Decision tree complexity

Swagato Sanyal

Department of CSE, IIT Kharagpur

February 21, 2019

Sensitivity, Query Complexity, Communication Complexity and Fourier Analysis of Boolean Function

ISI Kolkata

Total Boolean function

$$f : \{0,1\}^n \rightarrow \{0,1\}.$$

x_1	x_2	x_3	$f(x)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Partial Boolean function

$$f : \{0,1\}^n \rightarrow \{0,1,*\}.$$

x_1	x_2	x_3	$f(x)$
0	0	0	0
0	0	1	1
0	1	0	*
0	1	1	1
1	0	0	*
1	0	1	0
1	1	0	*
1	1	1	*

Relation: $f \subseteq \{0,1\}^n \times \mathcal{R}$.

x_1	x_2	x_3	$f(x)$
0	0	0	00,01
0	0	1	1,101
0	1	0	2,4,5
0	1	1	1001
1	0	0	A1b,2
1	0	1	0
1	1	0	011
1	1	1	<empty>

Outline

1. The query model
2. Composed functions
3. Our work

Outline

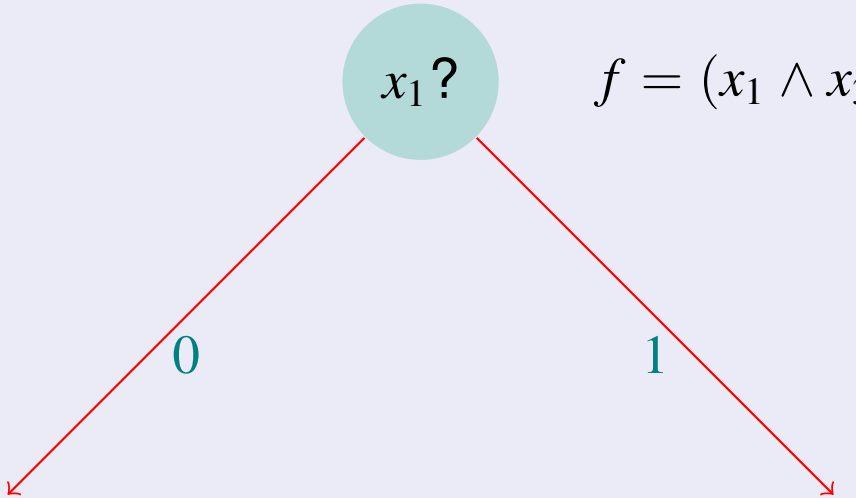
1. The query model
2. Composed functions
3. Our work

Decision tree: deterministic

x_1?

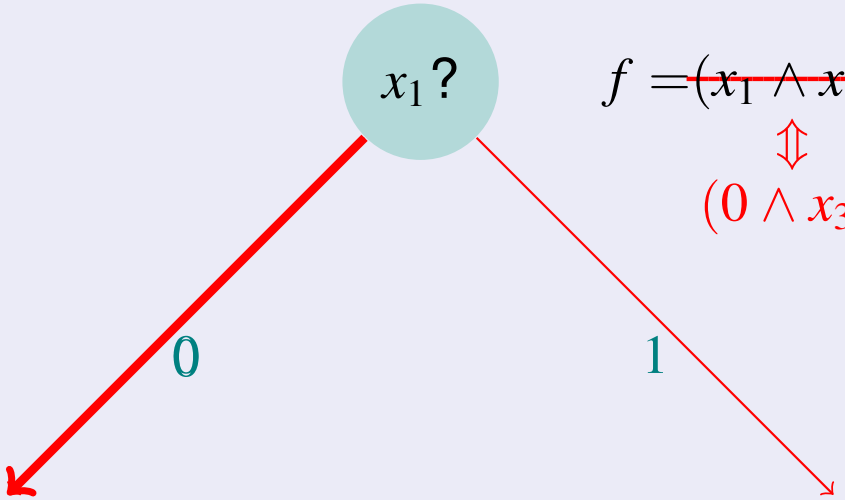
$$f = (x_1 \wedge x_3) \vee (\overline{x_1} \wedge x_2)\,.$$

Decision tree: deterministic



$$f = (x_1 \wedge x_3) \vee (\overline{x_1} \wedge x_2)\,.$$

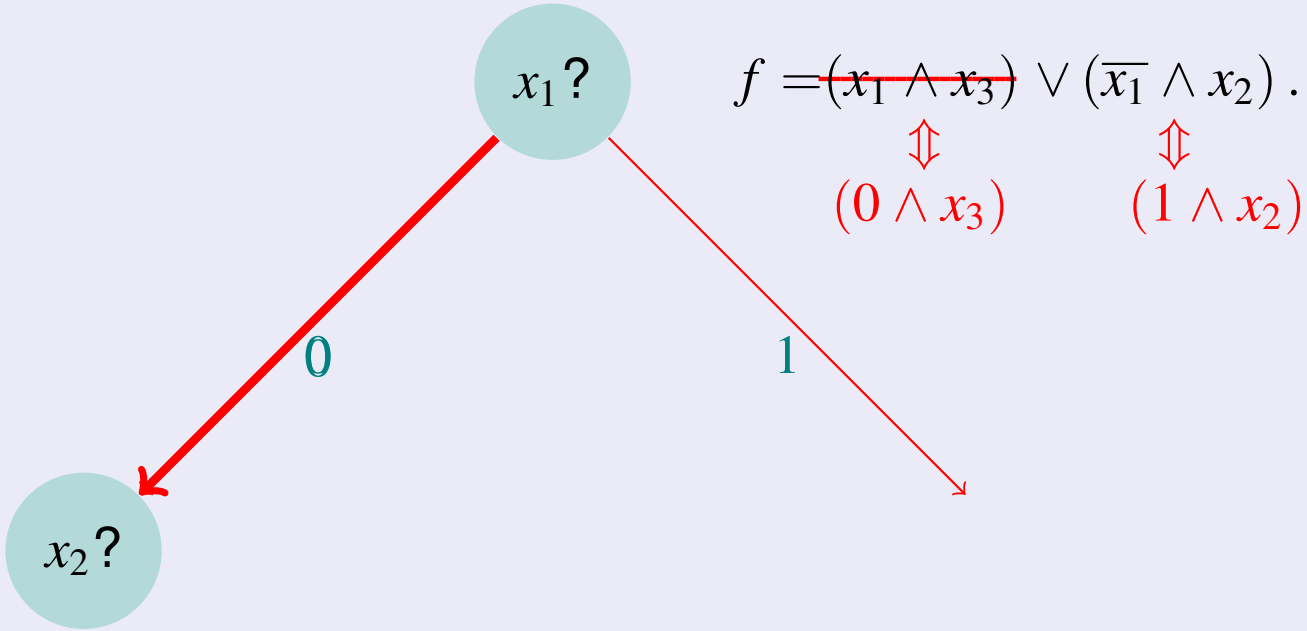
Decision tree: deterministic



$$f = \cancel{(x_1 \wedge x_3)} \vee (\overline{x_1} \wedge x_2)\,.$$

$$(0 \wedge x_3) \qquad (1 \wedge x_2)$$

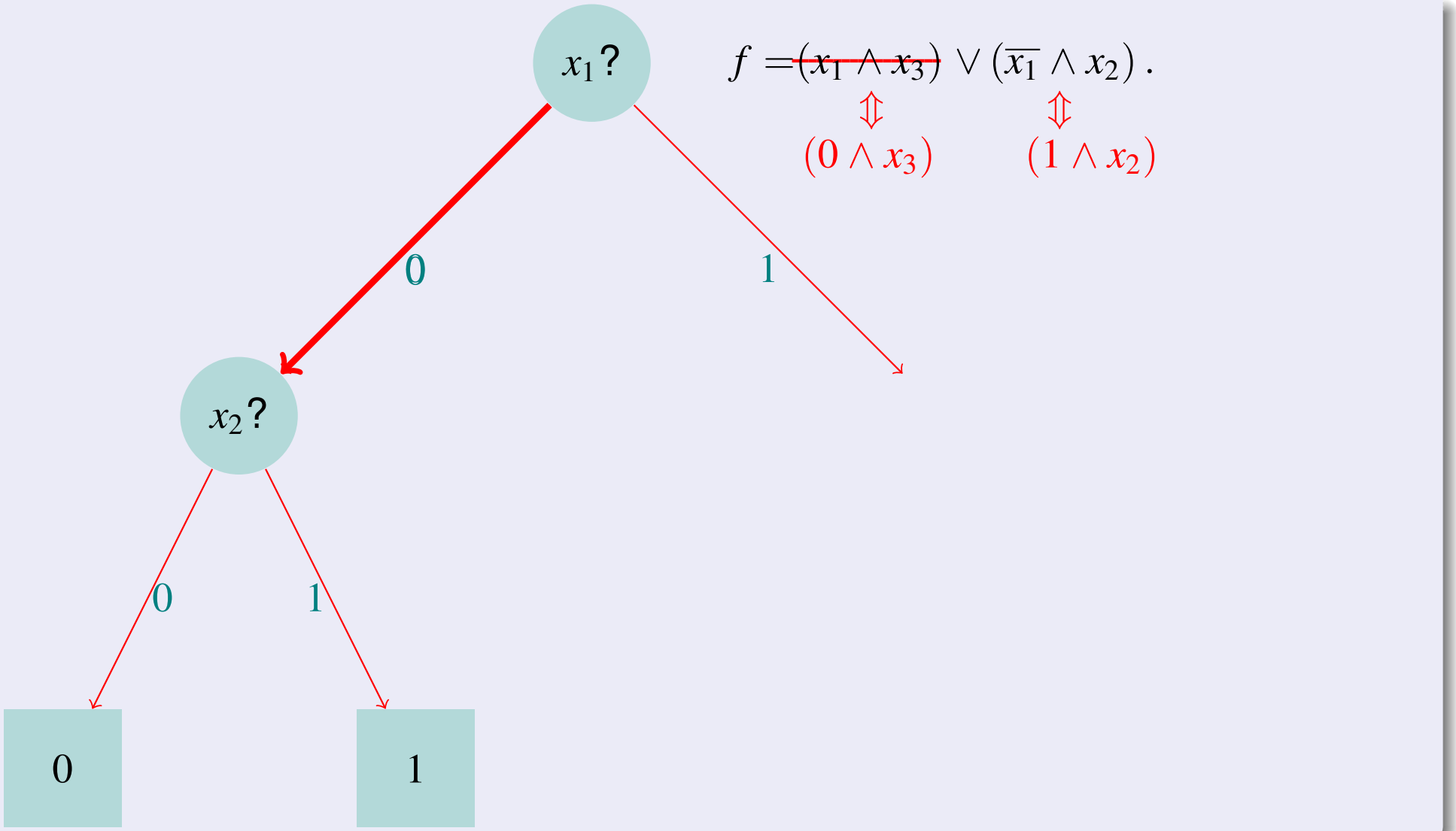
Decision tree: deterministic



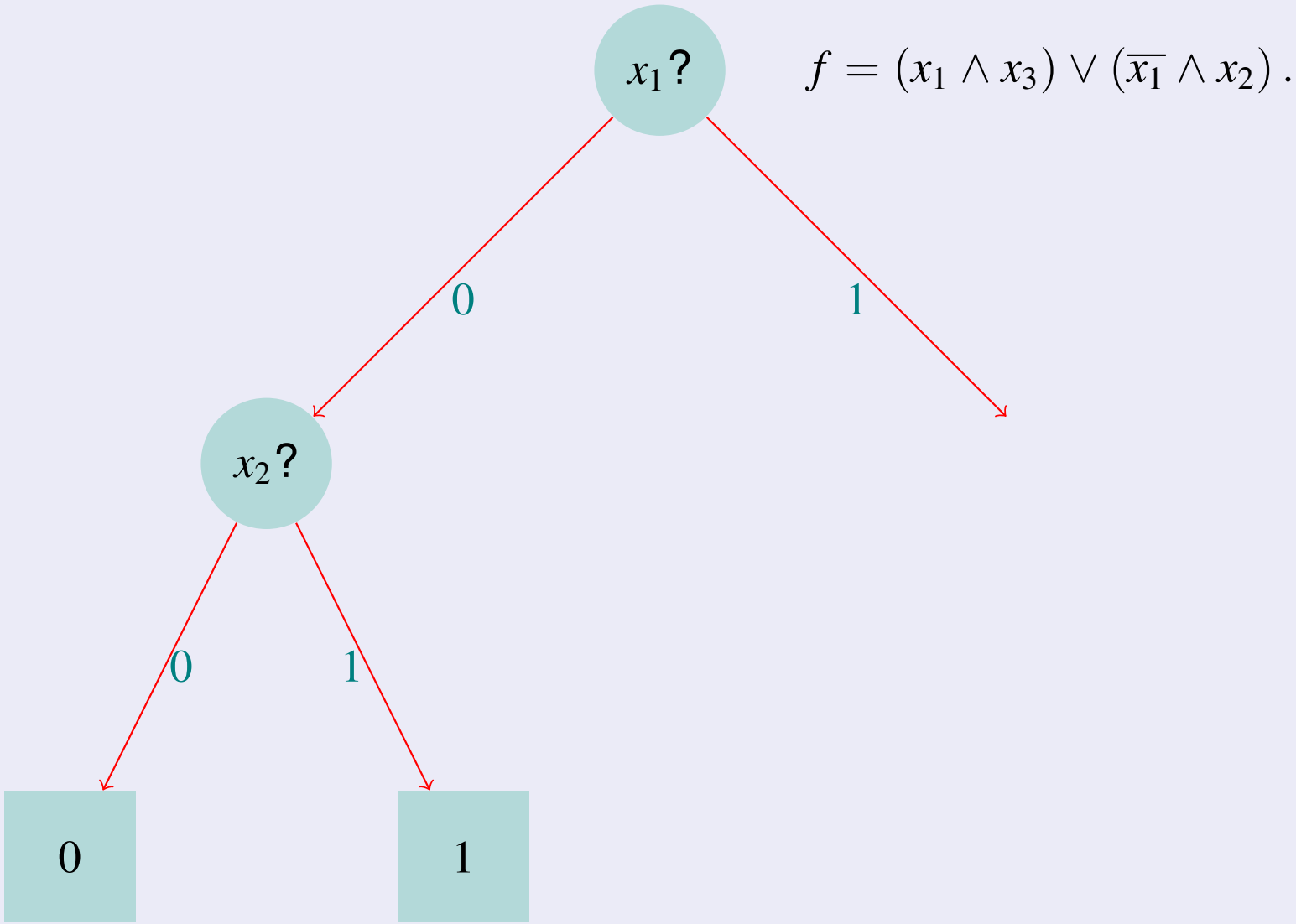
$$f = \cancel{(x_1 \wedge x_3)} \vee (\overline{x_1} \wedge x_2)\,.$$

$$(0 \wedge x_3) \qquad (1 \wedge x_2)$$

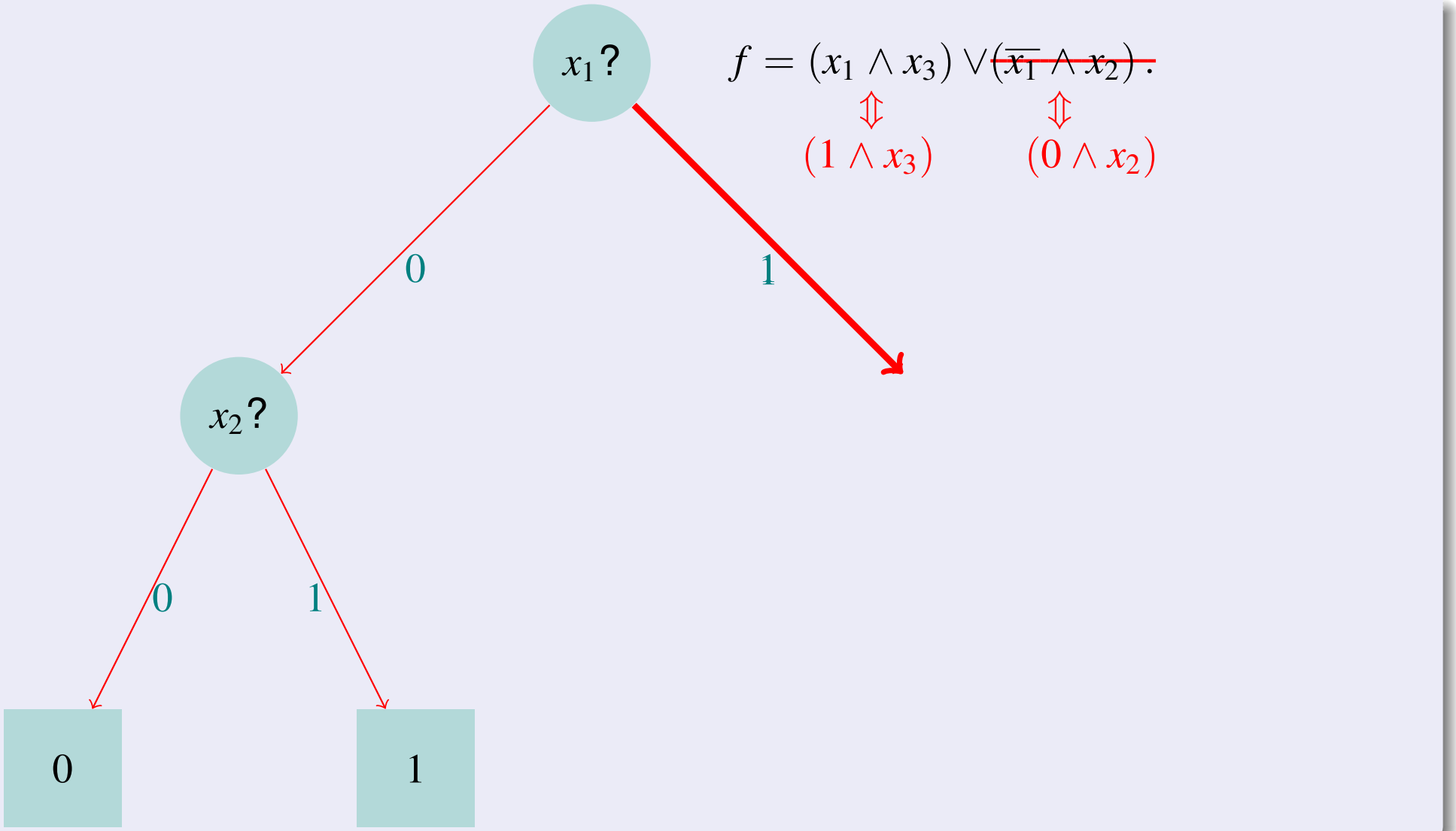
Decision tree: deterministic



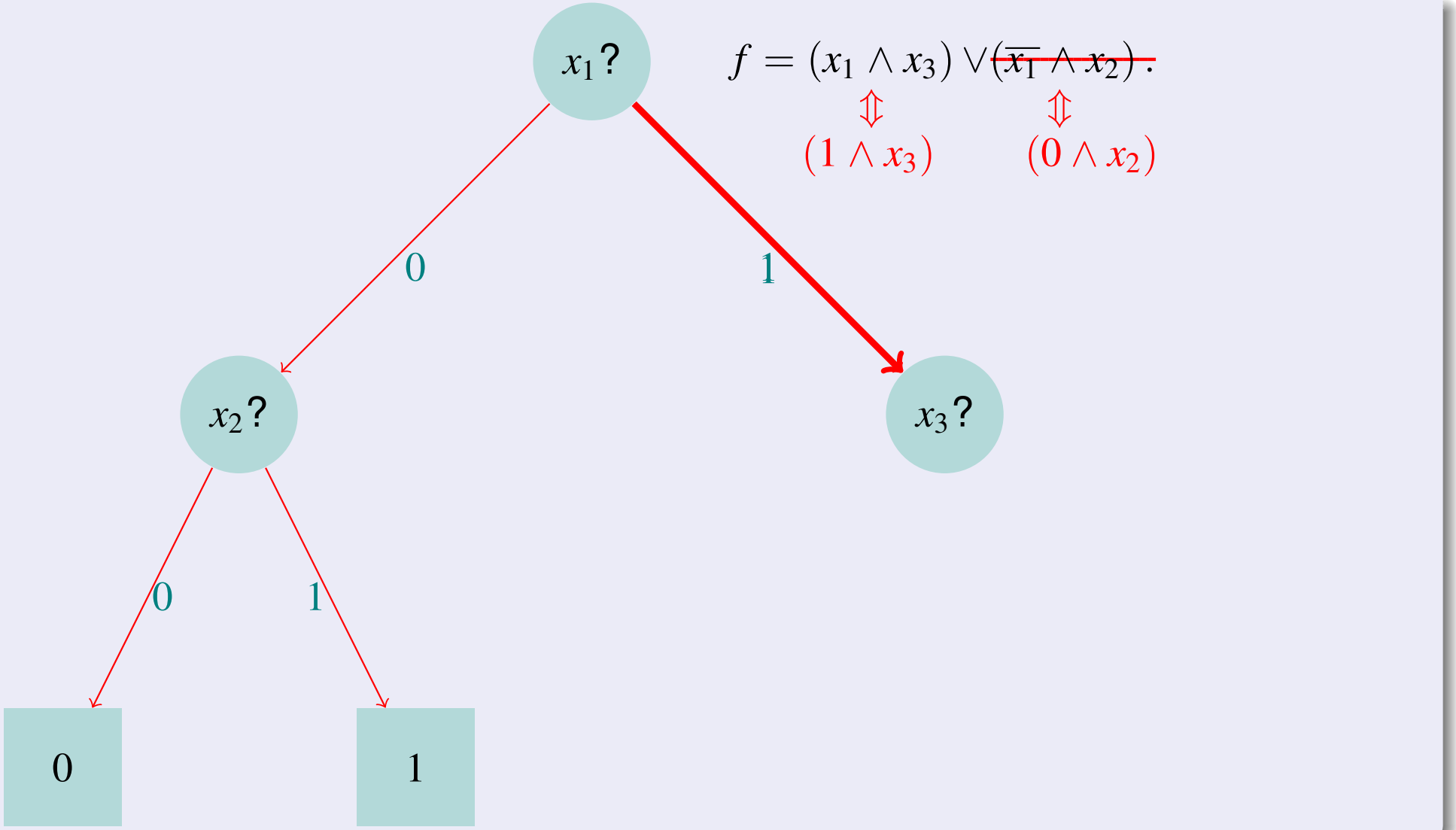
Decision tree: deterministic



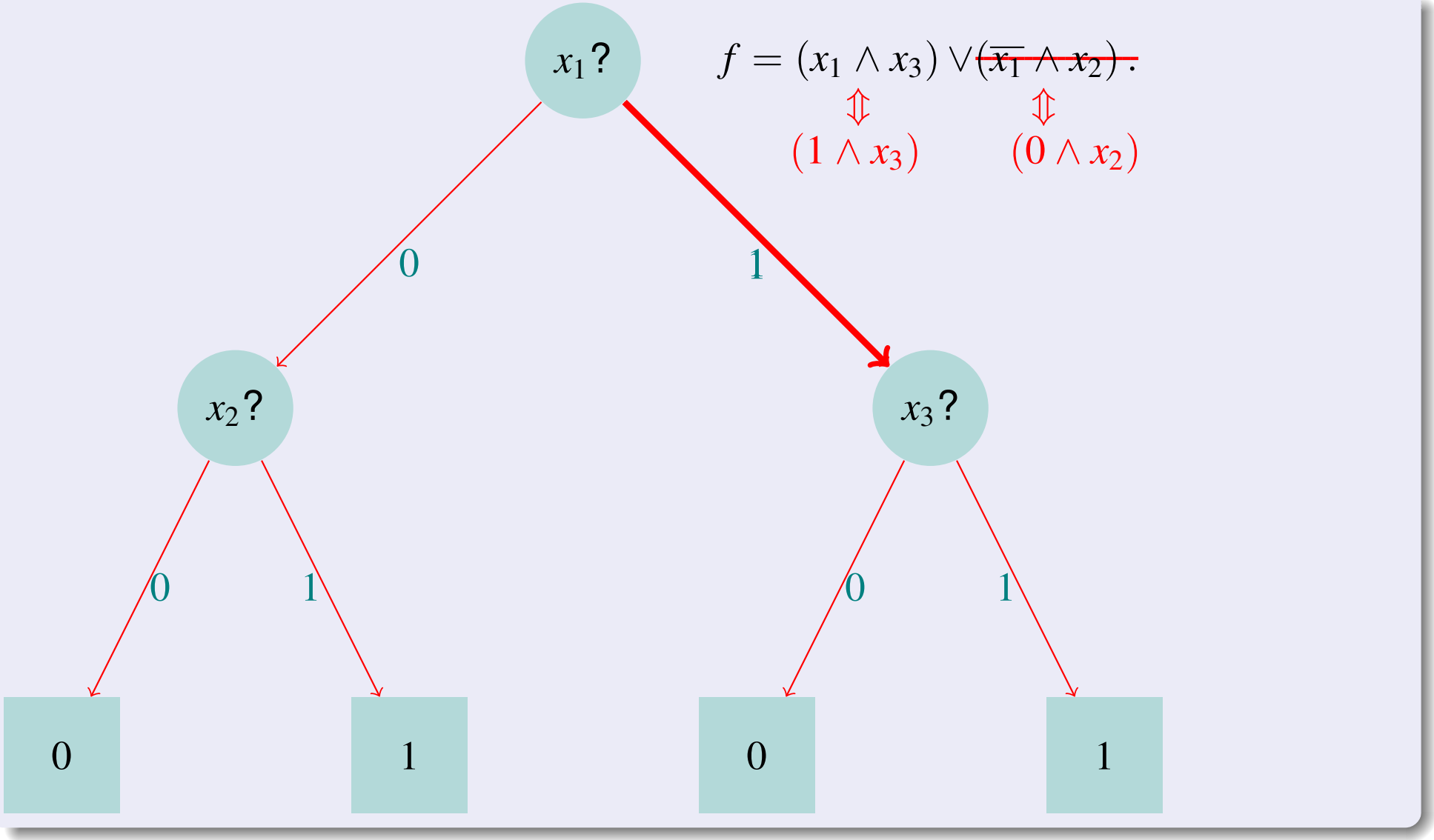
Decision tree: deterministic



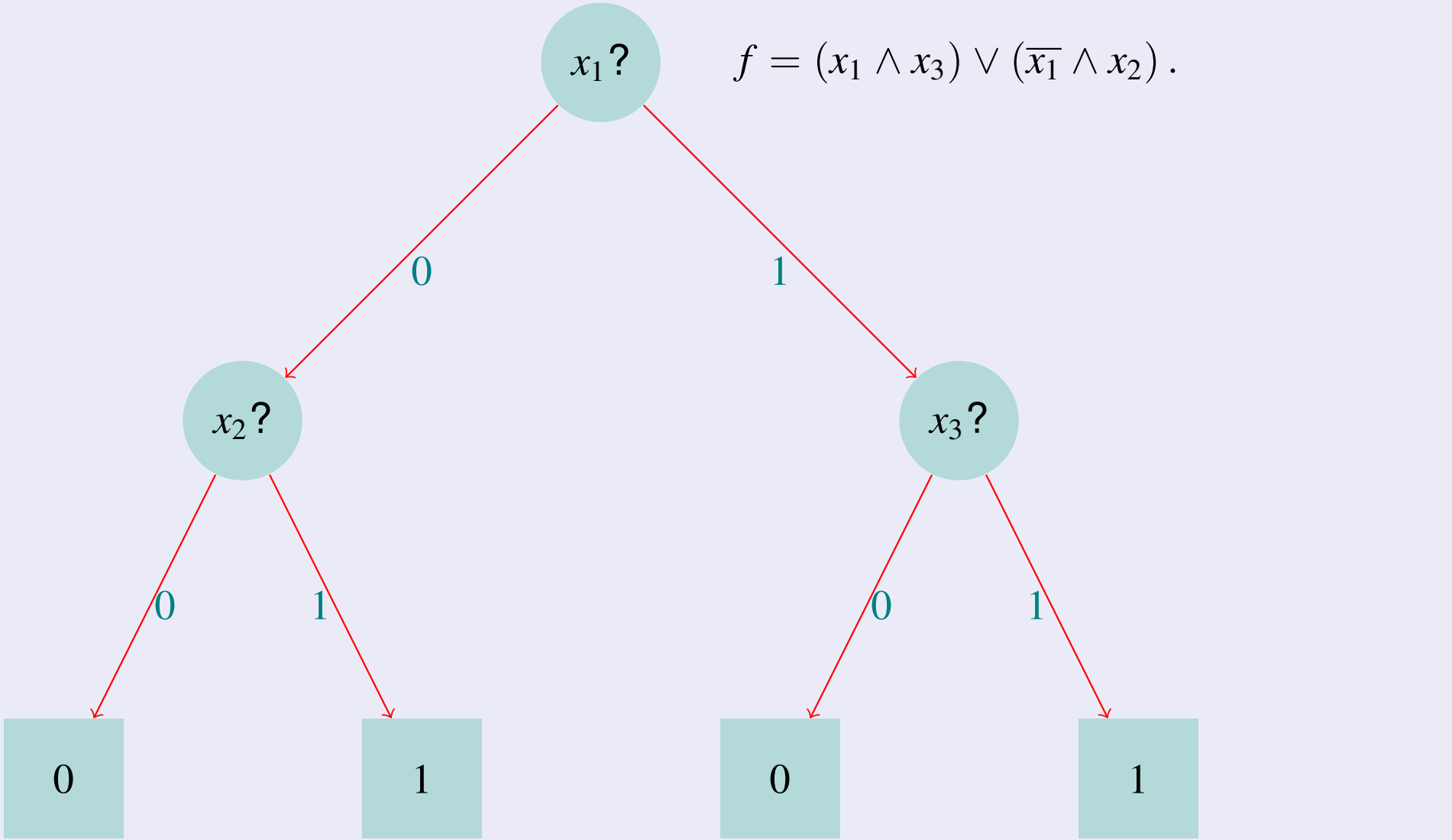
Decision tree: deterministic



Decision tree: deterministic



Decision tree: deterministic



$D(f) =$ depth of the shallowest decision tree that computes f.

Decision tree: randomized

- Algorithm has access to some source of randomness.

Decision tree: randomized

- Algorithm has access to some source of randomness.
- The next query depends on the past computation and the randomness.

Decision tree: randomized

- Algorithm has access to some source of randomness.
- The next query depends on the past computation and the randomness.
- Requirement: For every input x, $\Pr[f(x) = \text{output}(x)] \geq 1 - \epsilon$.

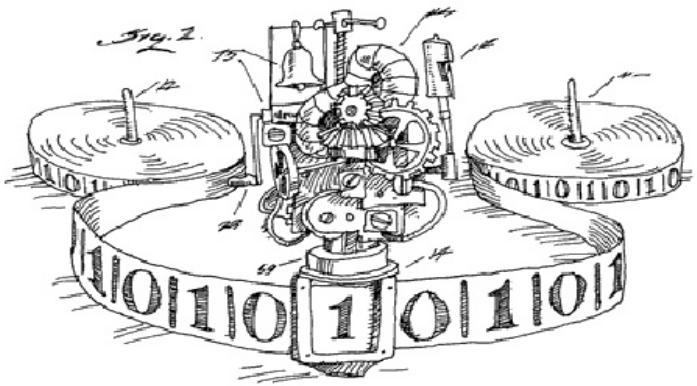
Decision tree: randomized

- Algorithm has access to some source of randomness.
- The next query depends on the past computation and the randomness.
- Requirement: For every input x, $\Pr[f(x) = \text{output}(x)] \geq 1 - \epsilon$.
- Complexity: Number of queries made in the worst case (over inputs and randomness).

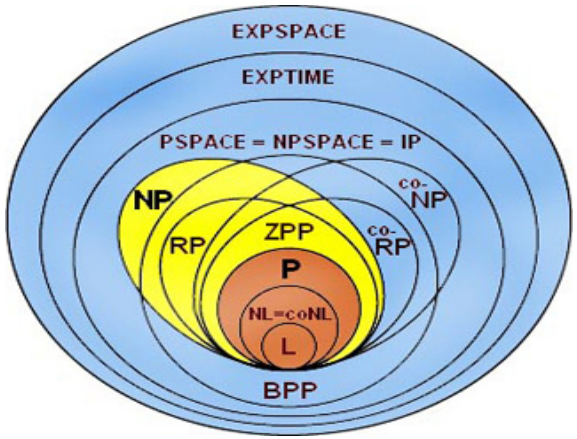
Decision tree: randomized

- Algorithm has access to some source of randomness.
- The next query depends on the past computation and the randomness.
- Requirement: For every input x, $\Pr[f(x) = \text{output}(x)] \geq 1 - \epsilon$.
- Complexity: Number of queries made in the worst case (over inputs and randomness).
- $R_\epsilon(f)$=Complexity of the best algorithm. $R(f) := R_{1/3}(f)$.

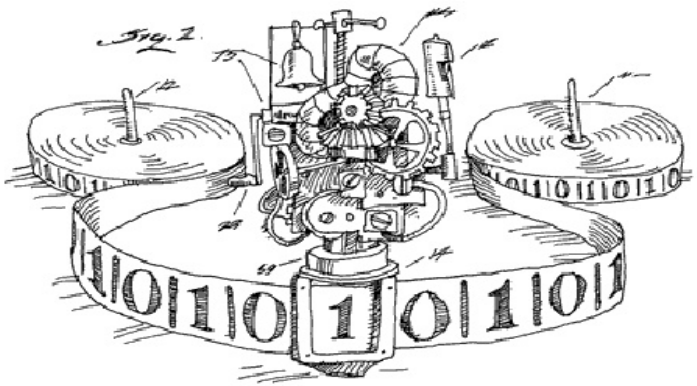
Motivation: Computational Complexity Theory



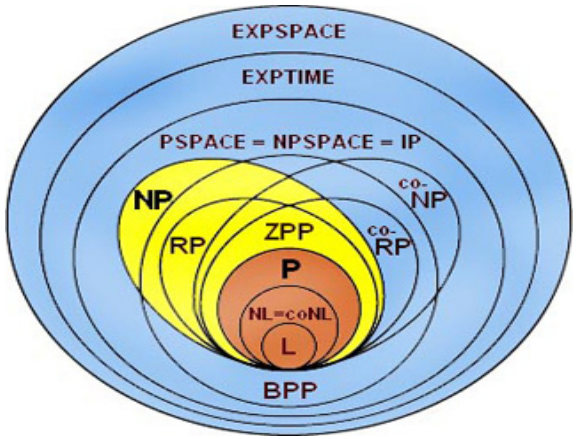
Turing machines

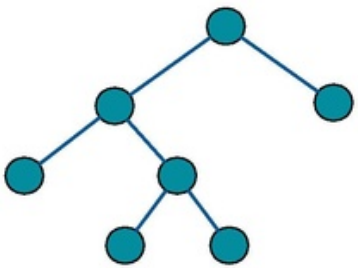


Motivation: Computational Complexity Theory



Turing machines





Simple model. Unconditional lower bounds.

Outline

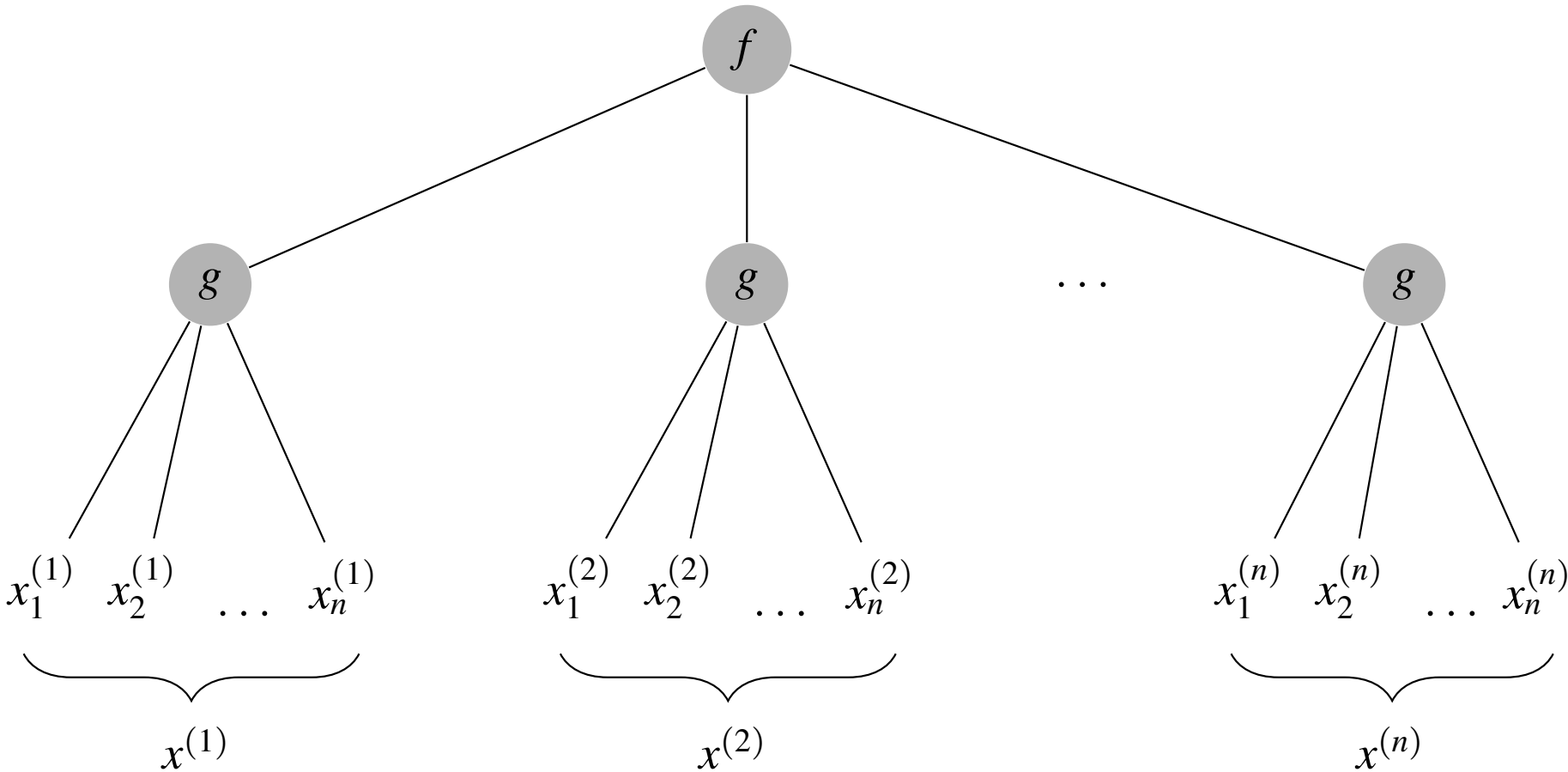
1. The query model
2. Composed functions
3. Our work

Composed functions

- $f, g : \{0,1\}^n \rightarrow \{0,1\}$.

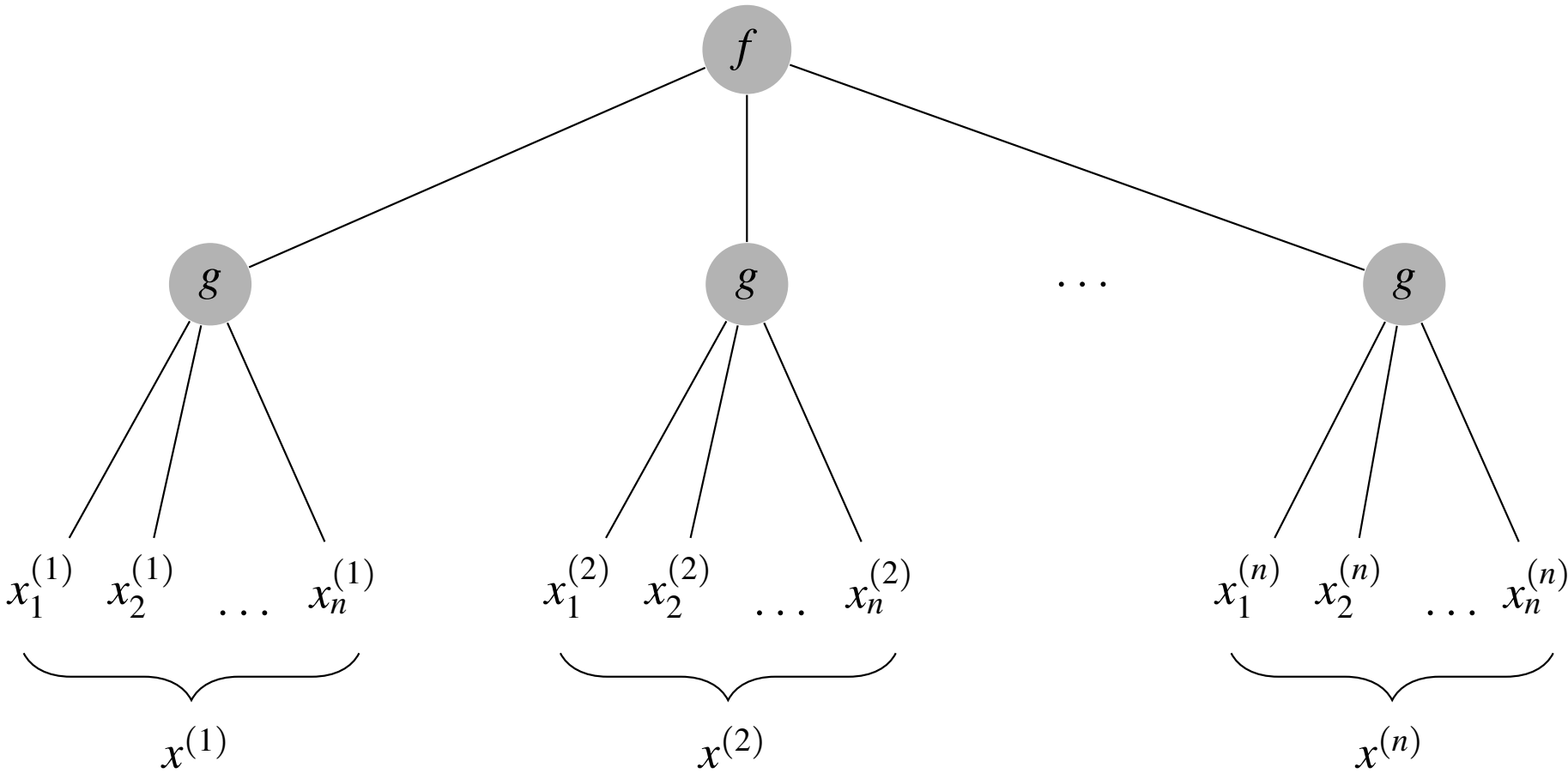
Composed functions

- $f, g : \{0,1\}^n \to \{0,1\}$.
- $f \circ g(x^{(1)}, \ldots, x^{(n)}) := f(g(x^{(1)}), \ldots, g(x^{(n)}))$.



Composed functions

- $f, g : \{0,1\}^n \to \{0,1\}$.
- $f \circ g(x^{(1)}, \ldots, x^{(n)}) := f(g(x^{(1)}), \ldots, g(x^{(n)}))$.

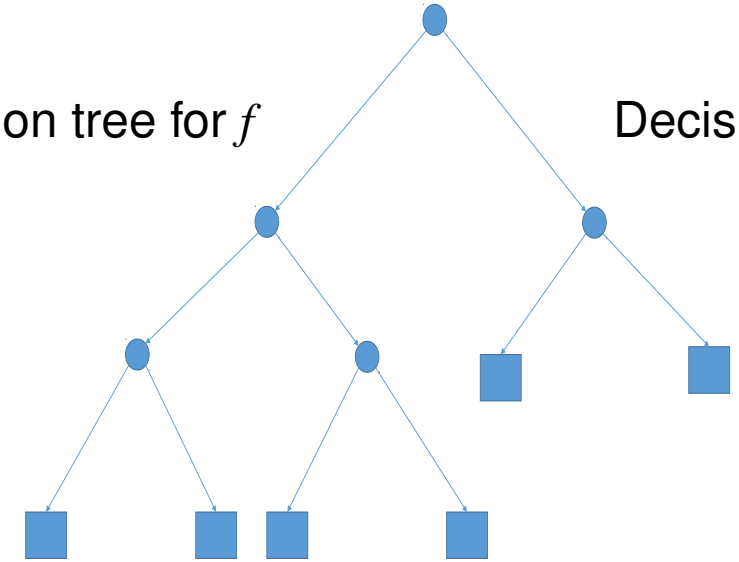


- Separating complexity measures, constructing hard functions.

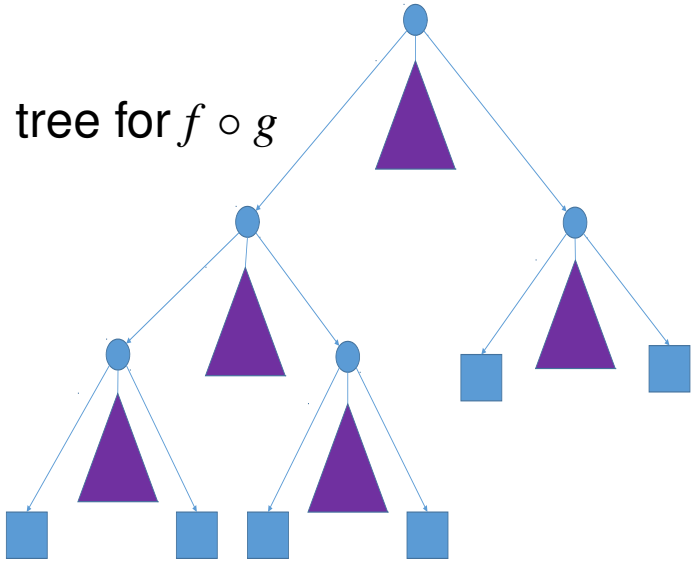
Composition questions

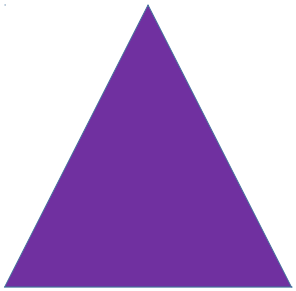
- $D(f \circ g) \leq D(f) \cdot D(g)$.

Decision tree for f



Decision tree for $f \circ g$





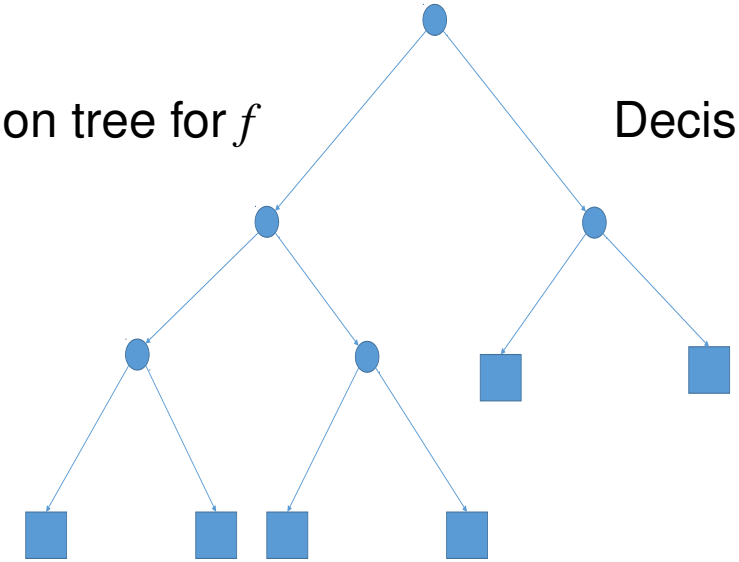
Decision tree for g

Simulate f's tree;
serve its queries
by simulating g's tree.

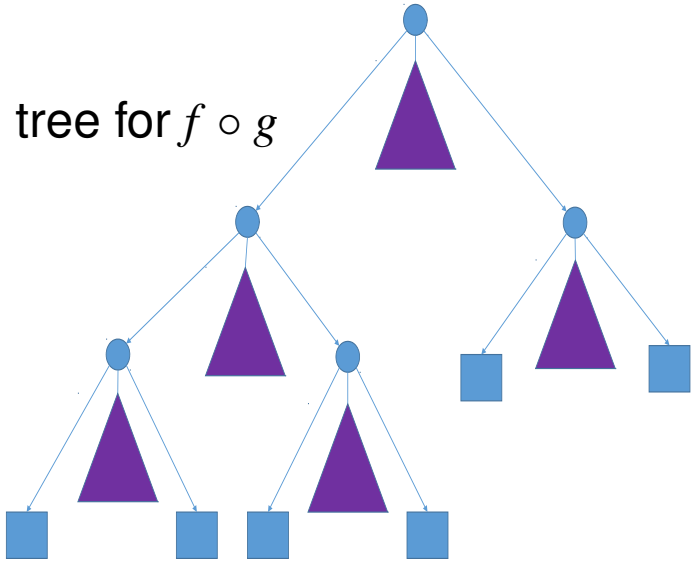
Composition questions

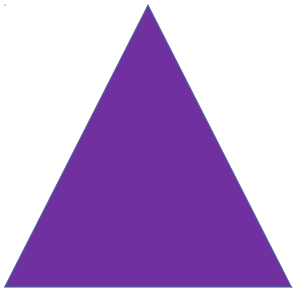
- $D(f \circ g) \leq D(f) \cdot D(g)$.

Decision tree for f



Decision tree for $f \circ g$





Decision tree for g

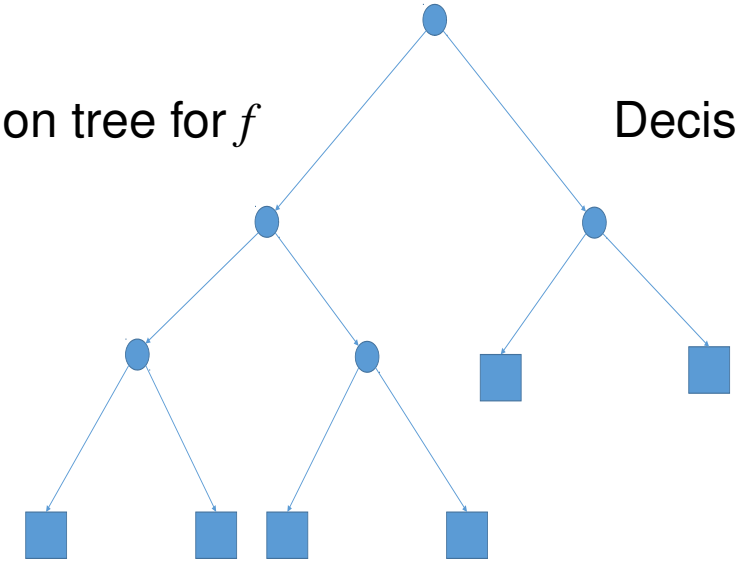
Simulate f's tree;
serve its queries
by simulating g's tree.

- $R(f \circ g) = \tilde{O}(R(f) \cdot R(g))$.

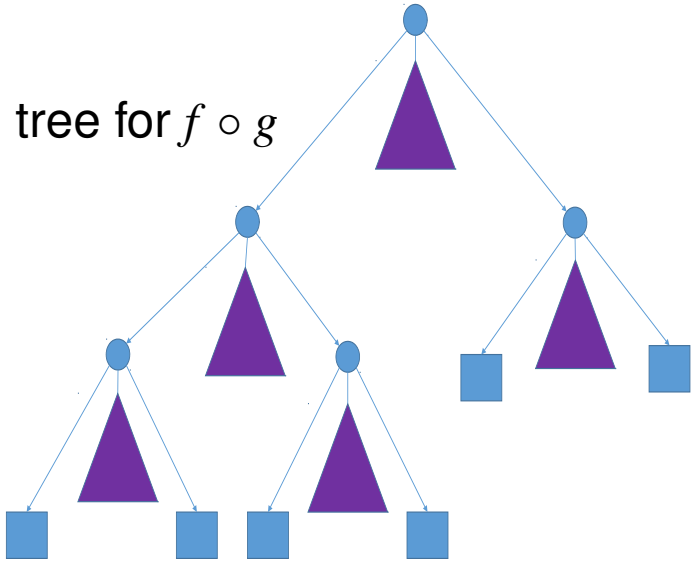
Composition questions

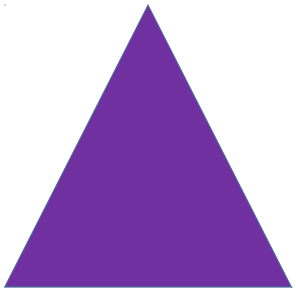
- $D(f \circ g) = D(f) \cdot D(g)$.

Decision tree for f



Decision tree for $f \circ g$





Decision tree for g

Simulate f's tree;
serve its queries
by simulating g's tree.

- $R(f \circ g) = \tilde{O}(R(f) \cdot R(g))$.

Composition questions

- Exact degree
- Approximate degree [Sherstov'12, Sherstov'13, Bun and Thaler'13]
- Communication complexity [Göös, Pitassi, Watson'15], [Göös, Pitassi, Watson'15], [Chattopadhyay et. al.'17], [Hatami, Hosseini, Lovett'16]
- Conical juntas [Göös, Jayram '16]

Outline

1. The query model
2. Composed functions
3. Our work

This work

Recall: $R(f \circ g) = \widetilde{O}(R(f) \cdot R(g))$.

This work

Recall: $R(f \circ g) = \widetilde{O}(R(f) \cdot R(g))$. Tight?

This work

Recall: $R(f \circ g) = \widetilde{O}(R(f) \cdot R(g))$. Tight?

- [AGJKLMSS'17] $R(f \circ g) = \Omega(R(f) \cdot R_{1/2-1/n^4}(g))$.
- [Ben-David and Kothari'16] $R(f \circ g) = \Omega(R(f)) \cdot \sqrt{R(g)})$ for total functions f, g.

This work

Main Theorems [Gavinsky, Lee, Santha, S.]

Let f be any relation, and g be any partial function. Then,

$$R(f \circ g) = \Omega(R(f) \cdot \sqrt{R(g)}).$$

Tight in this generality: there exists a relation f and a partial function g such that $R(f \circ g) = O(R(f) \cdot \sqrt{R(g)})$.

This work

Main Theorems [Gavinsky, Lee, Santha, S.]

Let f be any relation, and g be any partial function. Then,

$$R(f \circ g) = \Omega(R(f) \cdot \sqrt{R(g)}).$$

Tight in this generality: there exists a relation f and a partial function g such that $R(f \circ g) = O(R(f) \cdot \sqrt{R(g)})$.

- Proved via *Max-Conflict complexity*.

The tight example

g is the following Gap-Hamming problem on n bits:

$$g(x) = \begin{cases} 0 & \text{if } |x| \leq \frac{n}{2} - 10\sqrt{n}, \\ 1 & \text{if } |x| \geq \frac{n}{2} + 10\sqrt{n}. \end{cases}$$

The tight example

g is the following Gap-Hamming problem on n bits:

$$g(x) = \begin{cases} 0 & \text{if } |x| \leq \frac{n}{2} - 10\sqrt{n}, \\ 1 & \text{if } |x| \geq \frac{n}{2} + 10\sqrt{n}. \end{cases}$$

f is the following relation. Let x be an n-bit string. $H(\cdot)$ is the Hamming distance.

$$f(x) = \{y \in \{0,1\}^n \mid H(x,y) \leq \frac{n}{2} - \frac{\sqrt{n}}{10}\}$$

Query complexity of g is $\Theta(n)$:

Recall:

$$g(x) = \begin{cases} 0 & \text{if } |x| \leq \frac{n}{2} - 10\sqrt{n}, \\ 1 & \text{if } |x| \geq \frac{n}{2} + 10\sqrt{n}. \end{cases}$$

Query complexity of g is $\Theta(n)$:

Recall:

$$g(x) = \begin{cases} 0 & \text{if } |x| \leq \frac{n}{2} - 10\sqrt{n}, \\ 1 & \text{if } |x| \geq \frac{n}{2} + 10\sqrt{n}. \end{cases}$$

- $\Pr[\text{A random input bit} = \text{output}] \approx \frac{1}{2} + \frac{1}{\Omega(\sqrt{n})}$.

Query complexity of g is $\Theta(n)$:

Recall:

$$g(x) = \begin{cases} 0 & \text{if } |x| \leq \frac{n}{2} - 10\sqrt{n}, \\ 1 & \text{if } |x| \geq \frac{n}{2} + 10\sqrt{n}. \end{cases}$$

- $\Pr[\text{A random input bit} = \text{output}] \approx \frac{1}{2} + \frac{1}{\Omega(\sqrt{n})}$.
- $\Rightarrow$ a random input bit carries $\approx \frac{1}{\Omega(n)}$ bits of information about g.

Query complexity of g is $\Theta(n)$:

Recall:

$$g(x) = \begin{cases} 0 & \text{if } |x| \leq \frac{n}{2} - 10\sqrt{n}, \\ 1 & \text{if } |x| \geq \frac{n}{2} + 10\sqrt{n}. \end{cases}$$

- $\Pr[\text{A random input bit} = \text{output}] \approx \frac{1}{2} + \frac{1}{\Omega(\sqrt{n})}$.
- $\Rightarrow$ a random input bit carries $\approx \frac{1}{\Omega(n)}$ bits of information about g.
- $\Rightarrow$ Any algorithm must query $\Omega(n)$ input bits to infer g with non-trivial probability.

Query complexity of f is $\Theta(\sqrt{n})$

Recall: $f : x \rightarrow$ any y such that $|\{i : y_i = x_i\}| \geq \frac{n}{2} + \frac{\sqrt{n}}{10}$.

Query complexity of f is $\Theta(\sqrt{n})$

Recall: $f : x \to$ any y such that $|\{i : y_i = x_i\}| \geq \frac{n}{2} + \frac{\sqrt{n}}{10}$.

Query algorithm for f:

1. Query $x_1, \ldots, x_{50\sqrt{n}}$.

Query complexity of f is $\Theta(\sqrt{n})$

Recall: $f : x \to$ any y such that $|\{i : y_i = x_i\}| \geq \frac{n}{2} + \frac{\sqrt{n}}{10}$.

Query algorithm for f:

1. Query $x_1, \ldots, x_{50\sqrt{n}}$.
2. Sample $z_{50\sqrt{n}+1}, \ldots, z_n$ uniformly at random.
3. Return $x_1, \ldots, x_{50\sqrt{n}}, z_{50\sqrt{n}+1}, \ldots, z_n$.

Query complexity of f is $\Theta(\sqrt{n})$

Recall: $f : x \to$ any y such that $|\{i : y_i = x_i\}| \geq \frac{n}{2} + \frac{\sqrt{n}}{10}$.

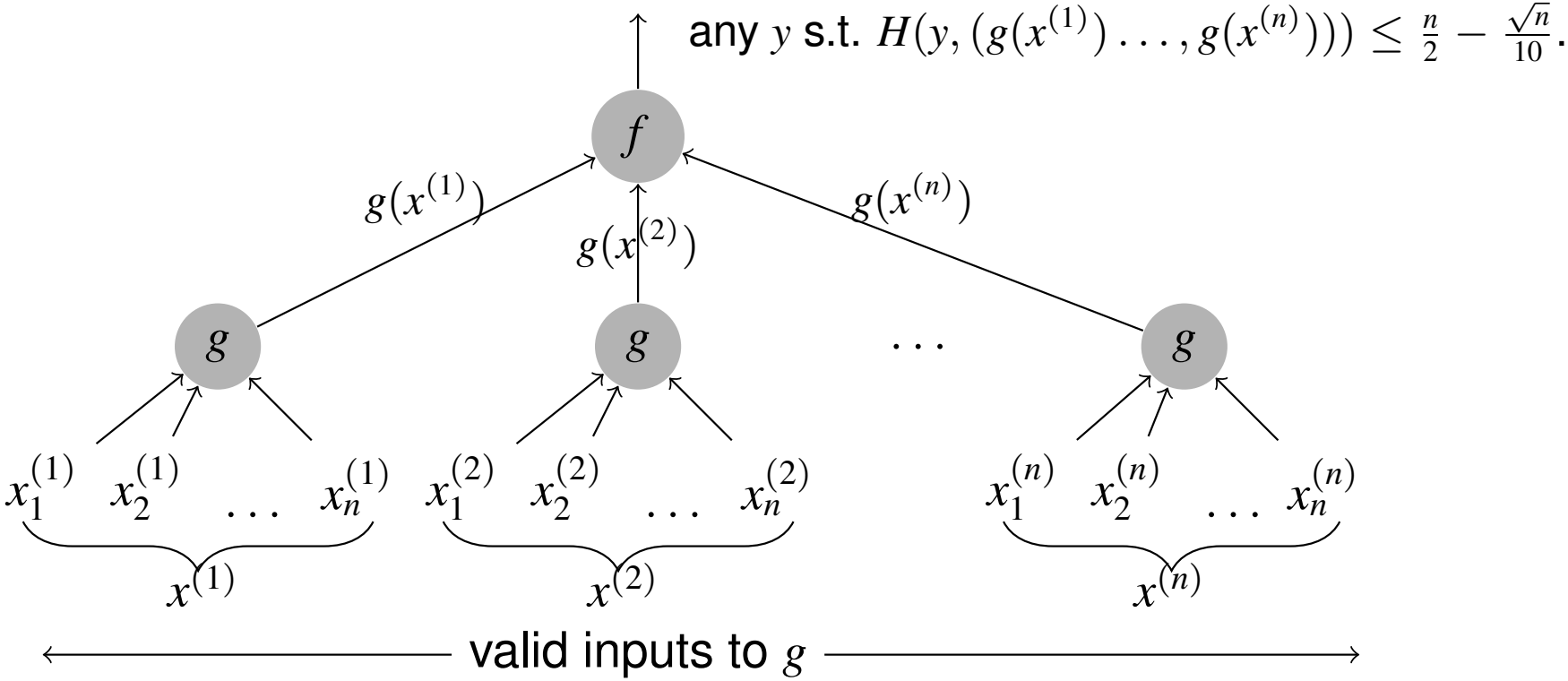
Query algorithm for f:

1. Query $x_1, \ldots, x_{50\sqrt{n}}$.
2. Sample $z_{50\sqrt{n}+1}, \ldots, z_n$ uniformly at random.
3. Return $x_1, \ldots, x_{50\sqrt{n}}, z_{50\sqrt{n}+1}, \ldots, z_n$.

Optimal. Reason: With $\frac{1}{2}$ probability a random bit vector matches x in at most $\frac{n}{2}$ locations.

$f \circ g$

g: Gap-Hamming.



Query complexity of $f \circ g$ is $O(n)$

Recall: Input to $f \circ g$ is a n^2 bit string $x^{(1)}, \ldots, x^{(n)}$, where each $x^{(i)} = (x_1^{(i)}, \ldots x_n^{(i)})$.

Query complexity of $f \circ g$ is $O(n)$

Recall: Input to $f \circ g$ is a n^2 bit string $x^{(1)}, \ldots, x^{(n)}$, where each $x^{(i)} = (x_1^{(i)}, \ldots x_n^{(i)})$.

Query algorithm for $f \circ g^n$:

Query complexity of $f \circ g$ is $O(n)$

Recall: Input to $f \circ g$ is a n^2 bit string $x^{(1)}, \ldots, x^{(n)}$, where each $x^{(i)} = (x_1^{(i)}, \ldots x_n^{(i)})$.

Query algorithm for $f \circ g^n$:

1. For each i, draw $z^{(i)}$ uniformly at random from among $\{x_1^{(i)}, \ldots, x_n^{(i)}\}$.

Query complexity of $f \circ g$ is $O(n)$

Recall: Input to $f \circ g$ is a n^2 bit string $x^{(1)}, \ldots, x^{(n)}$, where each $x^{(i)} = (x_1^{(i)}, \ldots x_n^{(i)})$.

Query algorithm for $f \circ g^n$:

1. For each i, draw $z^{(i)}$ uniformly at random from among $\{x_1^{(i)}, \ldots, x_n^{(i)}\}$.
2. Return $z^{(1)}, \ldots, z^{(n)}$.

Query complexity of $f \circ g$ is $O(n)$

Recall: Input to $f \circ g$ is a n^2 bit string $x^{(1)}, \ldots, x^{(n)}$, where each $x^{(i)} = (x_1^{(i)}, \ldots x_n^{(i)})$.

Query algorithm for $f \circ g^n$:

1. For each i, draw $z^{(i)}$ uniformly at random from among $\{x_1^{(i)}, \ldots, x_n^{(i)}\}$.
2. Return $z^{(1)}, \ldots, z^{(n)}$.

Analysis: Recall that the $g(x^{(i)})$'s are inputs to the top f. Each $z^{(i)}$ is independently equal to $g(x^{(i)})$ with probability at least $\frac{1}{2} + \frac{10}{\sqrt{n}}$. Then use Chernoff bound.

Query complexity of $f \circ g$ is $O(n)$

Recall: Input to $f \circ g$ is a n^2 bit string $x^{(1)}, \ldots, x^{(n)}$, where each $x^{(i)} = (x_1^{(i)}, \ldots x_n^{(i)})$.

Query algorithm for $f \circ g^n$:

1. For each i, draw $z^{(i)}$ uniformly at random from among $\{x_1^{(i)}, \ldots, x_n^{(i)}\}$.
2. Return $z^{(1)}, \ldots, z^{(n)}$.

Analysis: Recall that the $g(x^{(i)})$'s are inputs to the top f. Each $z^{(i)}$ is independently equal to $g(x^{(i)})$ with probability at least $\frac{1}{2} + \frac{10}{\sqrt{n}}$. Then use Chernoff bound.

$$R(f \circ g) = O(R(f) \cdot \sqrt{R(g)}).$$

Reflection

How did we beat the ‘trivial’ algorithm?

Reflection

How did we beat the 'trivial' algorithm?

- Solve copies of g with advantage $\frac{1}{2} + o(1)$. Takes fewer queries.

Reflection

How did we beat the 'trivial' algorithm?

- Solve copies of g with advantage $\frac{1}{2} + o(1)$. Takes fewer queries.
- Combine those 'noisy' responses to compute f. Takes more queries than $R(f)$.

Reflection

How did we beat the 'trivial' algorithm?

- Solve copies of g with advantage $\frac{1}{2} + o(1)$. Takes fewer queries.
- Combine those 'noisy' responses to compute f. Takes more queries than $R(f)$.
- Saves queries overall.

Reflection

How did we beat the 'trivial' algorithm?

- Solve copies of g with advantage $\frac{1}{2} + o(1)$. Takes fewer queries.
- Combine those 'noisy' responses to compute f. Takes more queries than $R(f)$.
- Saves queries overall.

Next question: What happens if both f and g are functions?

Lower bound

$$R(f \circ g) = \Omega(R(f) \cdot \sqrt{R(g)}).$$

Proof Outline.

Proof Outline

- Recall statement: $R(f) = O(R(f \circ g)/\sqrt{R(g)})$.

Proof Outline

- Recall statement: $R(f) = O(R(f \circ g)/\sqrt{R(g)})$.
- $R(f \circ g)$ small and $R(g)$ large $\Rightarrow$ efficient query algorithm for f.

Proof Outline

- Recall statement: $R(f) = O(R(f \circ g)/\sqrt{R(g)})$.
- $R(f \circ g)$ small and $R(g)$ large $\Rightarrow$ efficient query algorithm for f.
- $R(f \circ g)$ small $\Rightarrow$ Efficient algorithm $\mathcal{A}'$ for $f \circ g$.

Proof Outline

- Recall statement: $R(f) = O(R(f \circ g)/\sqrt{R(g)})$.
- $R(f \circ g)$ small and $R(g)$ large $\Rightarrow$ efficient query algorithm for f.
- $R(f \circ g)$ small $\Rightarrow$ Efficient algorithm $\mathcal{A}'$ for $f \circ g$.
- $R(g)$ large $\Rightarrow$?

Proof Outline

- Recall statement: $R(f) = O(R(f \circ g)/\sqrt{R(g)})$.
- $R(f \circ g)$ small and $R(g)$ large $\Rightarrow$ efficient query algorithm for f.
- $R(f \circ g)$ small $\Rightarrow$ Efficient algorithm $\mathcal{A}'$ for $f \circ g$.
- $R(g)$ large $\Rightarrow$?
- Suitably chosen probability distributions $\mu_0 \sim g^{-1}(0), \mu_1 \sim g^{-1}(1)$.

Proof Outline

- Recall statement: $R(f) = O(R(f \circ g)/\sqrt{R(g)})$.
- $R(f \circ g)$ small and $R(g)$ large $\Rightarrow$ efficient query algorithm for f.
- $R(f \circ g)$ small $\Rightarrow$ Efficient algorithm $\mathcal{A}'$ for $f \circ g$.
- $R(g)$ large $\Rightarrow$?
- Suitably chosen probability distributions $\mu_0 \sim g^{-1}(0), \mu_1 \sim g^{-1}(1)$.
- Use $\mu_0, \mu_1, \mathcal{A}'$ to design efficient algorithm $\mathcal{A}$ for f.

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

- Input: $z = (z_1, \ldots, z_n) \in \{0,1\}^n$.

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

- Input: $z = (z_1, \ldots, z_n) \in \{0,1\}^n$.
- Sample $x \in \{0,1\}^{n^2}$ such that $f(z) = f \circ g(x)$.

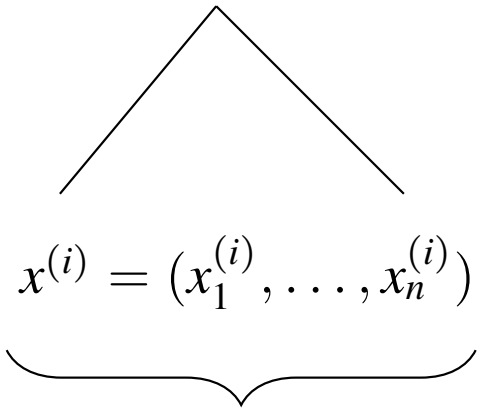
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

- Input: $z = (z_1, \ldots, z_n) \in \{0,1\}^n$.
- Sample $x \in \{0,1\}^{n^2}$ such that $f(z) = f \circ g(x)$.
- Idea: Run $\mathcal{A}'$ on x and return.

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

- Input: $z = (z_1, \ldots, z_n) \in \{0,1\}^n$.
- Sample $x \in \{0,1\}^{n^2}$ such that $f(z) = f \circ g(x)$.

$$z = (z_1, \ldots, z_i, \ldots, z_n)$$



$$x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$$

Sampled from μ_{z_i}

- Idea: Run $\mathcal{A}'$ on x and return.

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

- Input: $z = (z_1, \ldots, z_n) \in \{0,1\}^n$.
- Sample $x \in \{0,1\}^{n^2}$ such that $f(z) = f \circ g(x)$.

$$z = (z_1, \ldots, z_i, \ldots, z_n)$$

$$(x^{(1)}, \quad \ldots, \quad \underbrace{x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})}, \quad \ldots, \quad x^{(n)})$$

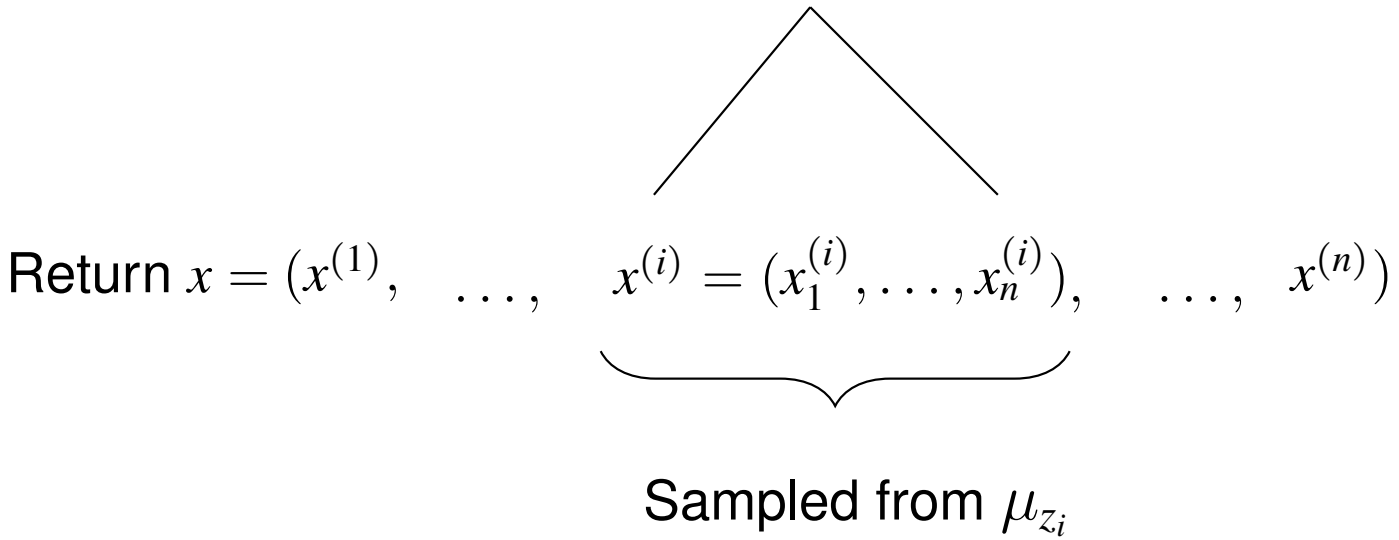
Sampled from μ_{z_i}

- Idea: Run $\mathcal{A}'$ on x and return.

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

- Input: $z = (z_1, \ldots, z_n) \in \{0,1\}^n$.
- Sample $x \in \{0,1\}^{n^2}$ such that $f(z) = f \circ g(x)$.

$$z = (z_1, \ldots, z_i, \ldots, z_n)$$



- Idea: Run $\mathcal{A}'$ on x and return.

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

- Input: $z = (z_1, \ldots, z_n) \in \{0,1\}^n$.
- Sample $x \in \{0,1\}^{n^2}$ such that $f(z) = f \circ g(x)$.

$$z = (z_1, \ldots, z_i, \ldots, z_n)$$

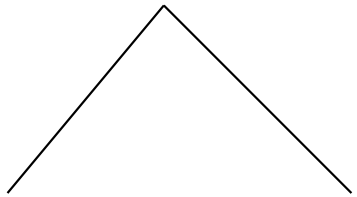
Return $x = (x^{(1)}, \quad \ldots, \quad \underbrace{x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})}_{\text{Sampled from } \mu_{z_i}}, \quad \ldots, \quad x^{(n)})$

- Note: $f \circ g(x) = f(z)$.
- Idea: Run $\mathcal{A}'$ on x and return.

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

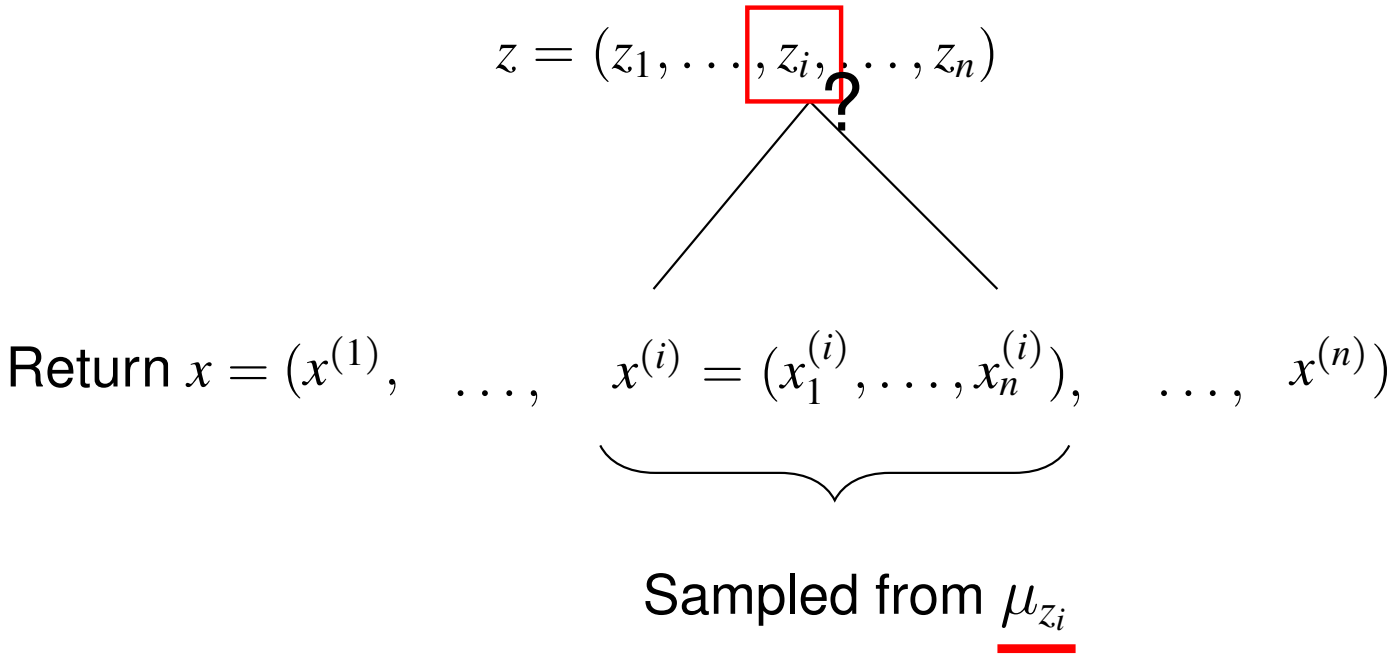
$$z = (z_1, \ldots, z_i, \ldots, z_n)$$



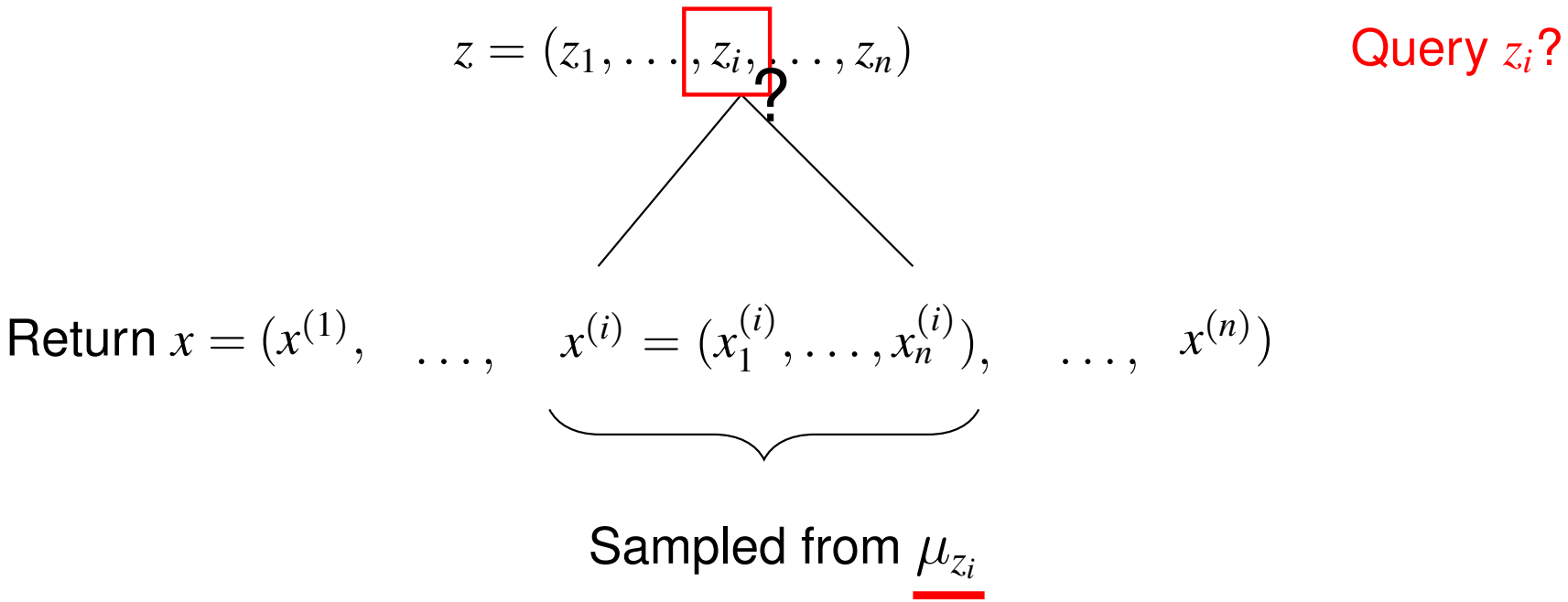
Return $x = (x^{(1)}, \quad \ldots, \quad \underbrace{x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})}, \quad \ldots, \quad x^{(n)})$

Sampled from μ_{z_i}

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

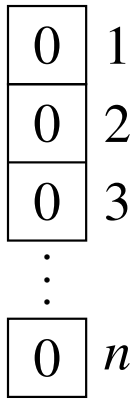


Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f



Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

$\mathcal{A}'$: algorithm for $f \circ g$

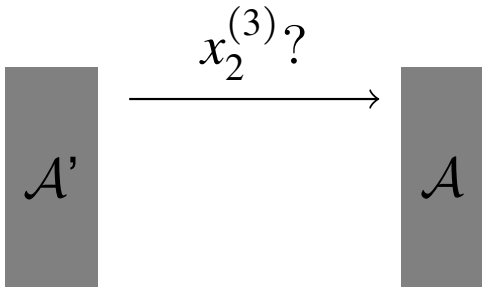


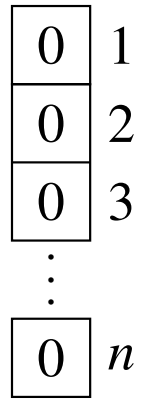
Query register

Input: $z_1 \mid z_2 \mid z_3 \mid \cdots \mid z_n$

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

$\mathcal{A}'$: algorithm for $f \circ g$





Query register

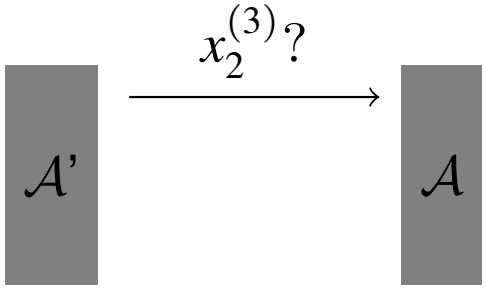
Input: z_1 z_2 z_3 $\cdots$ z_n

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

$\mathcal{A}'$: algorithm for $f \circ g$

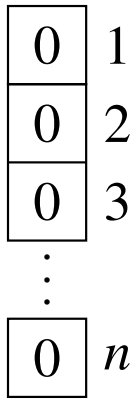
	$\Pr[x_2^{(3)} = 0]$	$\Pr[x_2^{(3)} = 1]$
μ_0	0.4	0.6
μ_1	0.65	0.35

to be drawn from μ_{z_3}



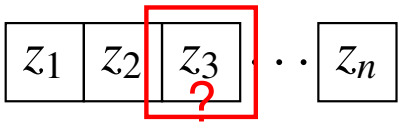
μ_0 : 0.4 0.6

μ_1 : 0.65 0.35



Query register

Input:

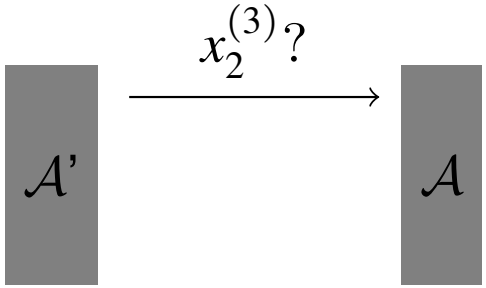


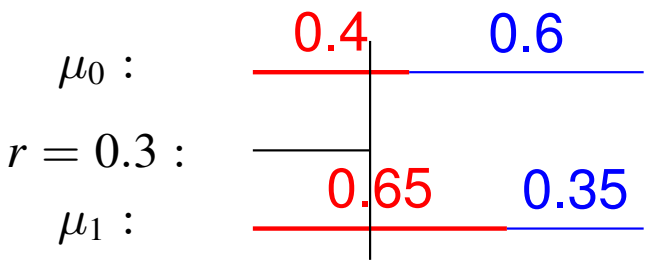
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

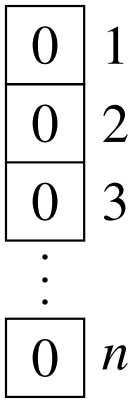
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_2^{(3)} = 0]$	$\Pr[x_2^{(3)} = 1]$
μ_0	0.4	0.6
μ_1	0.65	0.35

to be drawn from μ_{z_3}

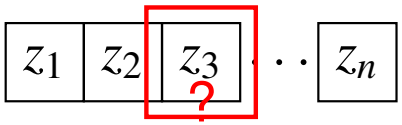






Query register

Input:

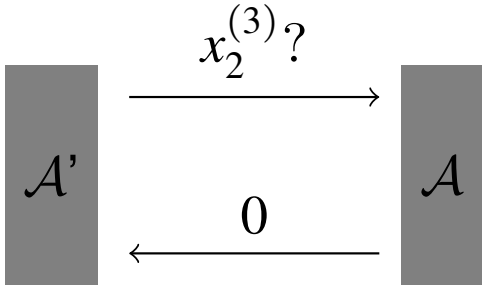


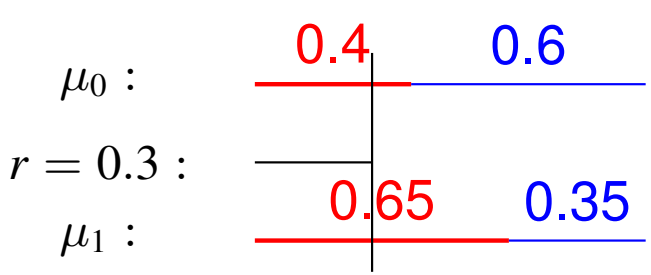
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

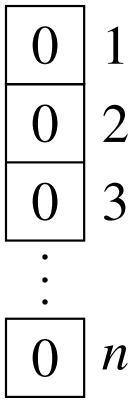
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_2^{(3)} = 0]$	$\Pr[x_2^{(3)} = 1]$
μ_0	0.4	0.6
μ_1	0.65	0.35

to be drawn from μ_{z_3}





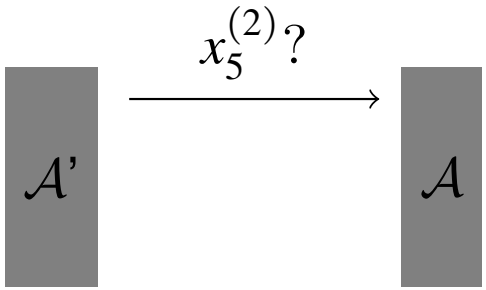


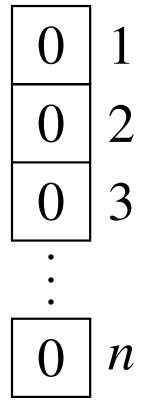
Query register

Input: $z_1 \mid z_2 \mid z_3 \mid \cdots \mid z_n$

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

$\mathcal{A}'$: algorithm for $f \circ g$





Query register

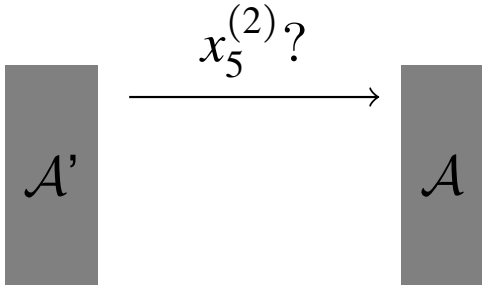
Input: z_1 z_2 z_3 $\cdots$ z_n

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

$\mathcal{A}'$: algorithm for $f \circ g$

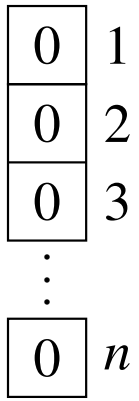
	$\Pr[x_5^{(2)} = 0]$	$\Pr[x_5^{(2)} = 1]$
μ_0	0.5	0.5
μ_1	0.6	0.4

to be drawn from μ_{z_2}



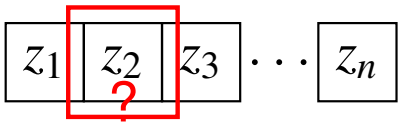
μ_0 : 0.5 0.5

μ_1 : 0.6 0.4



Query register

Input:

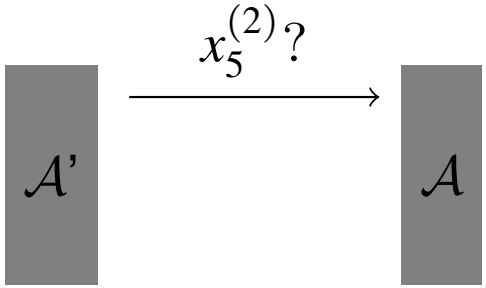


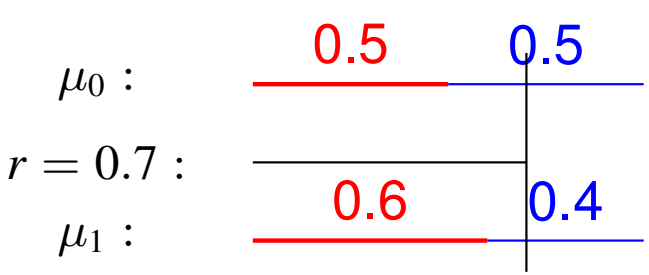
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

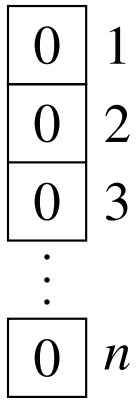
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_5^{(2)} = 0]$	$\Pr[x_5^{(2)} = 1]$
μ_0	0.5	0.5
μ_1	0.6	0.4

to be drawn from μ_{z_2}

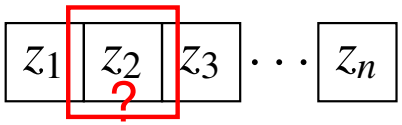






Query register

Input:

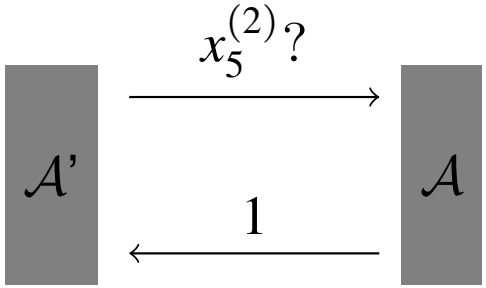


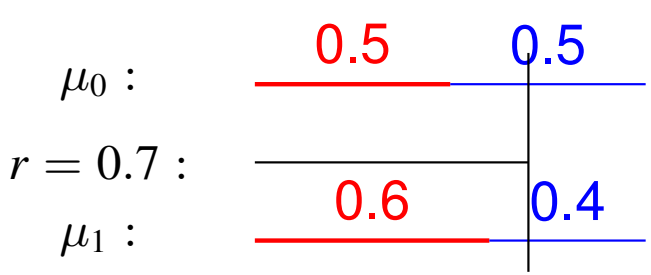
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

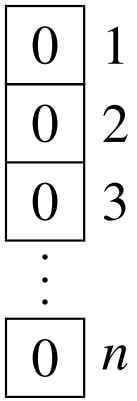
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_5^{(2)} = 0]$	$\Pr[x_5^{(2)} = 1]$
μ_0	0.5	0.5
μ_1	0.6	0.4

to be drawn from μ_{z_2}





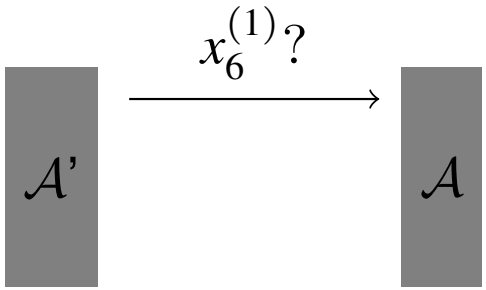


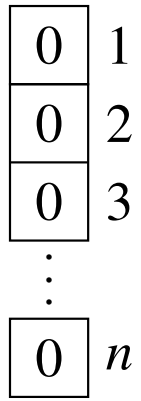
Query register

Input: $z_1 \mid z_2 \mid z_3 \mid \cdots \mid z_n$

Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

$\mathcal{A}'$: algorithm for $f \circ g$





Query register

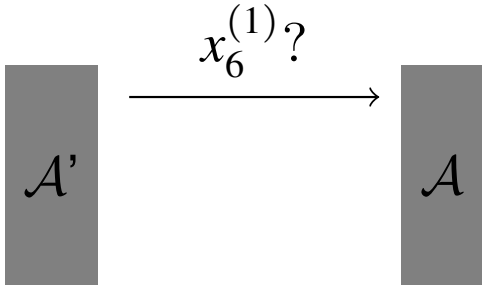
Input: z_1 z_2 z_3 $\cdots$ z_n

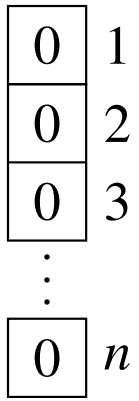
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_6^{(1)} = 0]$	$\Pr[x_6^{(1)} = 1]$
μ_0	0.3	0.7
μ_1	0.8	0.2

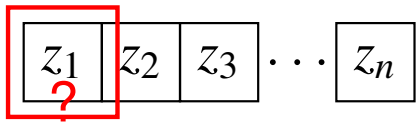
to be drawn from μ_{z_1}





Query register

Input:

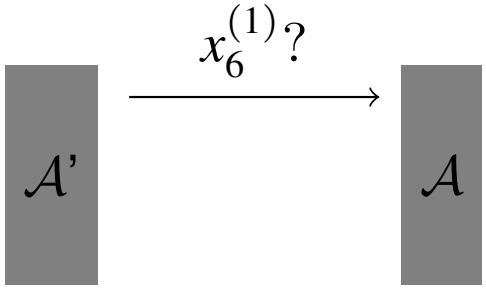


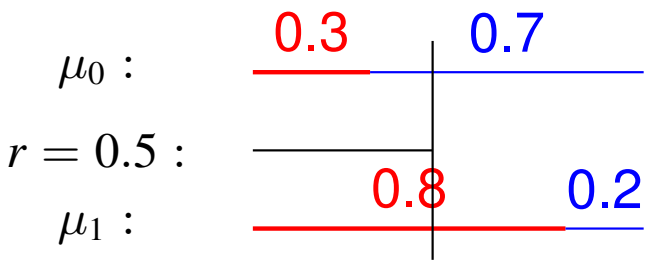
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

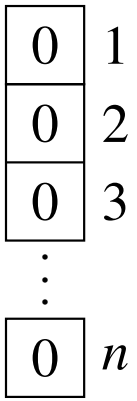
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_6^{(1)} = 0]$	$\Pr[x_6^{(1)} = 1]$
μ_0	0.3	0.7
μ_1	0.8	0.2

to be drawn from μ_{z_1}

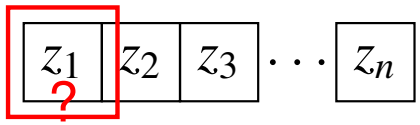






Query register

Input:

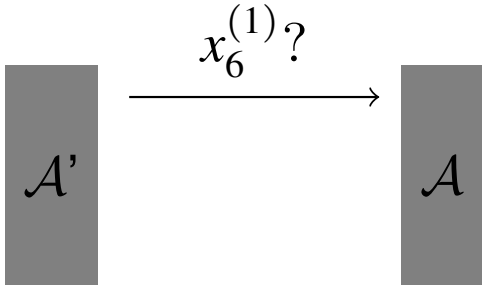


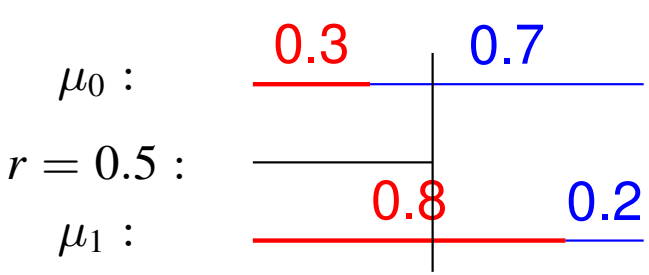
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

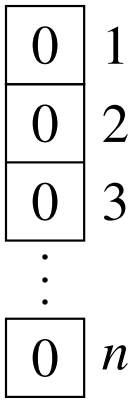
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_6^{(1)} = 0]$	$\Pr[x_6^{(1)} = 1]$
μ_0	0.3	0.7
μ_1	0.8	0.2

to be drawn from μ_{z_1}

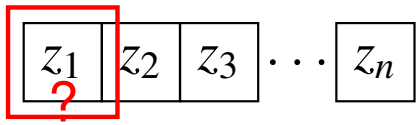






Query register

Input:



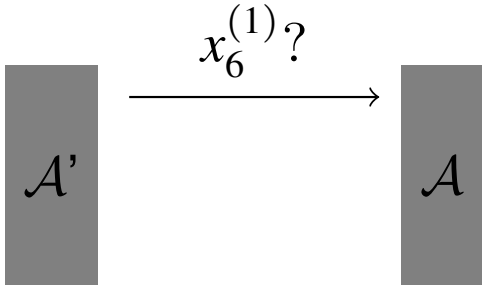
Query z_1!

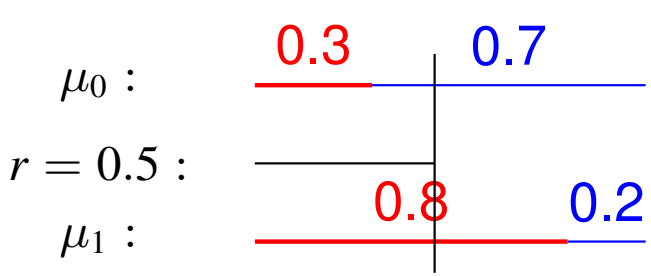
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

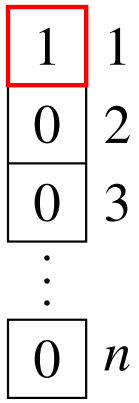
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_6^{(1)} = 0]$	$\Pr[x_6^{(1)} = 1]$
μ_0	0.3	0.7
μ_1	0.8	0.2

to be drawn from μ_{z_1}







Query register

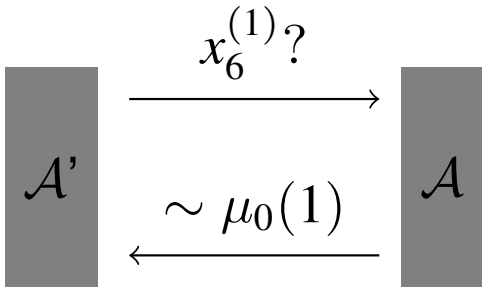
Input: 

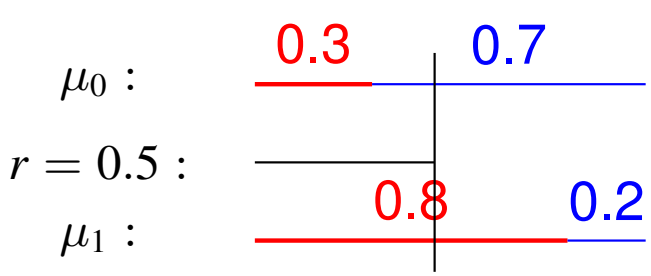
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

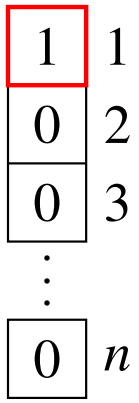
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_6^{(1)} = 0]$	$\Pr[x_6^{(1)} = 1]$
μ_0	0.3	0.7
μ_1	0.8	0.2

to be drawn from μ_{z_1}







Query register

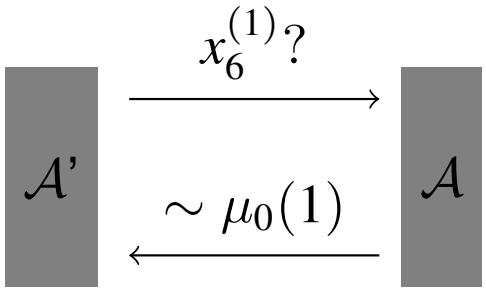
Input: $\boxed{0}\boxed{z_2}\boxed{z_3}\cdots\boxed{z_n}$

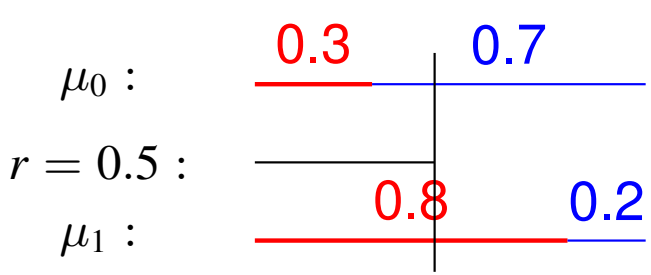
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

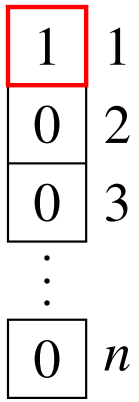
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_6^{(1)} = 0]$	$\Pr[x_6^{(1)} = 1]$
μ_0	0.3	0.7
μ_1	0.8	0.2

to be drawn from μ_{z_1}







Query register

Input: $\boxed{0}\boxed{z_2}\boxed{z_3}\cdots\boxed{z_n}$

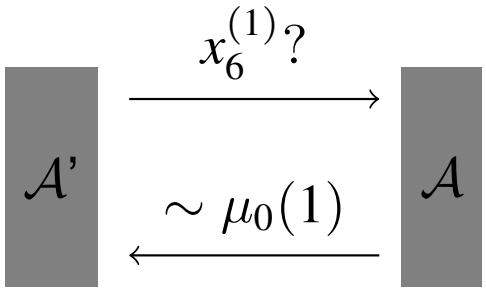
- One 'real' query in how many 'false' queries?

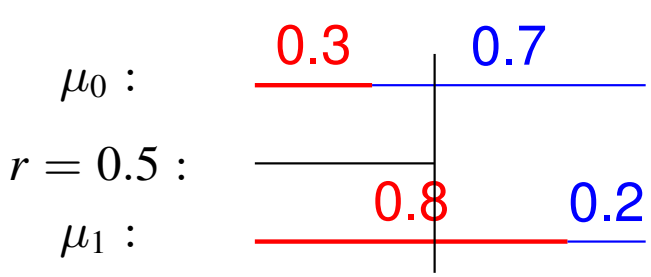
Proof Outline (Contd.): Designing algorithm $\mathcal{A}$ for f

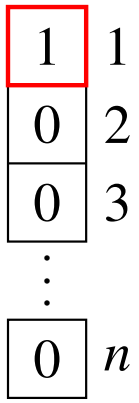
$\mathcal{A}'$: algorithm for $f \circ g$

	$\Pr[x_6^{(1)} = 0]$	$\Pr[x_6^{(1)} = 1]$
μ_0	0.3	0.7
μ_1	0.8	0.2

to be drawn from μ_{z_1}







Query register

Input: 

- One 'real' query in how many 'false' queries?
- Conflict complexity of g ($\chi(g)$). Choose μ_0, μ_1 optimally.

proof outline: finishing.

- $R(f) = O(R(f \circ g)/\chi(g))$.

proof outline: finishing.

- $R(f) = O(R(f \circ g)/\chi(g))$.
- $\chi(R(g)) = \Omega(\sqrt{R(g)})$. Proof uses information theory.

End of proof outline.

Thank you.