i}}) = f(\boldsymbol{x} + e_{\boldsymbol{i}})$
3. Output YES iff always pass check.

# Bad Case 2: Even though $\epsilon$-far from linear, hard to find evidence.

**Definition:** a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

$$f(x+e_i) = f(x) + f(e_i) \qquad \forall x \in \{0,1\}^n, i \in [n]$$



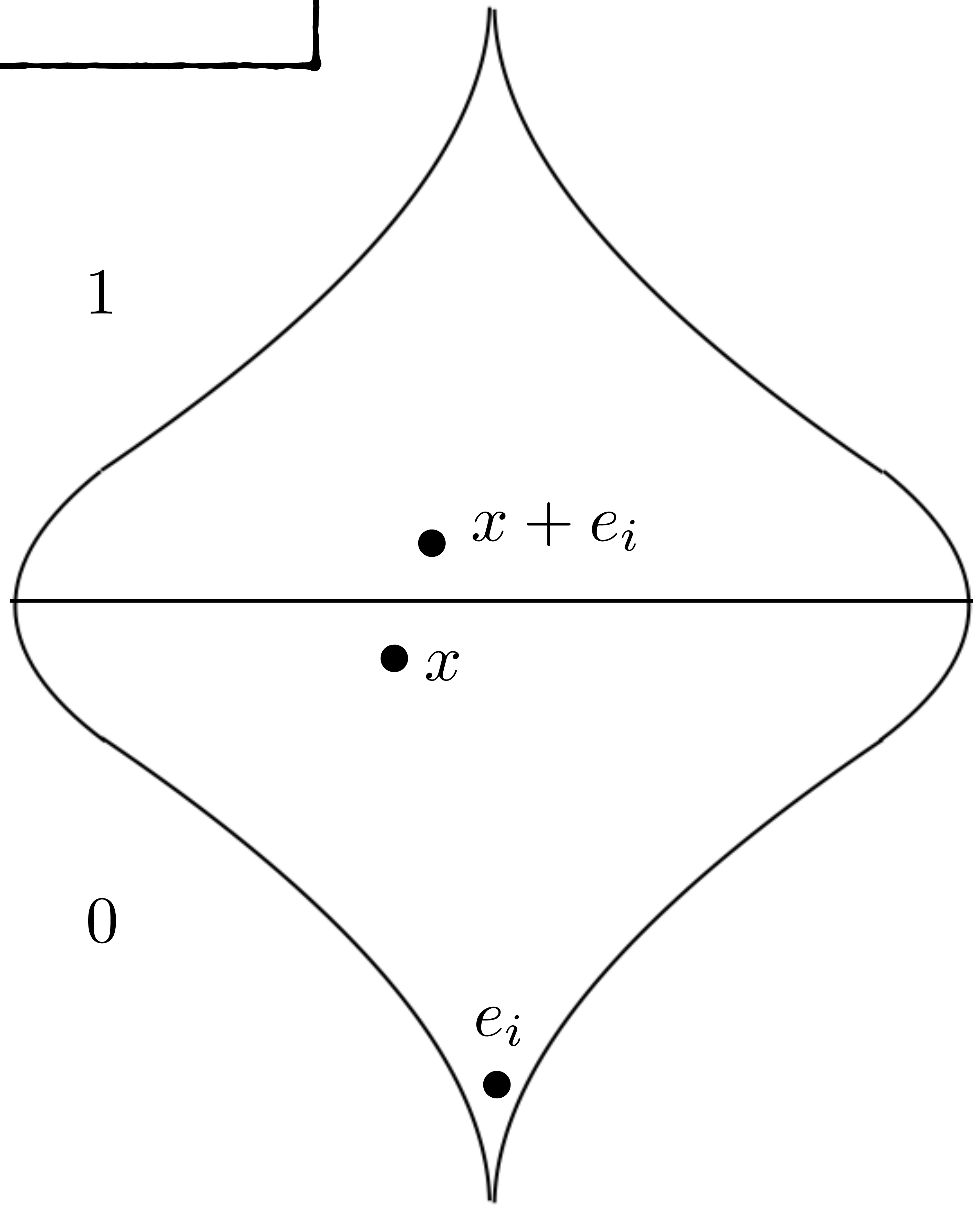

**Natural Algorithm (Check Consistency)**

1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}) + f(e_{\boldsymbol{i}}) = f(\boldsymbol{x} + e_{\boldsymbol{i}})$
3. Output YES iff always pass check.

# Bad Case 2: Even though $\epsilon$-far from linear, hard to find evidence.

**Definition:** a function $f \colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

$$f(x + e_i) = f(x) + f(e_i) \qquad \forall x \in \{0,1\}^n, i \in [n]$$



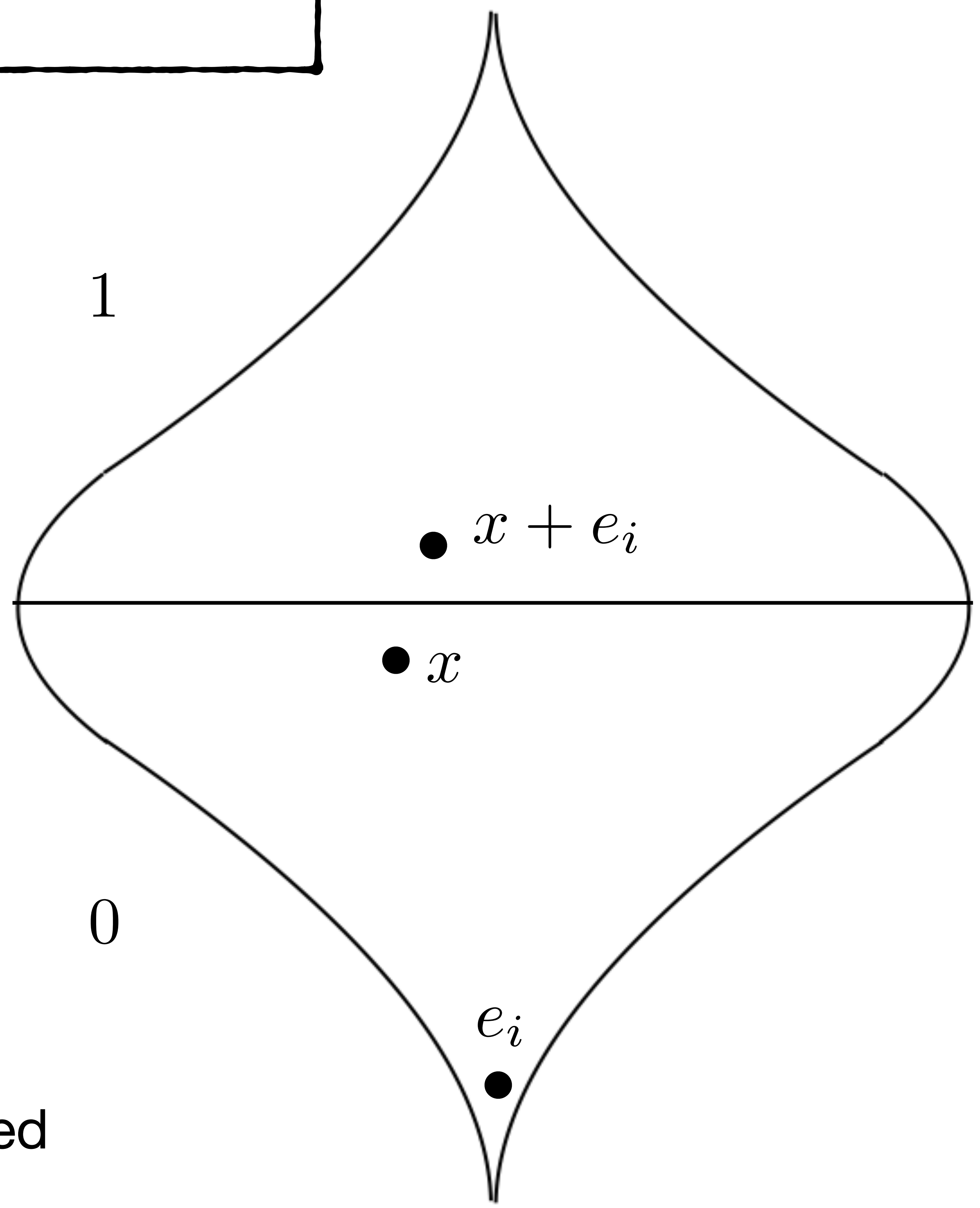

**Natural Algorithm (Check Consistency)**

1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}) + f(e_{\boldsymbol{i}}) = f(\boldsymbol{x} + e_{\boldsymbol{i}})$
3. Output YES iff always pass check.

# Bad Case 2: Even though $\epsilon$-far from linear, hard to find evidence.

**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is linear (in $\mathbb{F}_2$) iff

$$f(x+e_i) = f(x) + f(e_i) \qquad \forall x \in \{0,1\}^n, i \in [n]$$




**Natural Algorithm (Check Consistency)**

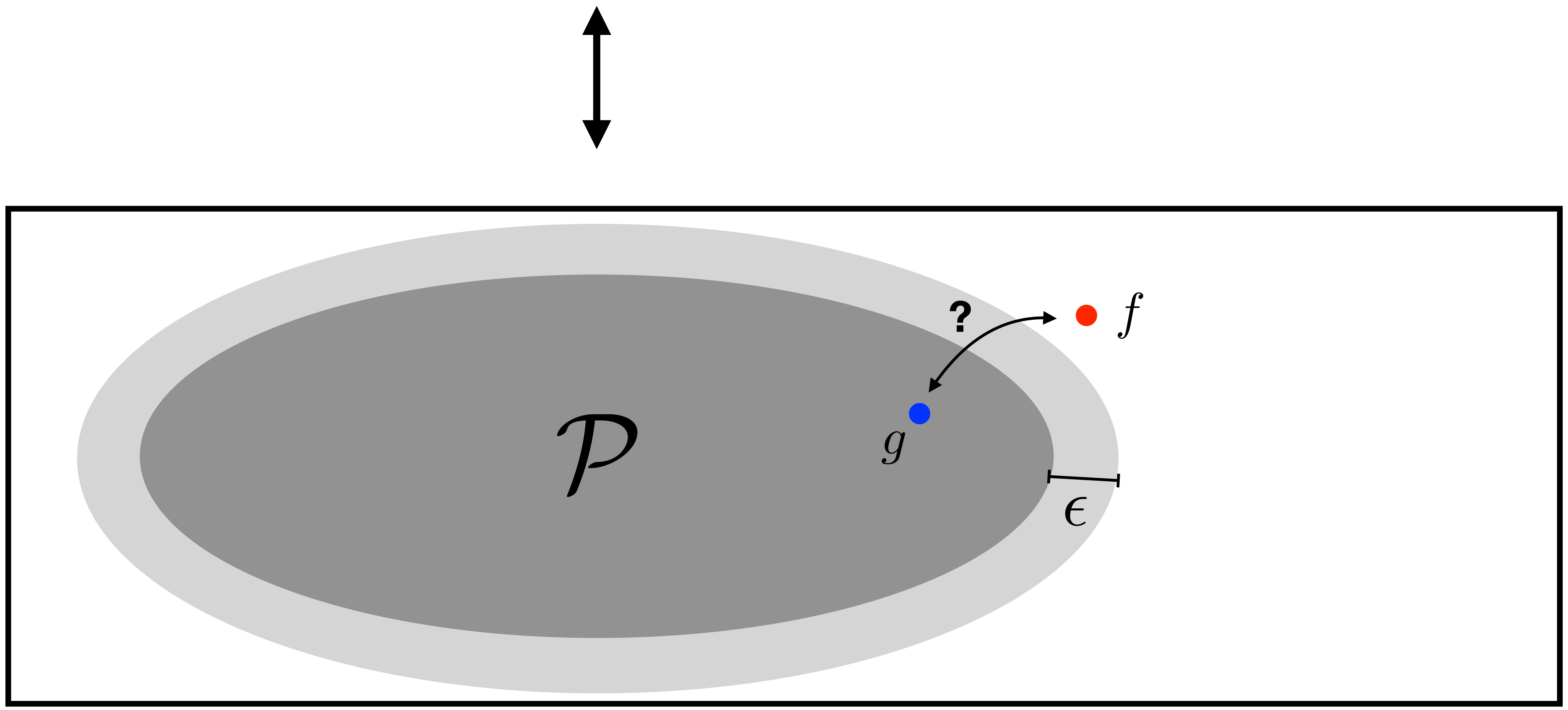
1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}) + f(e_{\boldsymbol{i}}) = f(\boldsymbol{x} + e_{\boldsymbol{i}})$
3. Output YES iff always pass check.

$\Omega(\sqrt{n})$ queries needed

Thm [BLR'93]: $O(1/\epsilon)$ repetitions suffice. Efficient test without learning.

**Natural Algorithm (Check Consistency)**

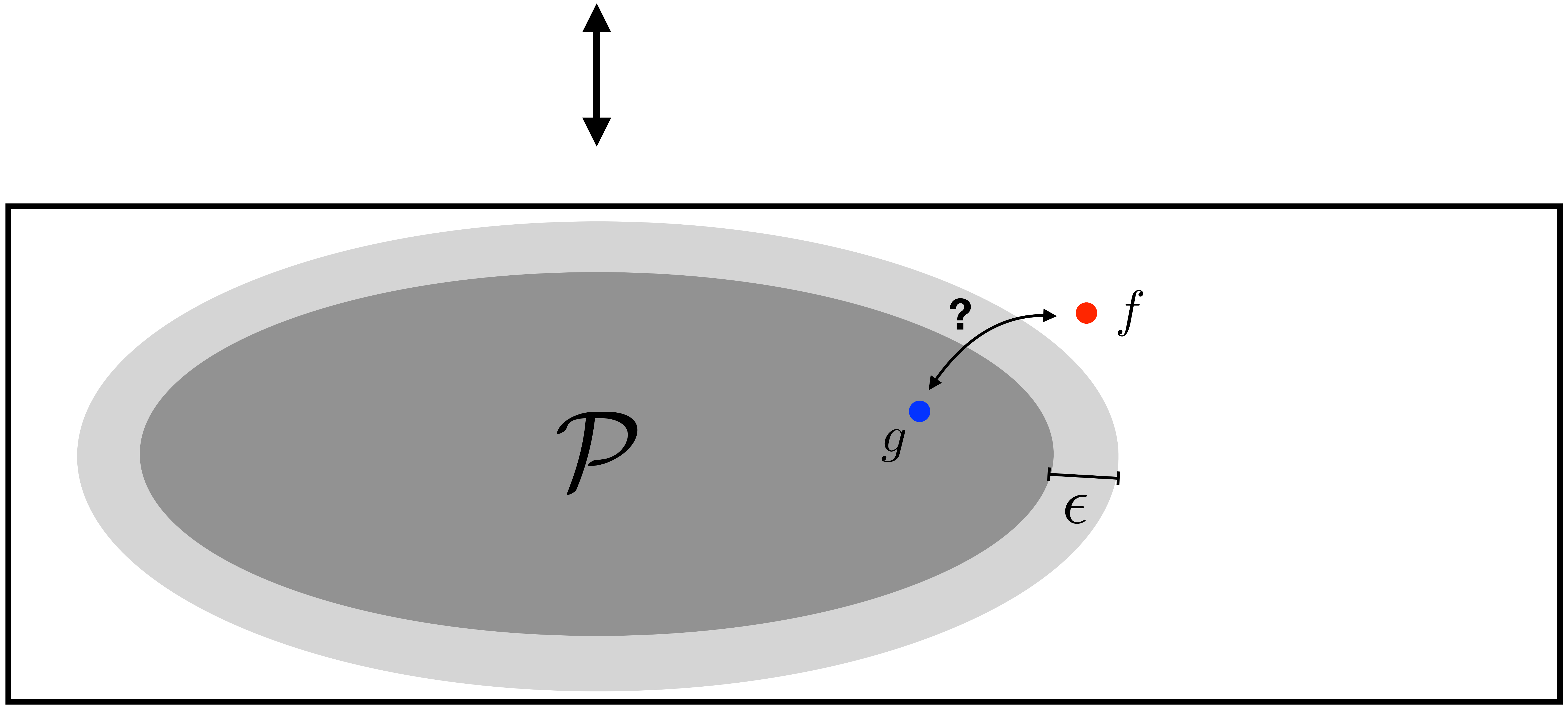
1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.



# Thm [BLR'93]: $O(1/\epsilon)$ repetitions suffice. Efficient test without learning.

**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.




**Voting Function:**

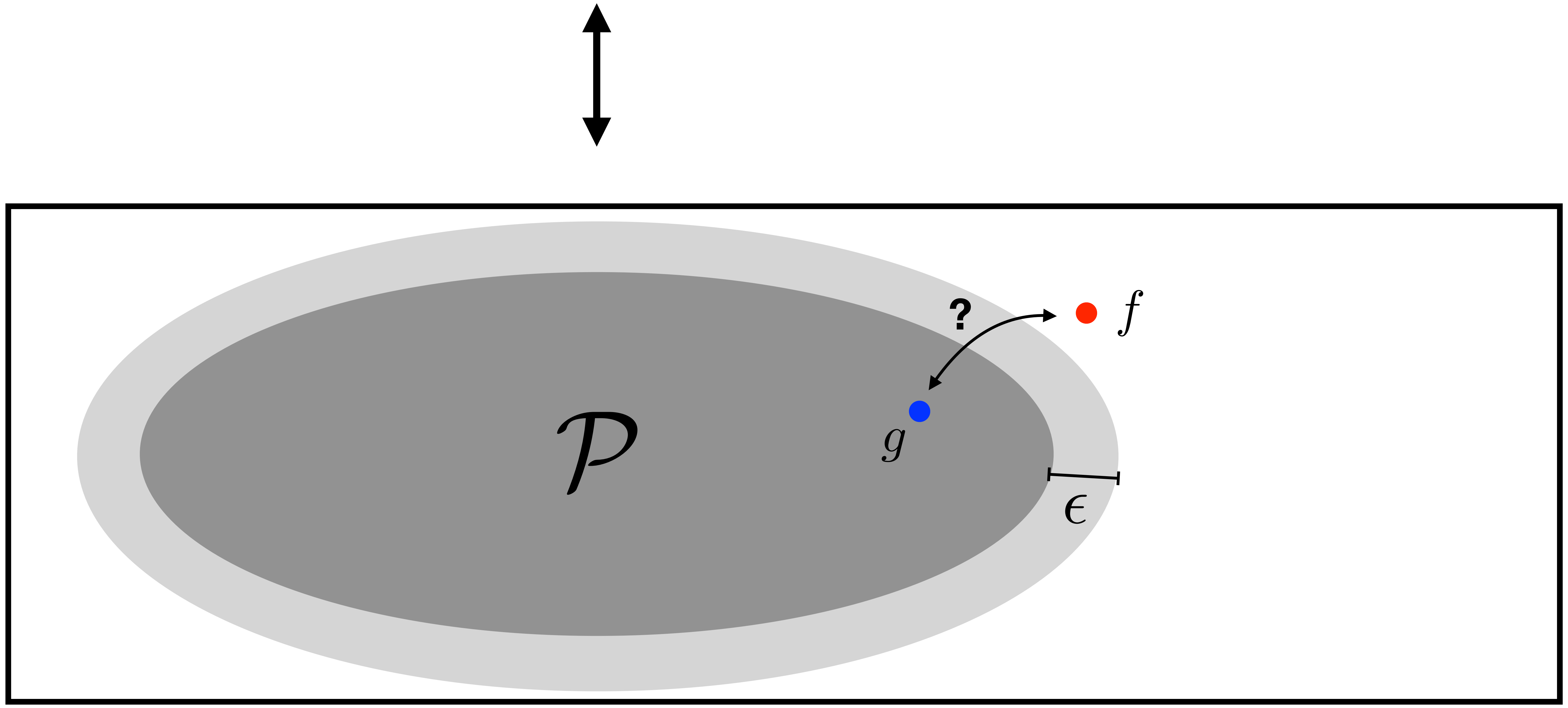
$$g(x) = \begin{cases} 0 & \mathbf{Pr}_{\boldsymbol{y}}\left[f(x+\boldsymbol{y}) - f(\boldsymbol{y}) = 0\right] \geq 1/2 \\ 1 & \mathbf{Pr}_{\boldsymbol{y}}\left[f(x+\boldsymbol{y}) - f(\boldsymbol{y}) = 1\right] > 1/2 \end{cases}$$

(what should $x$ be in order to "pass" often)

# Thm [BLR'93]: $O(1/\epsilon)$ repetitions suffice. Efficient test without learning.

**Natural Algorithm (Check Consistency)**

1. Sample pairs of inputs $\boldsymbol{x}_i, \boldsymbol{y}_i \sim \{0,1\}^n$
2. Query and check $f(\boldsymbol{x}_i) + f(\boldsymbol{y}_i) = f(\boldsymbol{x}_i + \boldsymbol{y}_i)$
3. Output YES iff always pass check.




**Voting Function:**

$$g(x) = \begin{cases} 0 & \mathbf{Pr}_{\boldsymbol{y}}[f(x+\boldsymbol{y}) - f(\boldsymbol{y}) = 0] \geq 1/2 \\ 1 & \mathbf{Pr}_{\boldsymbol{y}}[f(x+\boldsymbol{y}) - f(\boldsymbol{y}) = 1] > 1/2 \end{cases}$$

(what should $x$ be in order to "pass" often)

**Lemma 1:** If $f(x) \neq g(x)$, the battle is almost won.

$$\Rightarrow d(f,g) \cdot (1/2) \leq \mathbf{Pr}_{\boldsymbol{x},\boldsymbol{y}}[\text{catch mistake!}]$$

**Lemma 2:** If $f$ oftentimes passes test, then $g$ is linear.

# Thm [BLR'93]: $O(1/\epsilon)$ repetitions suffice. Efficient test without learning.

If often avoid catching mistakes

↓ **Lemma 2**

Then, the voting function $g$ is linear

↓ **Farness** $\varepsilon \leq d(f, \mathcal{P})$

Then, $\varepsilon \leq d(f, g)$.

↓ **Lemma 1**

Do catch mistakes!

**Voting Function:**

$$g(x) = \begin{cases} 0 & \mathbf{Pr}_{\boldsymbol{y}}\left[f(x+\boldsymbol{y}) - f(\boldsymbol{y}) = 0\right] \geq 1/2 \\ 1 & \mathbf{Pr}_{\boldsymbol{y}}\left[f(x+\boldsymbol{y}) - f(\boldsymbol{y}) = 1\right] > 1/2 \end{cases}$$

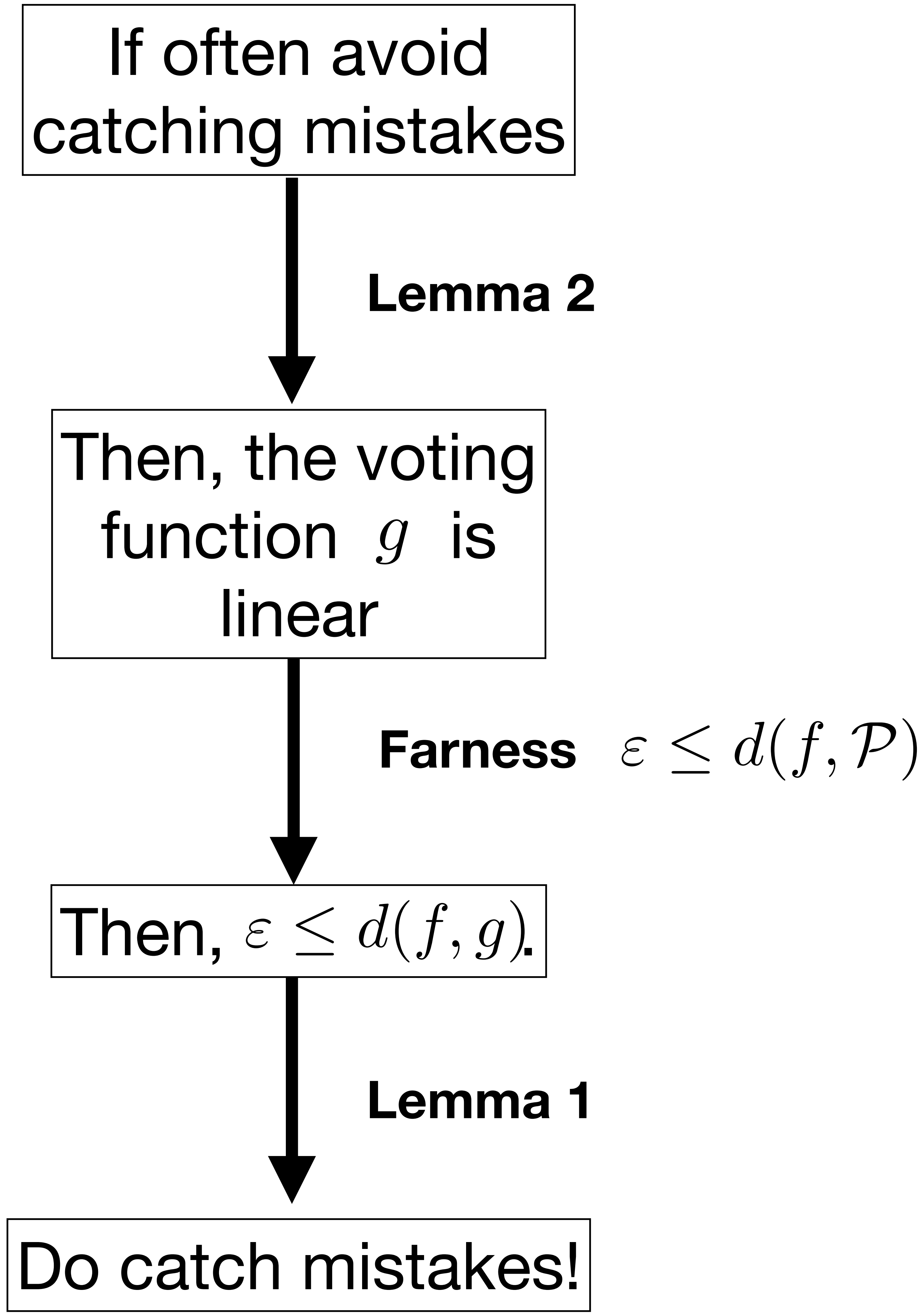
(what should $x$ be in order to "pass" often)

**Lemma 1:** If $f(x) \neq g(x)$, the battle is almost won.

$$\Rightarrow d(f, g) \cdot (1/2) \leq \mathbf{Pr}_{\boldsymbol{x}, \boldsymbol{y}}[\text{catch mistake!}]$$

**Lemma 2:** If $f$ oftentimes passes test, then $g$ is linear.

# Thm [BLR'93]: $O(1/\epsilon)$ repetitions suffice. Efficient test without learning.




**Voting Function:**

$$g(x) = \begin{cases} 0 & \mathbf{Pr}_{\boldsymbol{y}}\left[f(x+\boldsymbol{y}) - f(\boldsymbol{y}) = 0\right] \geq 0.99 \\ 1 & \mathbf{Pr}_{\boldsymbol{y}}\left[f(x+\boldsymbol{y}) - f(\boldsymbol{y}) = 1\right] \geq 0.99 \end{cases}$$

(what should $x$ be in order to "pass" often)

**Lemma 1:** If $f(x) \neq g(x)$, the battle is almost won.

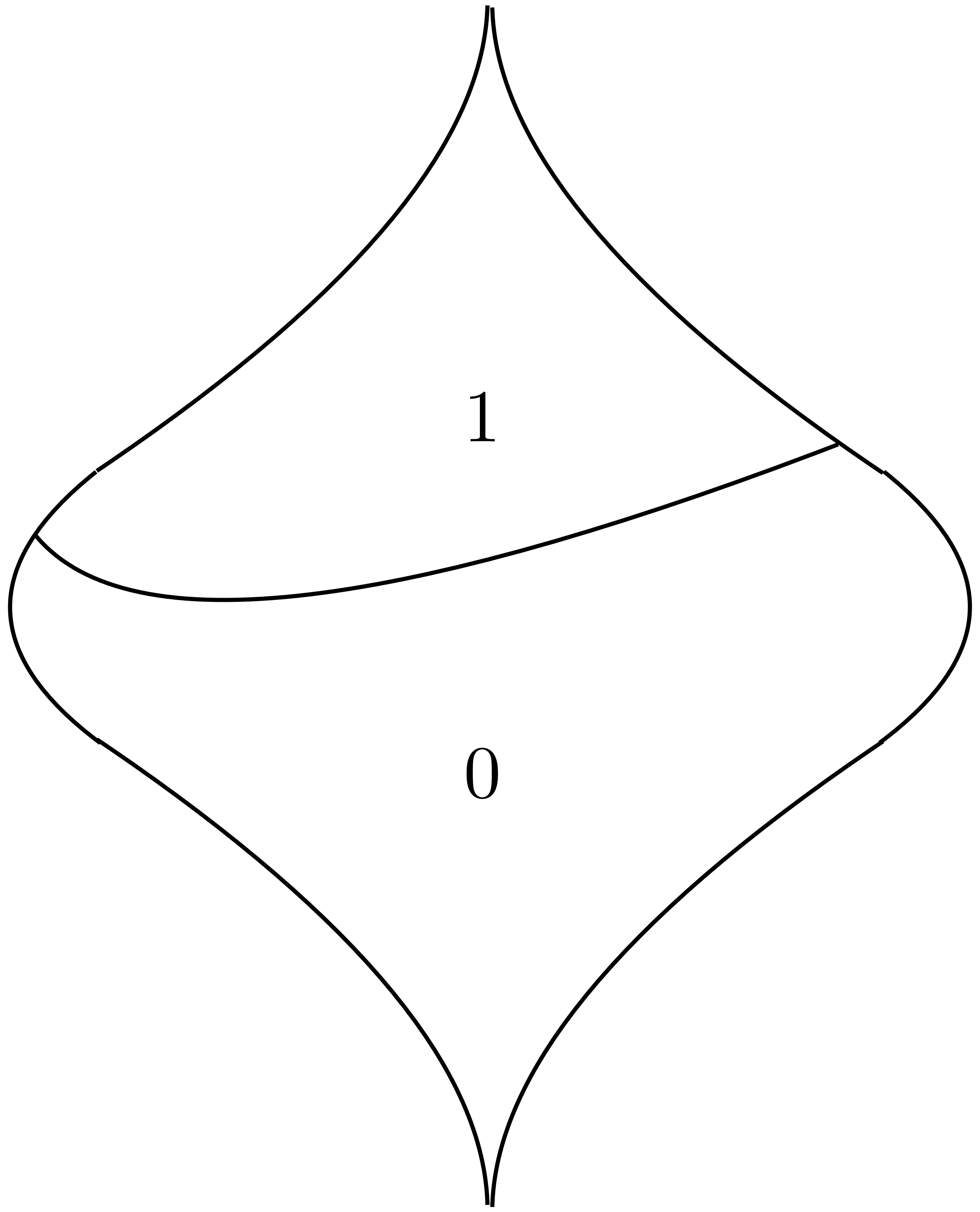
$$\Rightarrow d(f,g) \cdot (1/2) \leq \mathbf{Pr}_{\boldsymbol{x},\boldsymbol{y}}\left[\text{catch mistake!}\right]$$

**Lemma 2:** If $f$ oftentimes passes test, then $g$ is linear.

# Quick Digest of Important Ideas

- Definition of approx. in property testing and why its necessary.
- Generic $O(\log|\mathcal{P}|/\varepsilon)$-query tester for any property via naive learning.
- Linearity testing definition:
  1. An algorithm via *learning* using $n + O(1/\varepsilon)$ queries.
  2. Not all “local” definitions are equally good.
  3. Process getting to linear function (*self-correction*)
- Different Fourier-analytic proof [Bellare, Coppersmith, Håstad, Kiwi, Sudan ’96] in O’Donnel’s book on “Analysis of Boolean Functions”

# Is my function monotone? [Goldreich, Goldwasser, Lehman, Ron, Samorodnitsky '00]




**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone iff

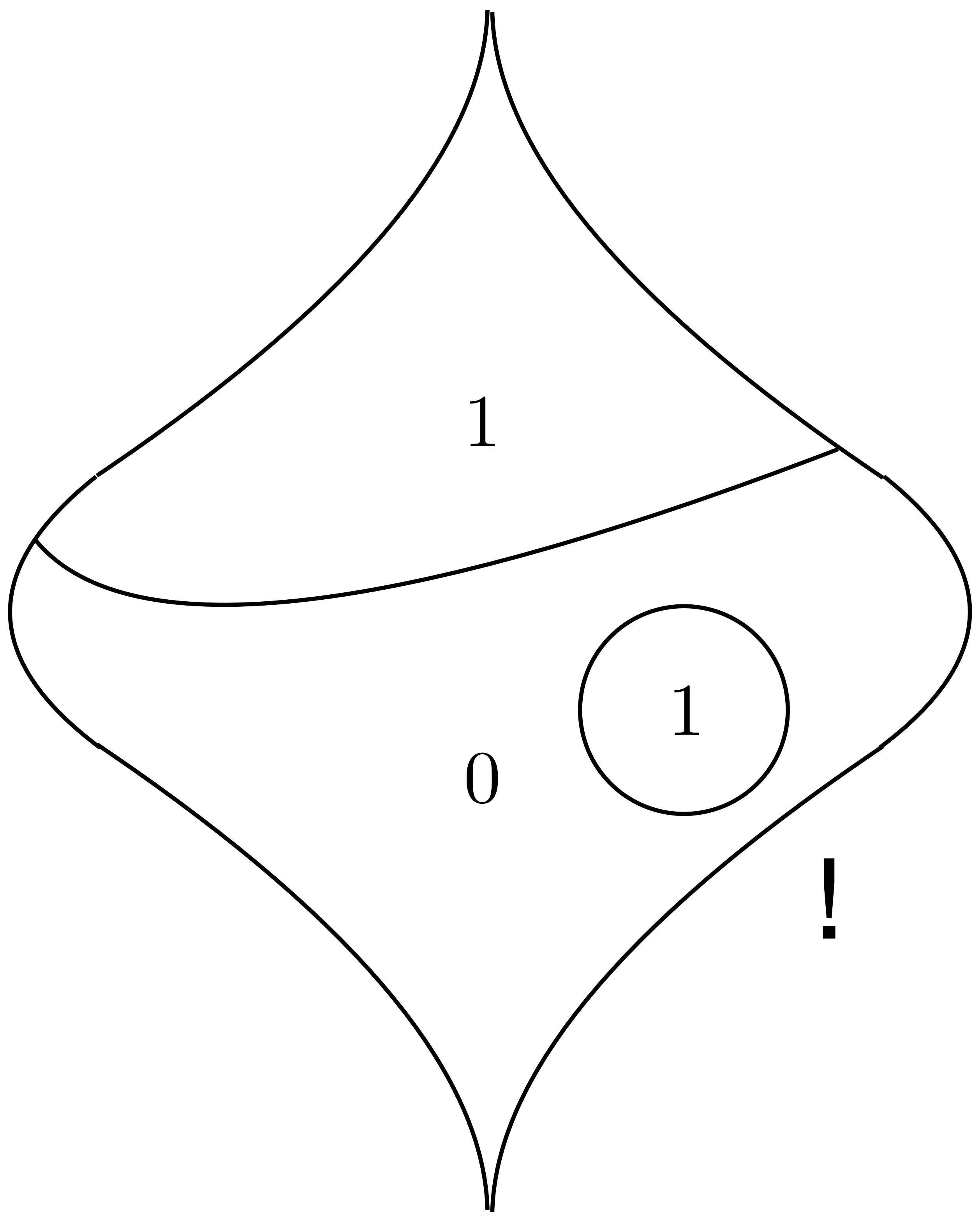
$$f(x) \leq f(y)$$

for any $x \preceq y$.

Examples: $f(x) = x_i$

$f(x) = \mathsf{MAJ}(x)$

# Is my function monotone? [Goldreich, Goldwasser, Lehman, Ron, Samorodnitsky '00]




**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone iff

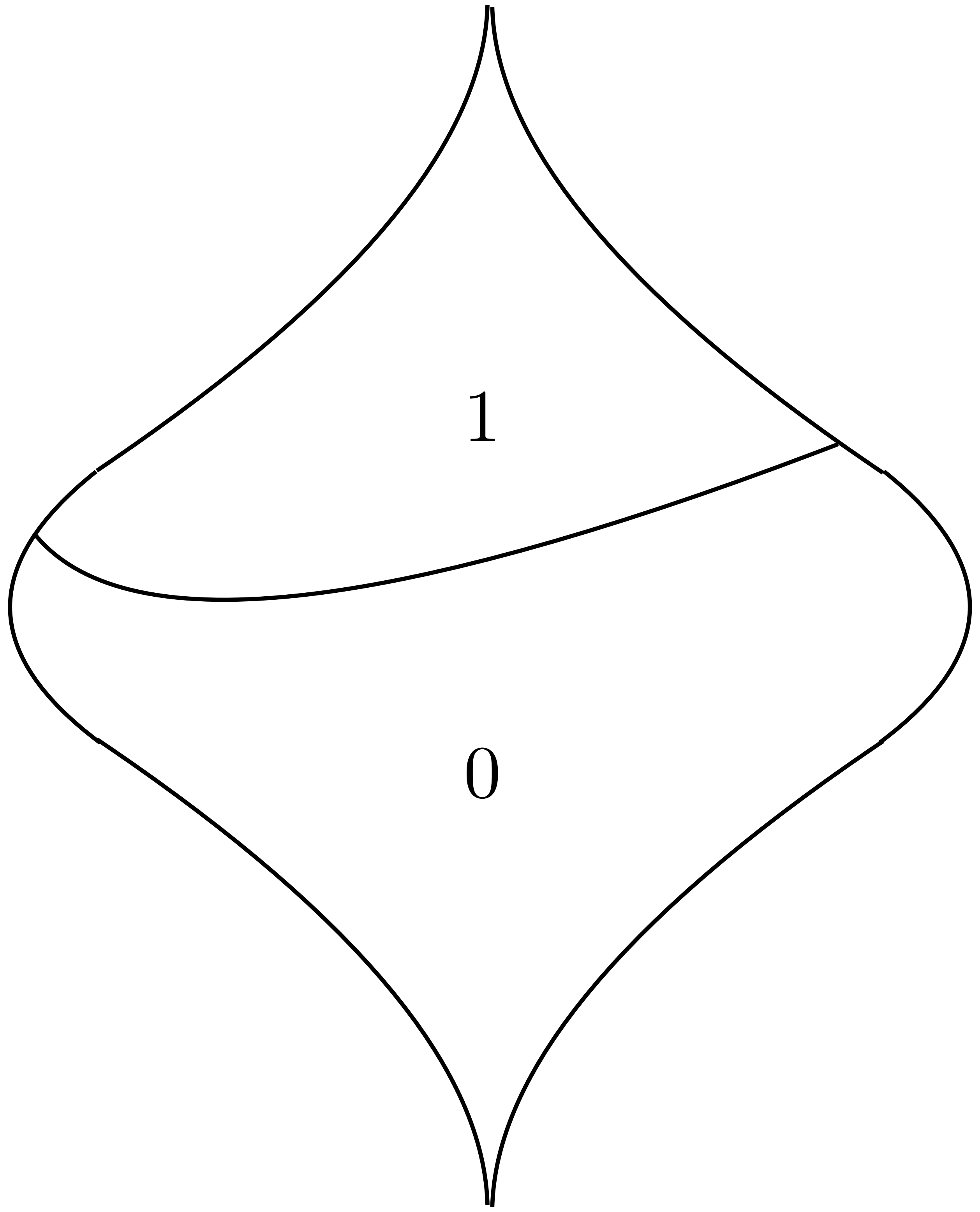
$$f(x) \leq f(y)$$

for any $x \preceq y$.

Examples: $f(x) = x_i$

$f(x) = \mathsf{MAJ}(x)$

# Is my function monotone? [Goldreich, Goldwasser, Lehman, Ron, Samorodnitsky '00]




**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone iff

$$f(x) \leq f(y)$$

for any $x \preceq y$.

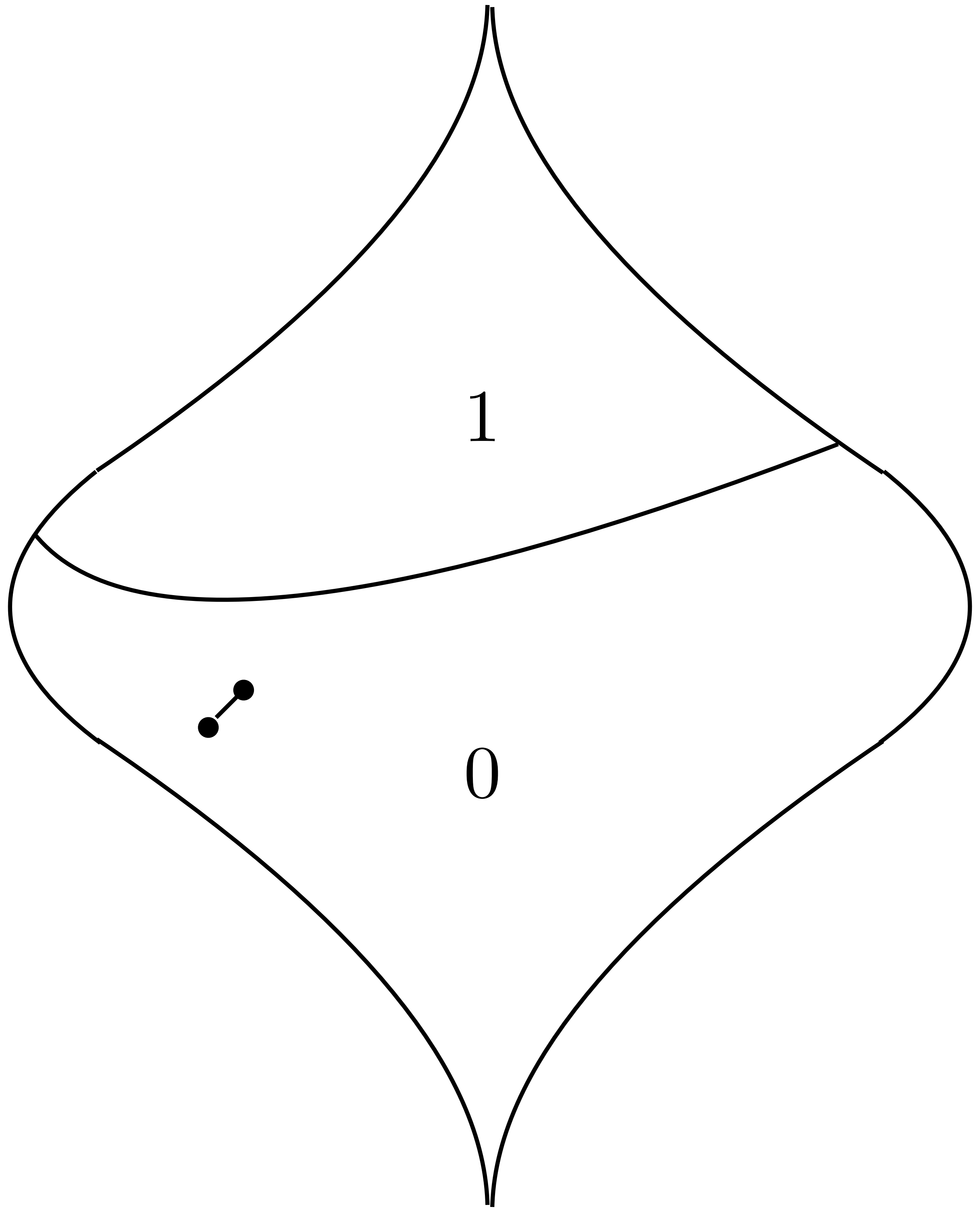
Examples: $f(x) = x_i$

$f(x) = \mathsf{MAJ}(x)$

Generic: $O(\log|\mathcal{P}|/\varepsilon) = O\left(\binom{n}{n/2}/\varepsilon\right)$

Learning: $2^{\tilde{O}(\sqrt{n}/\varepsilon)}$ [Bshouty, Tamon '96]

# A different definition leads to the **Edge Tester**.




**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone iff

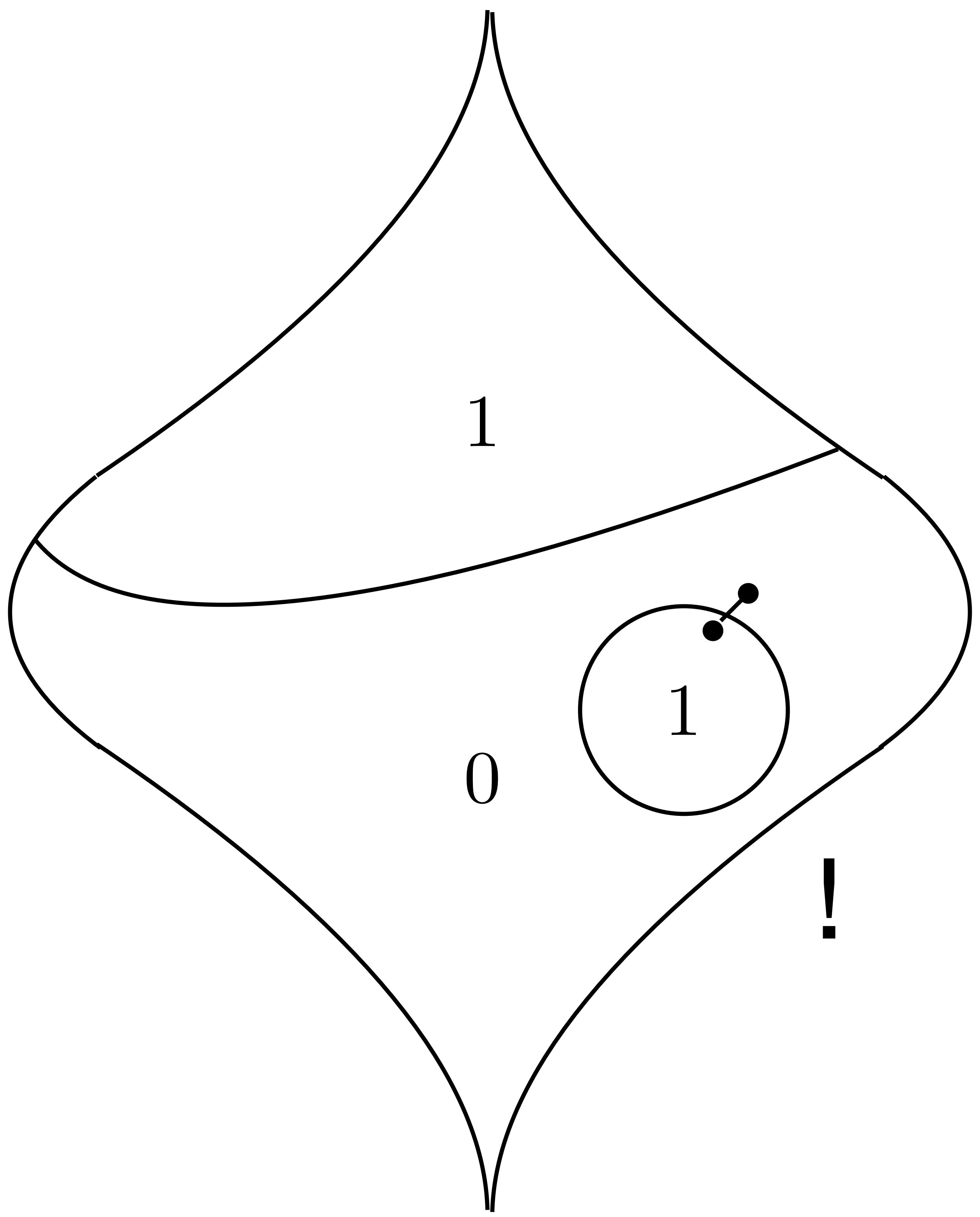
$$f(x^{(i\to 0)}) \leq f(x^{(i\to 1)})$$

for any $x$ and $i$ .

**Edge Tester**

1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}^{(\boldsymbol{i}\to 0)}) \leq f(\boldsymbol{x}^{(\boldsymbol{i}\to 1)})$
3. Output YES iff always pass check.

# A different definition leads to the **Edge Tester**.




**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone iff

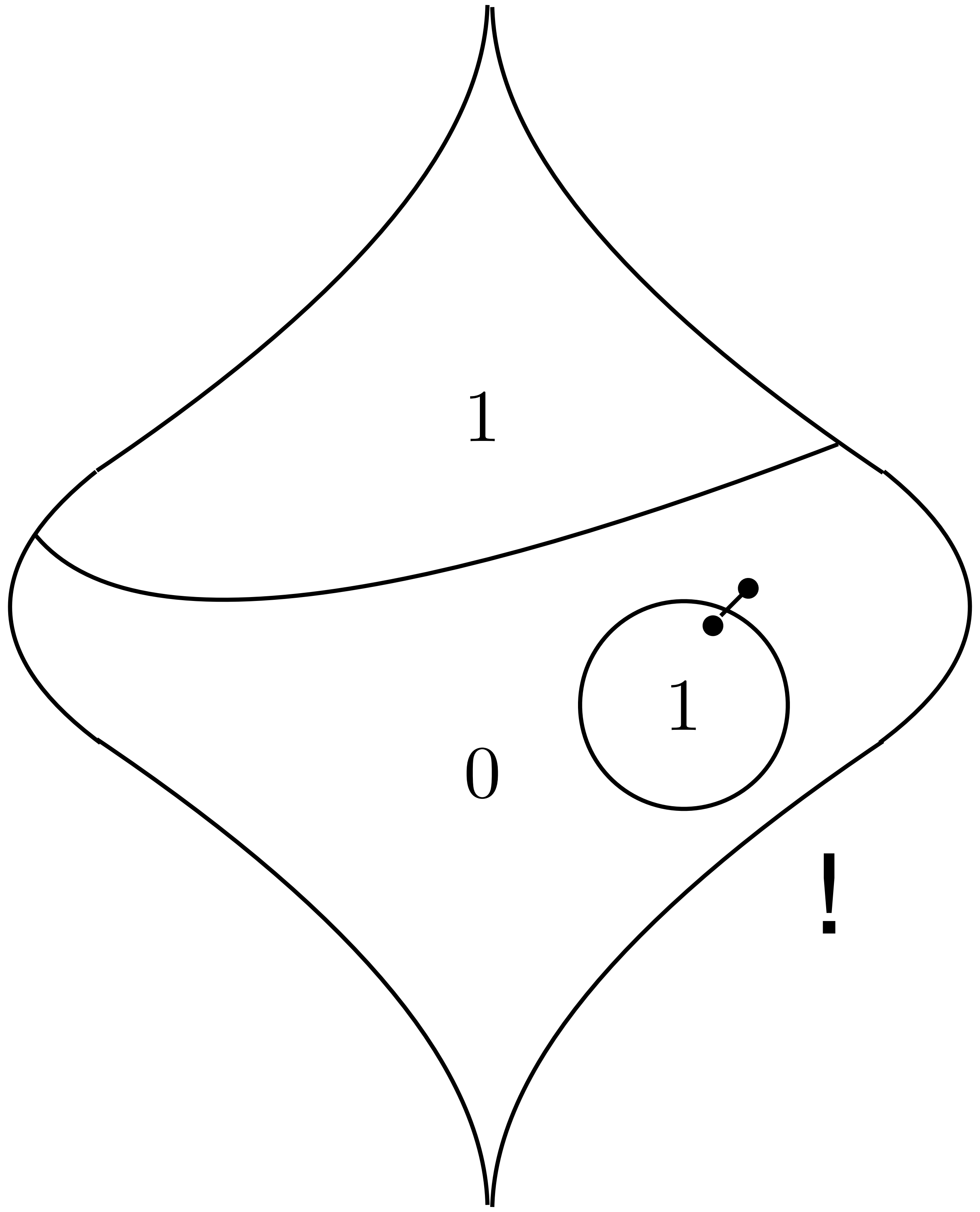
$$f(x^{(i\to 0)}) \leq f(x^{(i\to 1)})$$

for any $x$ and $i$ .

**Edge Tester**

1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}^{(\boldsymbol{i}\to 0)}) \leq f(\boldsymbol{x}^{(\boldsymbol{i}\to 1)})$
3. Output YES iff always pass check.

# A different definition leads to the **Edge Tester**.




**Definition**: a function $f \colon \{0,1\}^n \to \{0,1\}$ is monotone iff

$$f(x^{(i \to 0)}) \leq f(x^{(i \to 1)})$$

for any $x$ and $i$ .

**Edge Tester**

1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}^{(\boldsymbol{i} \to 0)}) \leq f(\boldsymbol{x}^{(\boldsymbol{i} \to 1)})$
3. Output YES iff always pass check.

**Theorem [GGLRS'00]:** $O(n/\varepsilon)$ suffice.

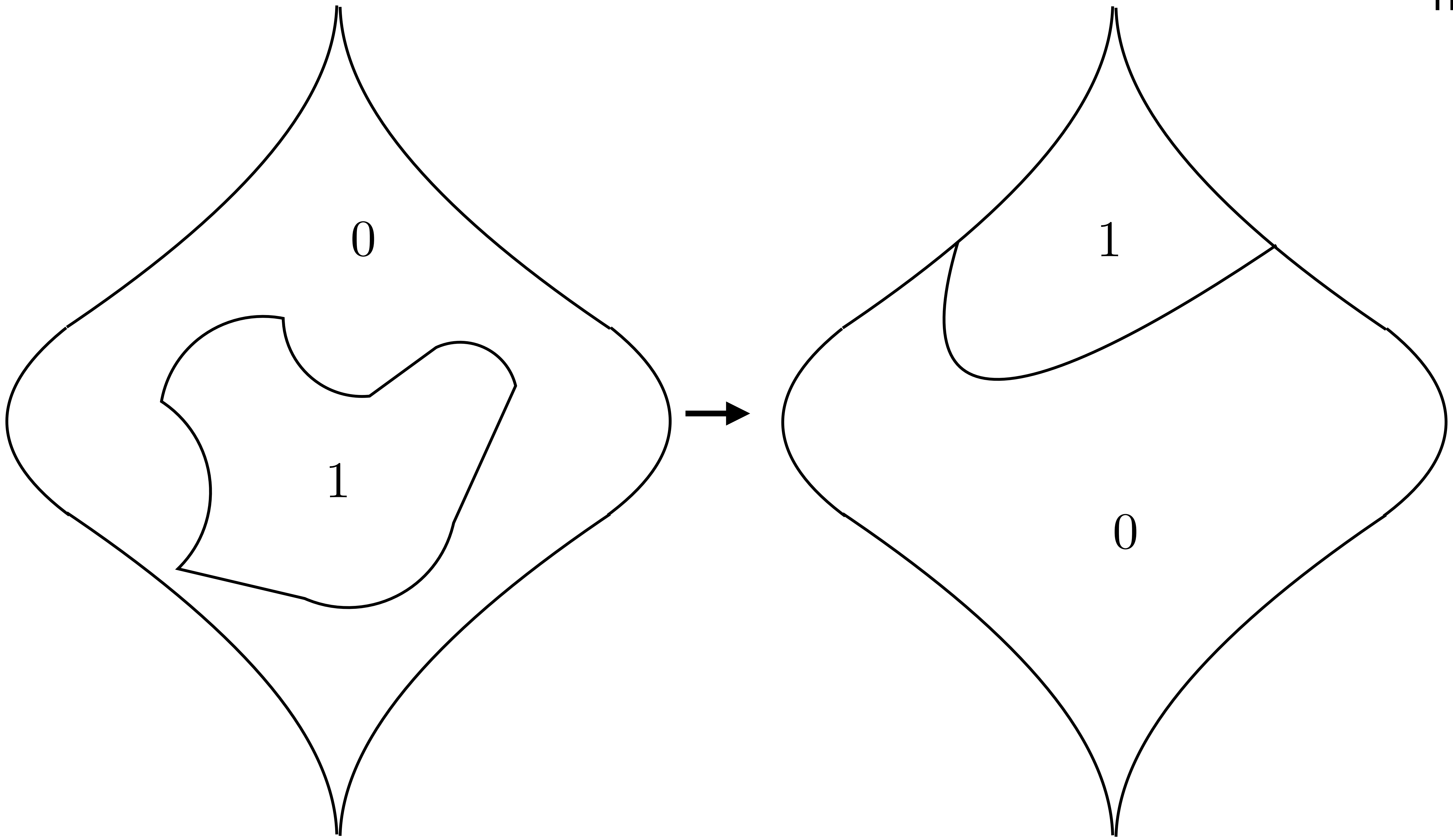
# [GGLRS'00] Far from monotone functions have a lot of "edge" evidence.

**Structural Theorem:**
If $f$ is far from monotone, there is a lot of *structured evidence* of non-monotonicity.

**Transformed function:**

$$g(x) = S_1(S_2(S_3(\ldots(S_n(f))\ldots)(x)$$

is a monotone function, where changes from $f$ came from non-monotone edges.




**Edge Tester**

1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}^{(\boldsymbol{i}\to 0)}) \leq f(\boldsymbol{x}^{(\boldsymbol{i}\to 1)})$
3. Output YES iff always pass check.

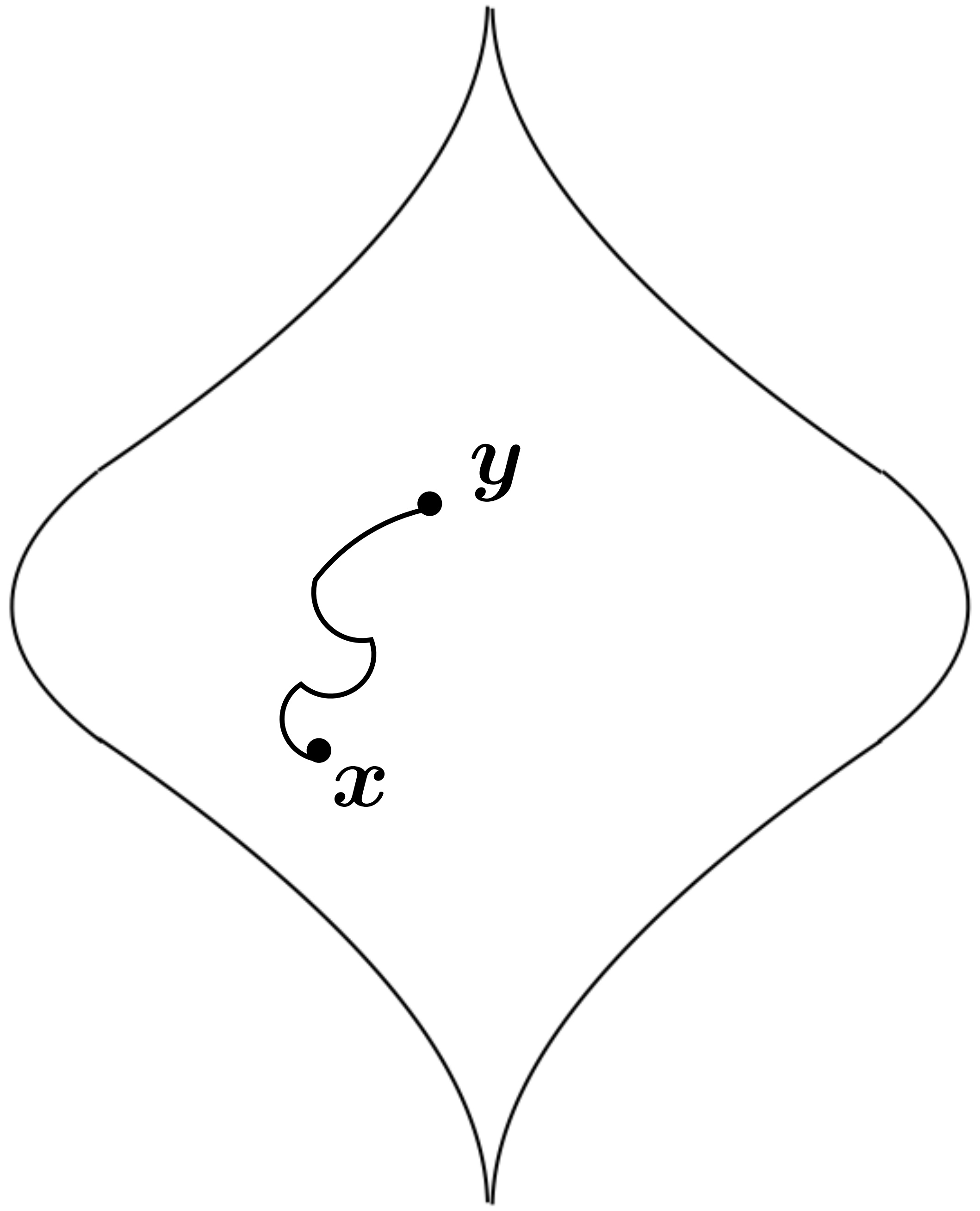
# A different **path** for monotonicity testing

[Chakrabarty, Seshadhri '13, Chen, Tan, Servedio '14, Khot, Minzer, Safra '15]

**Structural Theorem:**
If $f$ is far from monotone, there is a lot of *nicely structured evidence* of non-monotonicity.

**Path Tester**

1. Sample $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$ .
2. Query and check $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$
3. Output YES iff always pass check.



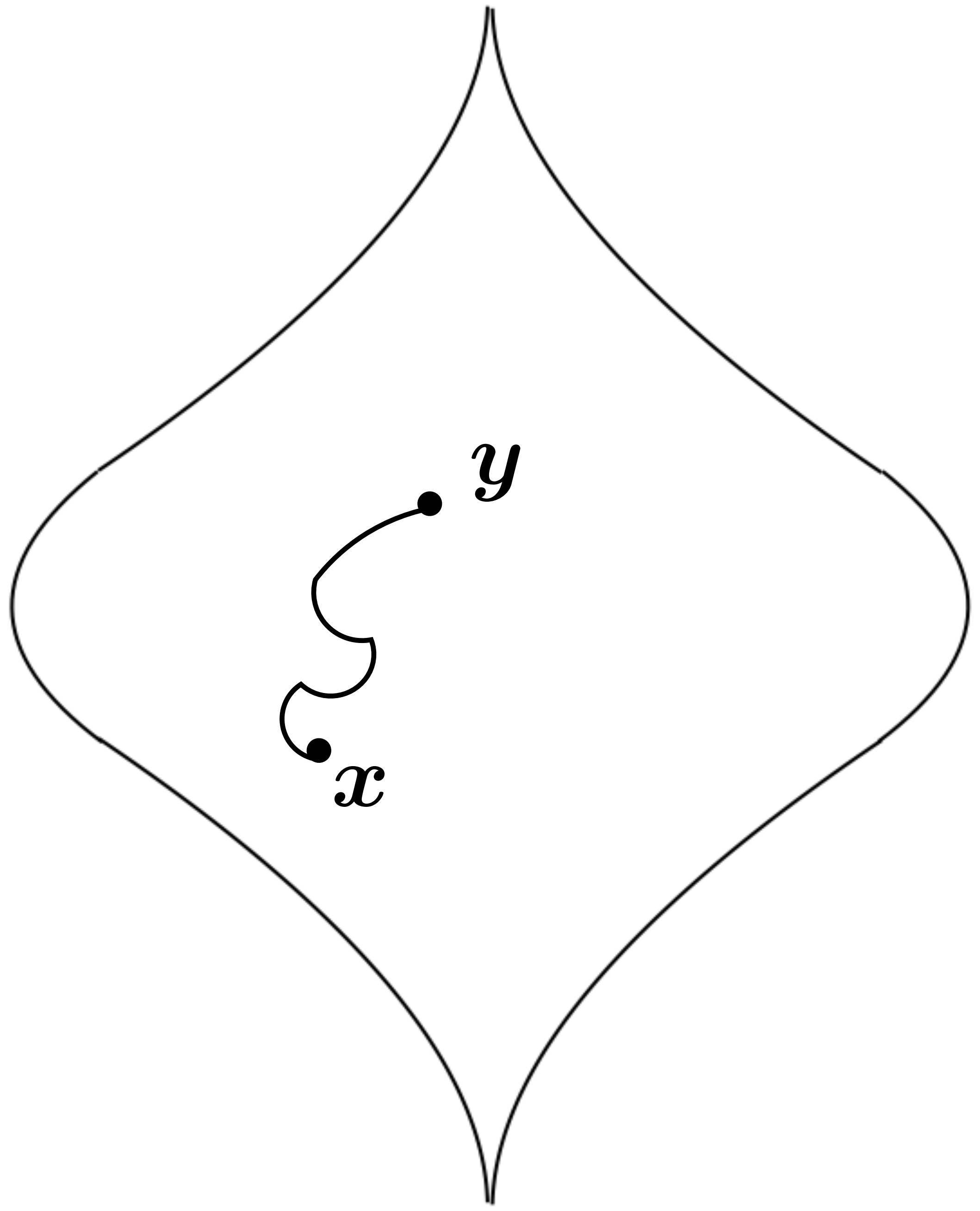

**Theorem [KMS'15]:** $\tilde{O}(\sqrt{n}/\varepsilon^2)$ suffice.

**Additional properties**: non-adaptive, one-sided.

# A different **path** for monotonicity testing

[Chakrabarty, Seshadhri ’13, Chen, Tan, Servedio ’14, Khot, Minzer, Safra ’15]

**Structural Theorem:**
If $f$ is far from monotone, there is a lot of *nicely structured evidence* of non-monotonicity.




**Path Tester**

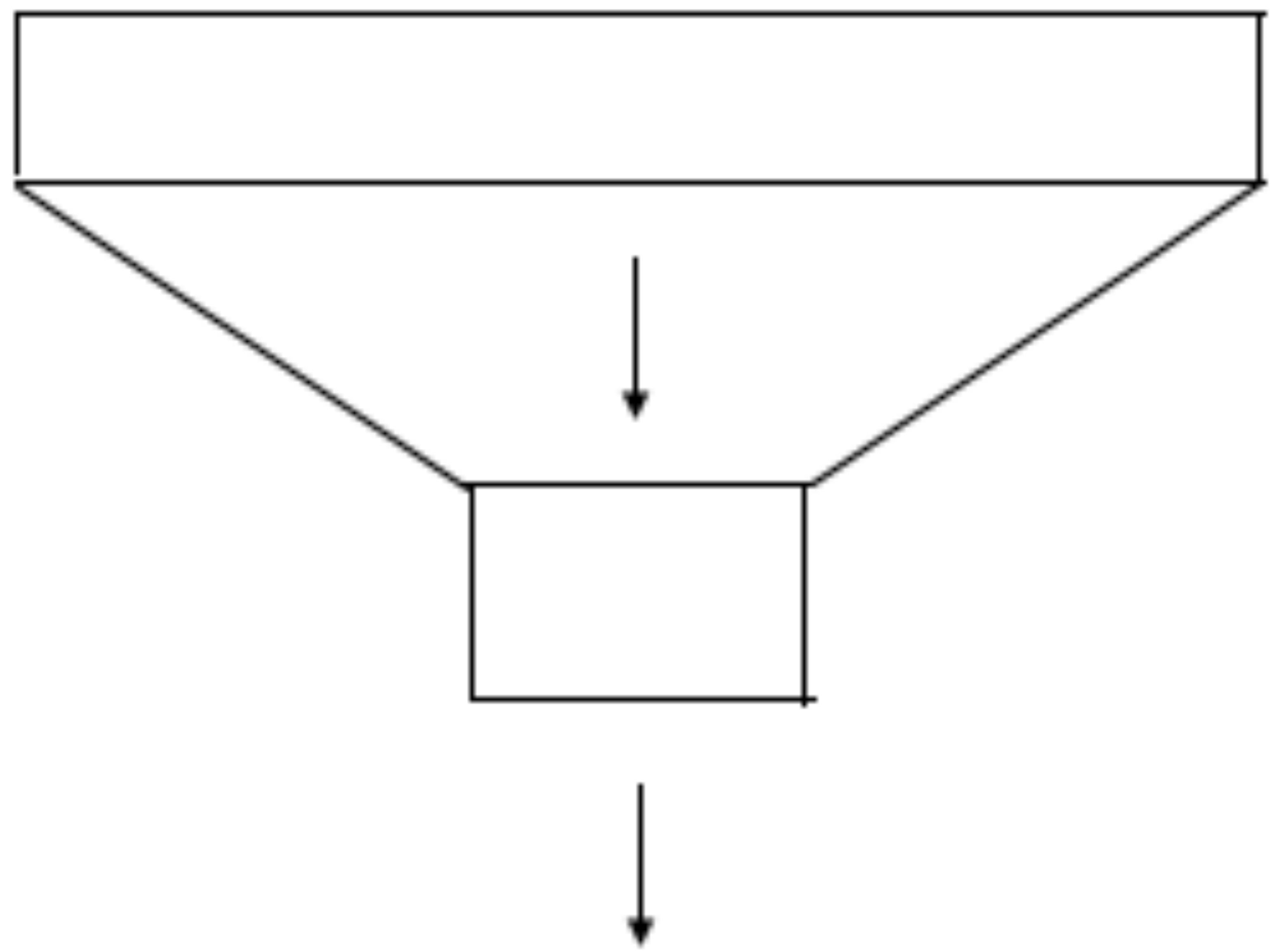
1. Sample $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$ .
2. Query and check $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$
3. Output YES iff always pass check.

**Theorem [KMS’15]:** $\tilde{O}(\sqrt{n}/\varepsilon^2)$ suffice.

**Additional properties**: non-adaptive, one-sided.

[Black, Chakrabarty, Seshadhri ’23] for hypergrids
[Chen-W-Xie ’18] query lower bounds
[Black, Kalemaj, Raskhodnikova ’23, Pinto ‘24] real-valued functions

# Is my function a junta? [Fischer, Kindler, Ron, Safra, Samorodnitsky '03]



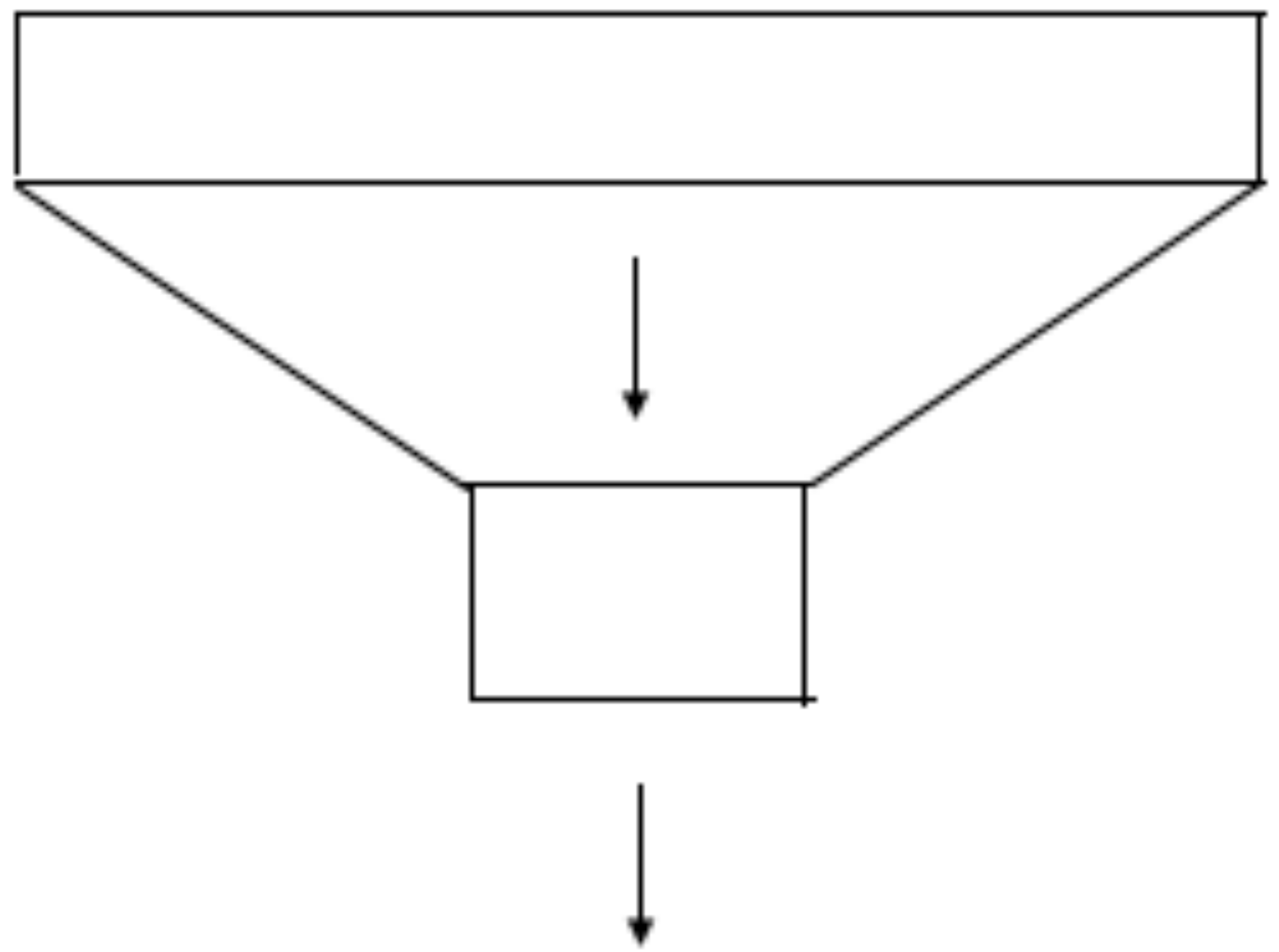
**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is a k-junta if there is a set $S \subset [n]$ of size $k$ such that

$$f(x) = g(x_S)$$

for some $g\colon \{0,1\}^k \to \{0,1\}$.

Examples: $f(x) = x_i$ is a 1-junta.

# Is my function a junta? [Fischer, Kindler, Ron, Safra, Samorodnitsky ’03]



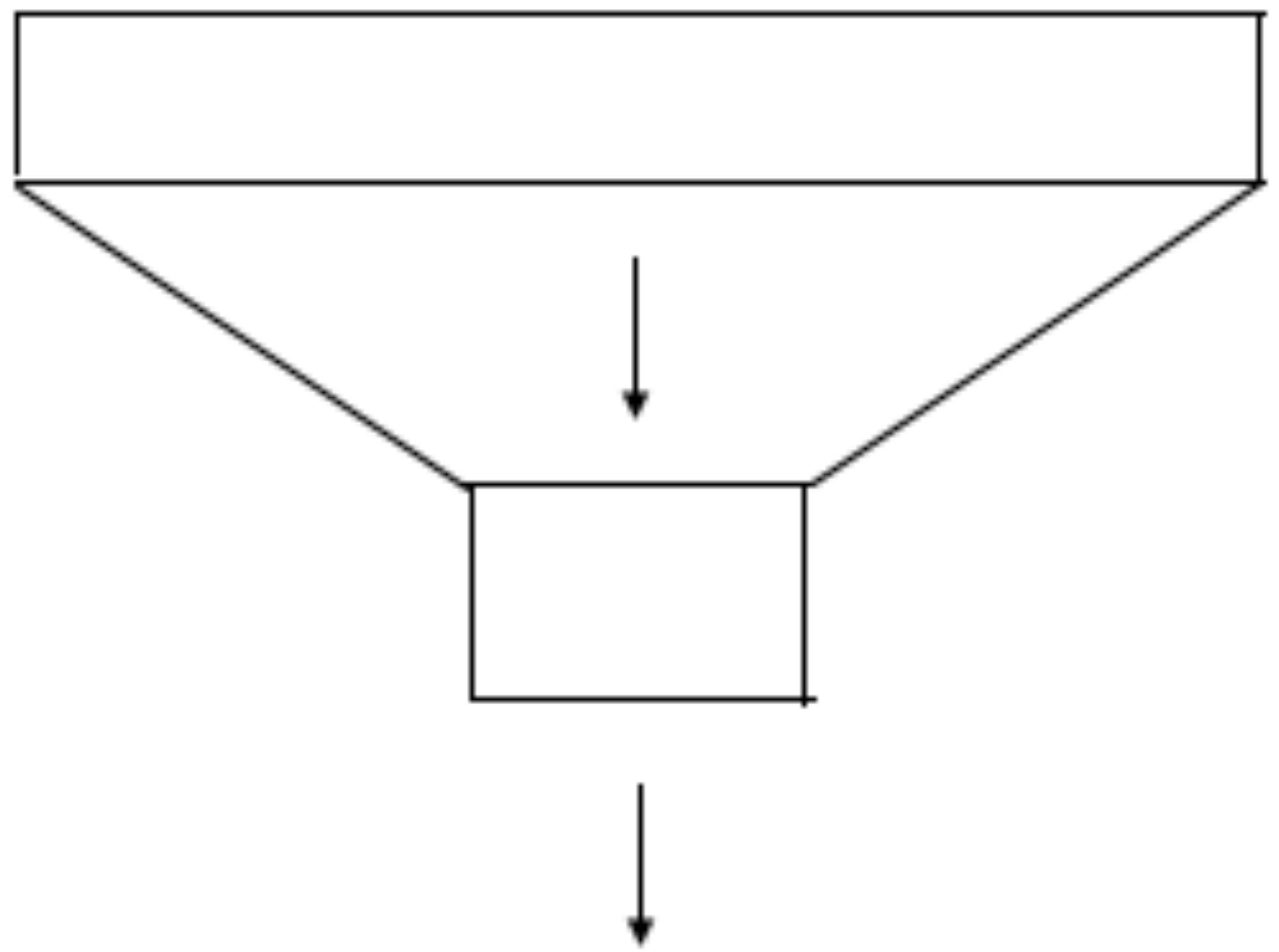
**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is a k-junta if there is a set $S \subset [n]$ of size $k$ such that

$$f(x) = g(x_S)$$

for some $g\colon \{0,1\}^k \to \{0,1\}$.

Examples: $f(x) = x_i$ is a 1-junta.
Majority is not a junta.

# Is my function a junta? [Fischer, Kindler, Ron, Safra, Samorodnitsky '03]



**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is a k-junta if there is a set $S \subset [n]$ of size $k$ such that

$$f(x) = g(x_S)$$

for some $g\colon \{0,1\}^k \to \{0,1\}$.

Examples: $f(x) = x_i$ is a 1-junta.
Majority is not a junta.

By learning variables:

$$O(k \log n + k \log k/\varepsilon)$$

**Theorem [Blais '10]**: $O(k \log k + k/\varepsilon)$

# Is my function almost a junta? (Tolerant testing)

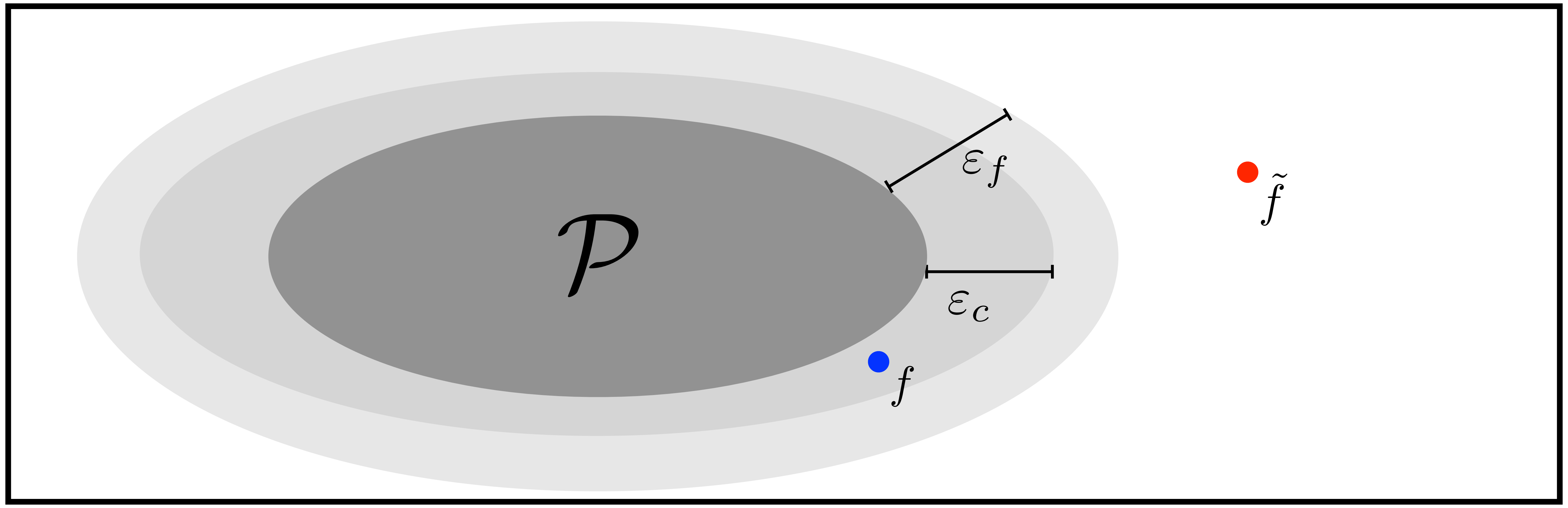
**Definition:** a function $f\colon \{0,1\}^n \to \{0,1\}$ is a k-junta if there is a set $S \subset [n]$ of size $k$ such that

$$f(x) = g(x_S)$$

for some $g\colon \{0,1\}^k \to \{0,1\}$.

$f$ output YES w.p 2/3

$\tilde{f}$ output NO w.p 2/3



# Is my function almost a junta? (Tolerant testing)

**Definition:** a function $f\colon \{0,1\}^n \to \{0,1\}$ is a k-junta if there is a set $S \subset [n]$ of size $k$ such that

$$f(x) = g(x_S)$$

for some $g\colon \{0,1\}^k \to \{0,1\}$.

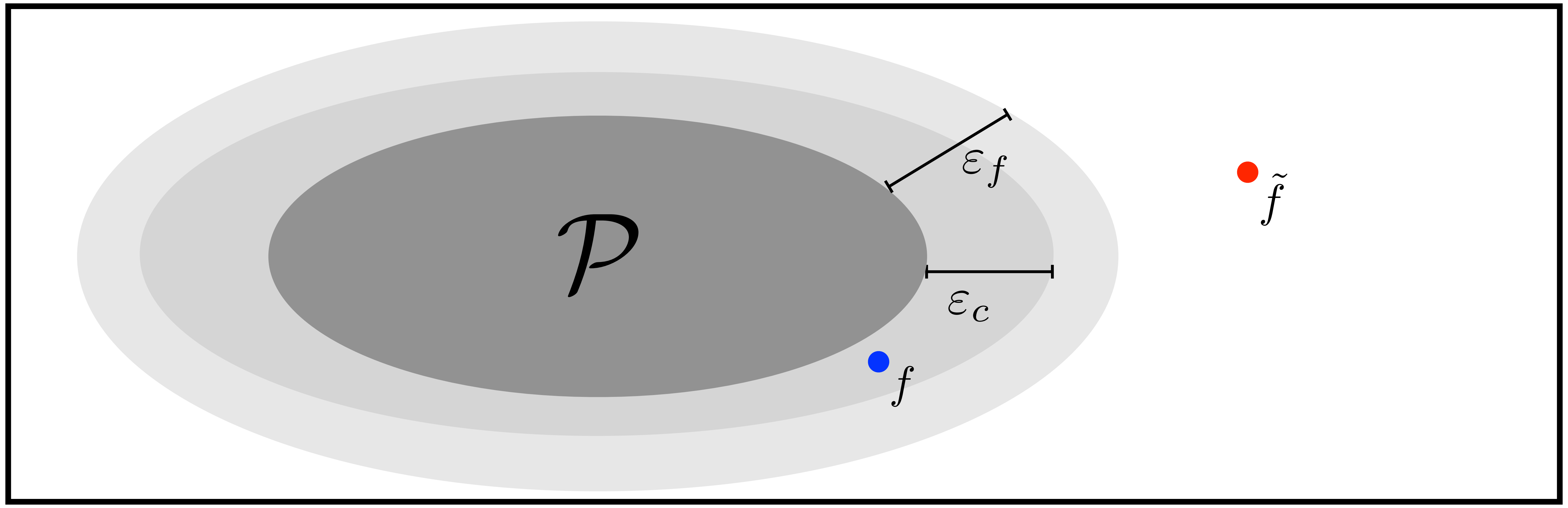
[Blais, Canonne, Eden, Levi, Ron '19]
[De, Mossel, Neeman '19]
[Iyer, Tal, Whitmeyer '21]
[Nadimpalli, Patel '24]

$$2^{\tilde{O}(\sqrt{k \log(1/(\varepsilon_f - \varepsilon_c))})}$$

*near-optimal for non-adaptive algorithms.*

$f$ output YES w.p 2/3

$\tilde{f}$ output NO w.p 2/3



# Is my function a junta, w.r.t this distribution? (Distribution-free)

**Definition:** a function $f\colon \{0,1\}^n \to \{0,1\}$ is a k-junta if there is a set $S \subset [n]$ of size $k$ such that

$$f(x) = g(x_S)$$

for some $g\colon \{0,1\}^k \to \{0,1\}$.

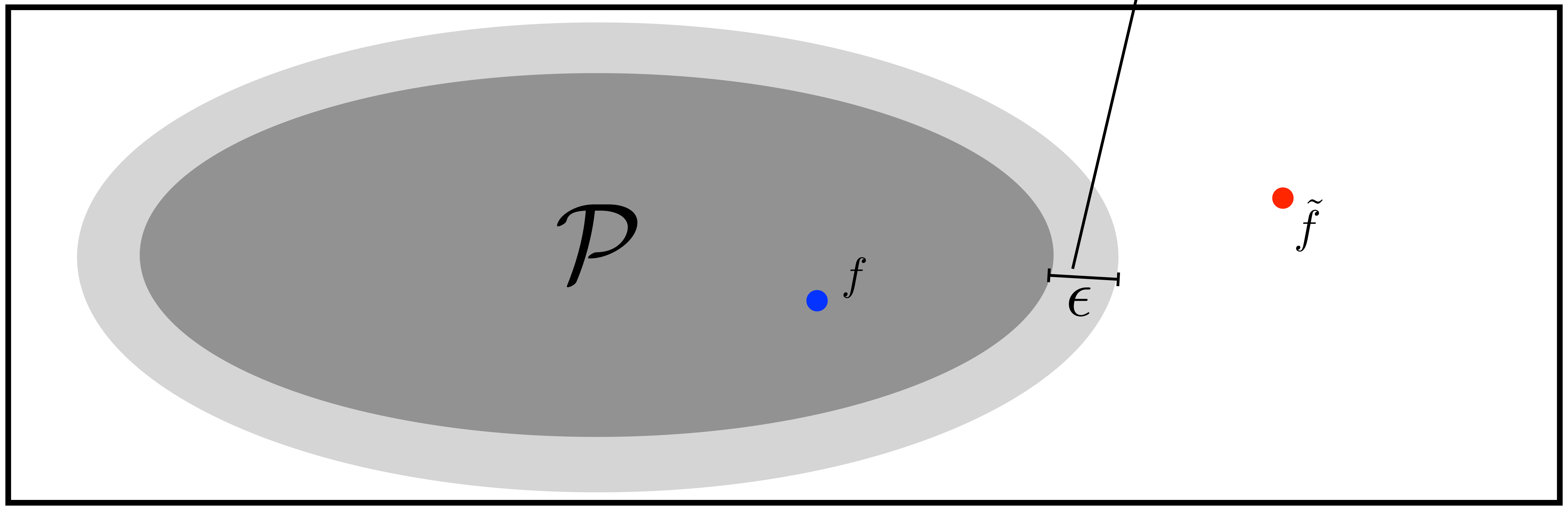
[Chen, Liu, Servedio, Sheng, Xie '18]
[Bshouty '19]

$\tilde{O}(k/\varepsilon)$

$f$ output YES w.p 2/3

$\tilde{f}$ output NO w.p 2/3

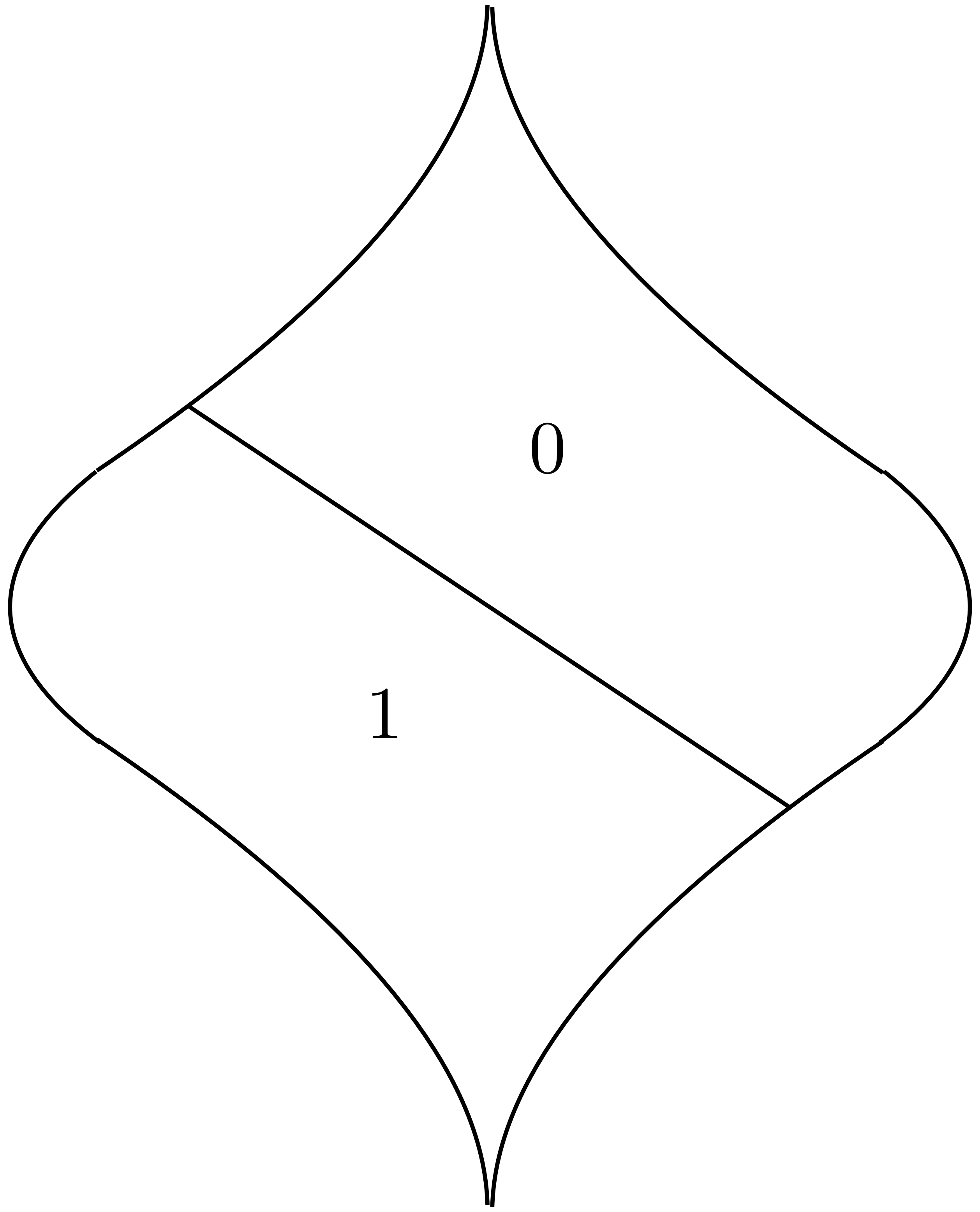
$$d(f,g) = \Pr_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$$



# Thanks

Reference: “Introduction to Property Testing” by Goldreich ‘17
“Algorithmic and Analysis Techniques in Property Testing” by Ron ‘10

# A different definition leads to the **Edge Tester**.




**"Hard" Instance:** $f(x) = x_j$

Need $\Omega(n)$ queries

**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone iff

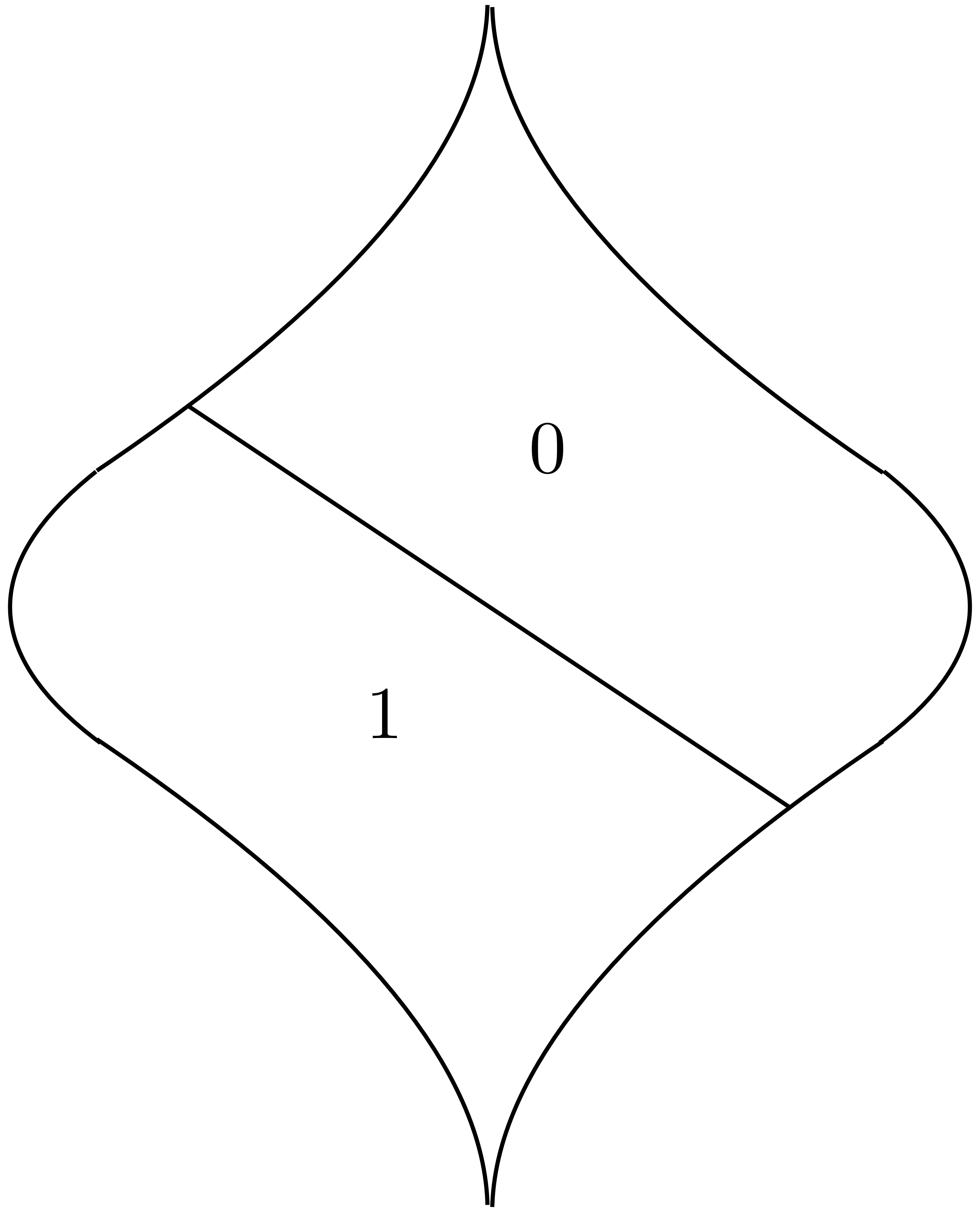
$$f(x^{(i \to 0)}) \leq f(x^{(i \to 1)})$$

for any $x$ and $i$.

**Edge Tester**

1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}^{(\boldsymbol{i} \to 0)}) \leq f(\boldsymbol{x}^{(\boldsymbol{i} \to 1)})$
3. Output YES iff always pass check.

# Is my function monotone? [Goldreich, Goldwasser, Lehman, Ron, Samorodnitsky '00]




**"Hard" Instance:** $f(x) = x_j$

Need $\Omega(n)$ queries

**Theorem [GGLRS'00]:** $O(n/\varepsilon)$ suffice.

**Definition**: a function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone iff

$$f(x^{(i\to 0)}) \leq f(x^{(i\to 1)})$$

for any $x$ and $i$.

**Edge Tester**

1. Sample $\boldsymbol{x} \sim \{0,1\}^n, \boldsymbol{i} \sim [n]$
2. Query and check $f(\boldsymbol{x}^{(\boldsymbol{i}\to 0)}) \leq f(\boldsymbol{x}^{(\boldsymbol{i}\to 1)})$
3. Output YES iff always pass check.