A lifting-esque theorem for constant depth formulas

with consequences for MCSP and lower bounds

UNIVERSITY OF OXFORD

WARWICK
THE UNIVERSITY OF WARWICK

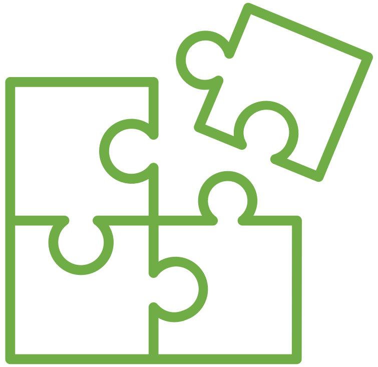
Rahul Ilango

MIT

Talk Goals

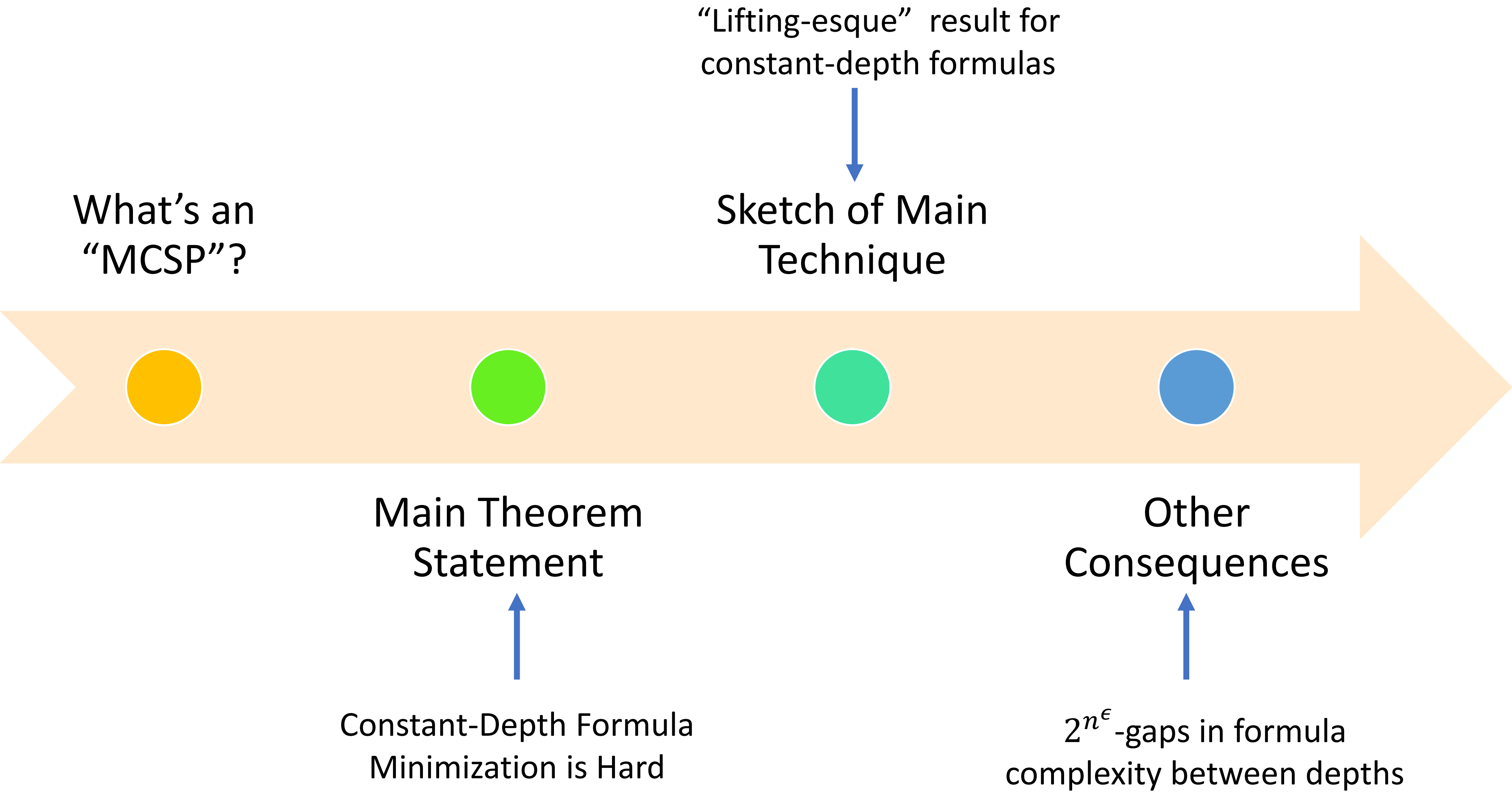
Learn something about:

Some problem called
MCSP



Proving this “lifting-esque” theorem

Road Map



"Lifting-esque" result for constant-depth formulas

What's an "MCSP"?

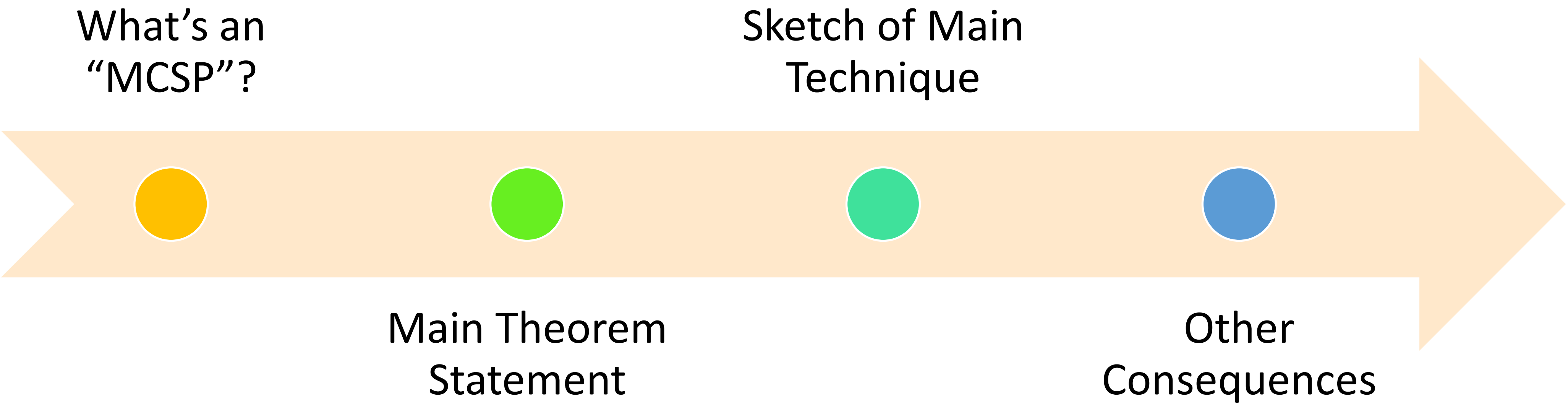
Sketch of Main Technique

Main Theorem Statement

Other Consequences

Constant-Depth Formula Minimization is Hard

$2^{n^{\epsilon}}$-gaps in formula complexity between depths



What's an "MCSP"?

Main Theorem Statement

Sketch of Main Technique

Other Consequences

What is MCSP?

The Minimum Circuit Size Problem (MCSP)

<u>Input</u>

<u>Output</u>

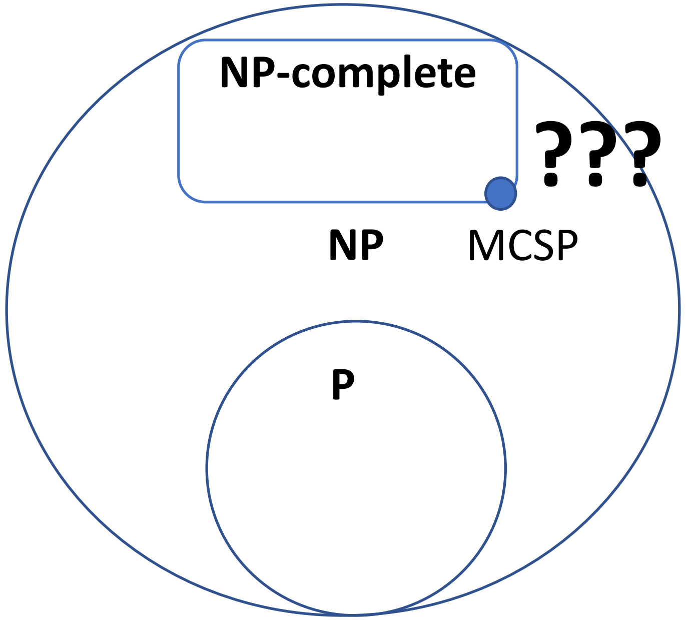
$\Bigg($ Truth table T of a Boolean function f , "size threshold" $s \in \mathbb{N}$ in unary $\Bigg) \longrightarrow$ ∃ circuit with $\leq s$ gates computing f?

$T =$

x	0^n	…	1^n
$f(x)$	$f(0^n)$	…	$f(1^n)$

$N = 2^n$

Complexity:



Why care about MCSP?

The search for fundamental problems

What problem have we learned the most from? SAT !!

Study of SAT →

NP-completeness,

PCPs,

SAT solvers,

Fine-grained complexity

SAT is fundamental because

Natural questions ⇒ important (often unexpected) advances

Can we can find more fundamental problems?

A potential fundamental problem?

"MCSP is **more fundamental** than SAT!"

-- Rahul (Santhanam)

1. Connections to:

Cryptography Learning Structural Complexity Average Case Complexity Circuit Complexity

2. Its complexity is a mystery

Is MCSP NP-complete?

Is MCSP hard to approximate?

Can you beat the naïve brute-force algorithm?

X is true about MCSP	$\Rightarrow$	Solution to a long-standing open problem
MCSP is NP-complete	$\Rightarrow$ [Murray-Williams '15]	$\mathrm{EXP} \neq \mathrm{ZPP}$
An approximation to MCSP is NP-complete	$\Rightarrow$ [Hirahara '18]	Computing NP "on average" is as hard as computing NP in the "worst-case"
A version of MCSP does not have $n \operatorname{poly}(\log n)$ circuits	$\Rightarrow$ [McKay-Murray-Williams '19]	NP does not have polynomial-size circuits

Cryptography

Learning

Average Case Complexity

Structural Complexity

Circuit Complexity

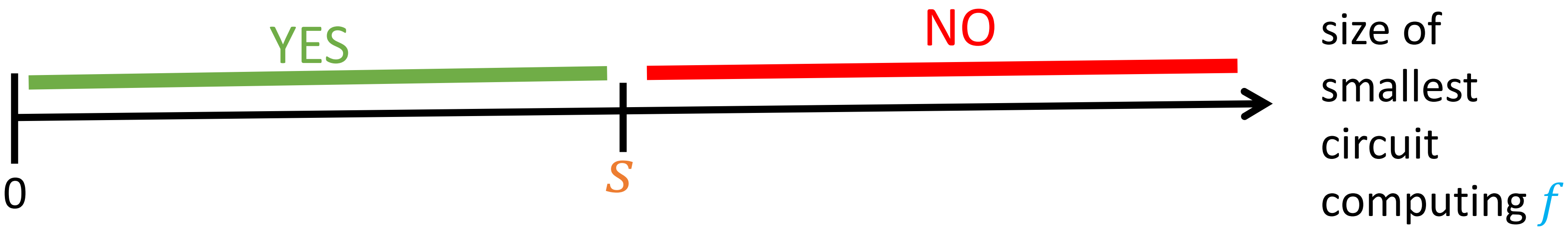
What are these connections?

YouTube Rahul Ilango TCS+ Talk

Is MCSP NP-hard?

Input: function f and integer s

MCSP

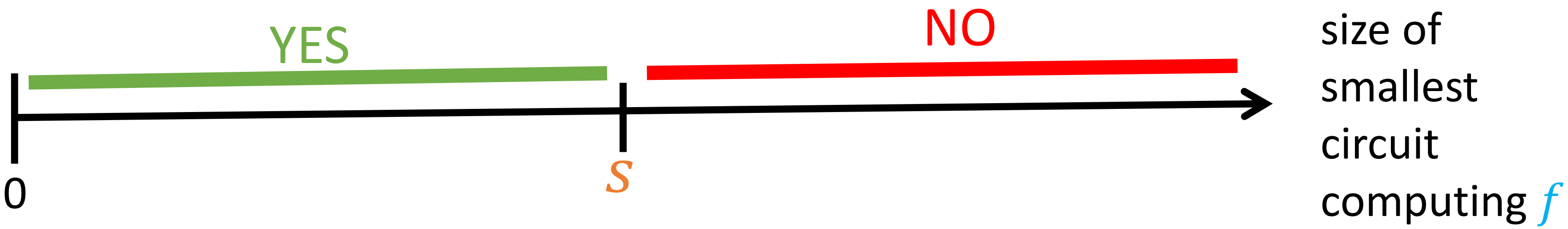


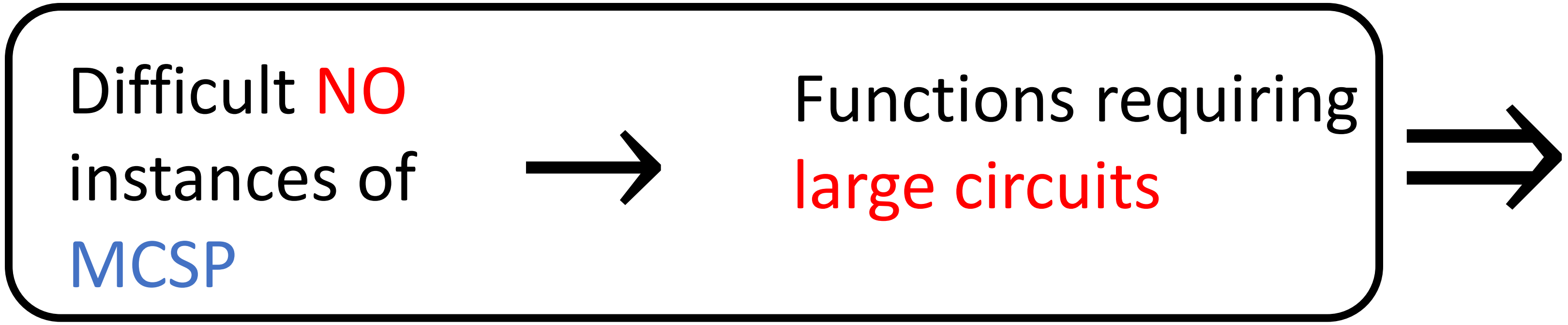
> “If it is NP-complete, it would have to require techniques that are not like any polynomial time reduction that we have ever seen”

Is MCSP NP-hard?

Input: function f and integer s

MCSP





Difficult NO instances of MCSP → Functions requiring large circuits ⇒

Deterministic poly-time reduction requires breakthrough

[Kabanets-Cai ‘00, Murray-Williams ’16, Saks-Santhanam ‘20]

Conjecture: ETH ⇒ MCSP ∉ P

Randomized Reductions?

Is $\mathcal{C}$-MCSP NP-hard?

Don’t know functions requiring large circuits → Hard to prove MCSP is NP-hard

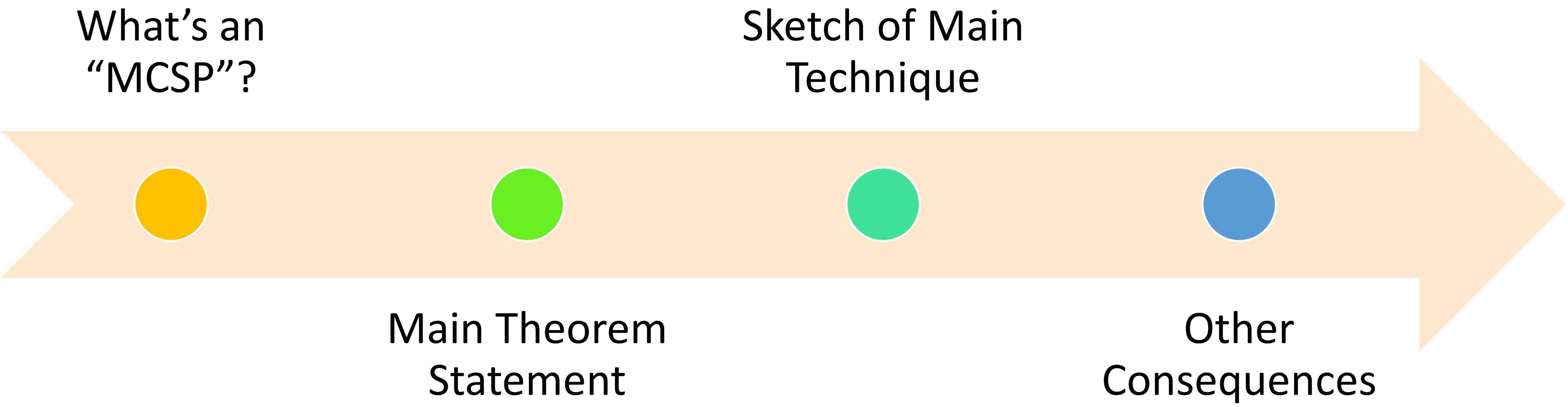
Know functions requiring large $\mathcal{C}$-circuits → Can we prove $\mathcal{C}$-MCSP is NP-hard?

Circuit class $\mathcal{C} \in \{\underline{\mathrm{DNF},} \quad \underline{\mathrm{DNF} \circ XOR}, \ldots, \mathrm{AC}^0_\mathrm{d}, \mathrm{AC}^0_\mathrm{d}[2]\}$

NP-hard by
[Masek ’79,…, Khot-Saket ‘08]

NP-hard by
[Hirahara-Oliveira-Santhanam ‘18]

???



What’s an “MCSP”?

Main Theorem Statement

Sketch of Main Technique

Other Consequences

Main Result

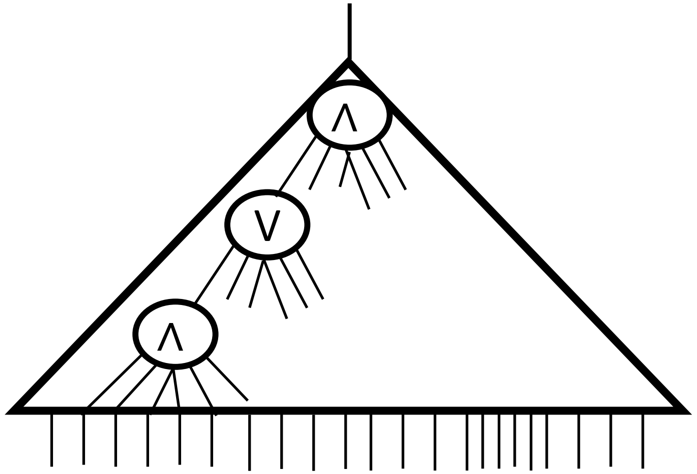
Main Result: Preliminaries

Def

Let $L_d(f) :=$ min. # leaves in depth-d formula computing f

Constant Depth Formula Model

- Rooted tree of constant depth
- Internal nodes labeled by AND, OR gates of unbounded fan-in
- Leaf nodes labelled by $\{0,1, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$
- Size of formula = # of leaves (**ignoring constant leaves**)
- Gates alternate between AND and OR



Note: Computing $L_d(f)$ reduces to (depth-d formula)-MCSP

Main Result

Def

Let $L_d(f) \coloneqq$ min. # leaves in depth-d formula computing f

Theorem

For all $d \geq 2$, computing $L_d(\cdot)$ is NP-hard under quasi-poly time randomized Turing reductions.

Proof Outline: An Inductive Approach

Theorem: Computing $L_d(\cdot)$ is NP-hard for all $d \geq 2$.

Step 1: Restrict to top OR gate

Def: $L_d^{OR}(f) :=$ min. leaves in OR-top depth-d formula for f

Thm: If computing $L_d^{OR}(\cdot)$ is NP-hard, then so is computing $L_d(\cdot)$

Step 2: $d = 2$ Base Case

Thm: "Approx." computing $L_2^{OR}(\cdot)$ is NP-hard

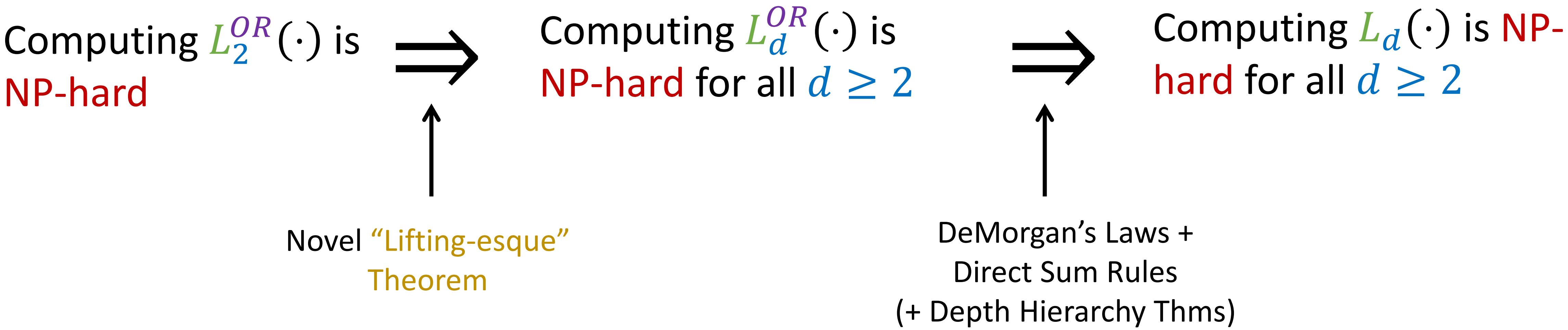
Known from [Masek '79,..., Allender et al. '06, Feldman '06, Khot-Saket '08]

Step 3: $d \geq 3$ Inductive Argument

Thm: "approx." computing $L_d^{OR}(\cdot)$ reduces to "approx." computing $L_{d+1}^{OR}(\cdot)$

Proof Outline: Techniques

Theorem: Computing $L_d(\cdot)$ is NP-hard for all $d \geq 2$.



Computing $L_2^{OR}(\cdot)$ is NP-hard $\Rightarrow$ Computing $L_d^{OR}(\cdot)$ is NP-hard for all $d \geq 2$ $\Rightarrow$ Computing $L_d(\cdot)$ is NP-hard for all $d \geq 2$

Novel “Lifting-esque” Theorem

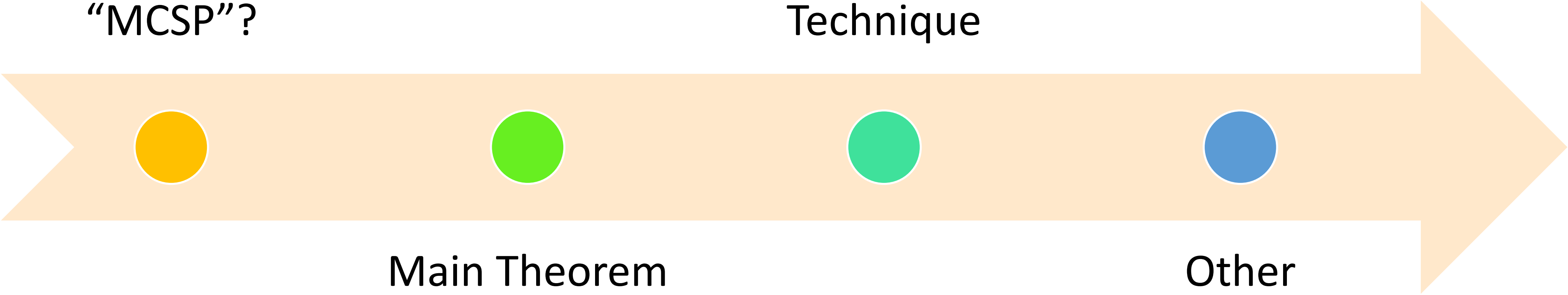
DeMorgan’s Laws +
Direct Sum Rules
(+ Depth Hierarchy Thms)

Reducing depth-d to d+1: Pseudocode

Given f and oracle to $L_{d+1}^{OR}(\cdot)$, estimate $L_d^{AND}(f) \; = \; L_d^{OR}(\neg f)$

```
while True:
    Sample $(g, error\_bound) \leftarrow \mathcal{D}$
    Let $H(x,y) = f(x) \wedge g(y)$
    Set $f\_estimate$ = $L_{d+1}^{OR}(H) - L_{d+1}^{OR}(g)$
    If $f\_estimate$ >> $error\_bound$ :
        Output that $L_d^{AND}(f) \; \approx \; f\_estimate$.
```

⟵ I'll try to explain why this quantity roughly estimates $L_d^{AND}(f)$



What’s an “MCSP”?

Main Theorem Statement

Sketch of Main Technique

Other Consequences

Sketch of “Lifting-esque Result”

Intuition

$L_d^{OR}(\neg f)$

$\|$

Want: Given f and oracle access to $L_{d+1}^{OR}(\cdot)$, compute $L_d^{AND}(f)$

Idea: Find function H whose optimal depth-(d+1) OR-top formula *contains* an optimal depth-d AND-top formula for f

How? Switching Lemma??

Direct Sum Idea! $H(x, y) = f(x) \wedge g(y)$ for some function g

Intuition for H

$$H(x,y) = f(x) \wedge g(y)$$

Naïve family of OR-top depth-(d+1) formulas for H:

OR-top depth-(d+1) formulas for f and g $\mapsto$ OR-top depth-(d+1) formulas for H

$$f(x) = \phi(x) = \bigvee_{i \in [t_f]} \phi_i(x)$$

$$g(y) = \Psi(y) = \bigvee_{j \in [t_g]} \Psi_j(y),$$

$$H(x,y) = \bigvee_{(i,j) \in [t_f] \times [t_g]} (\phi_i(x) \wedge \Psi_j(y))$$

Size: $t_g \cdot |\phi| + t_f \cdot |\Psi|$

If

- g is waaaay more complex than f and
- has optimal formulas with $t_g = 1$,

then the size is plausibly minimized by using the smallest ϕ with $t_f = 1$

In which case:

$$L^{OR}_{d+1}(H) = L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$$

Main Technical Result

$H(x,y) = f(x) \wedge g(y)$

What does this mean?

Theorem (Informal): If g is “expensive” compared to f, then

$$L_{d+1}^{OR}(H) \geq L_{d}^{AND}(f) + L_{d+1}^{OR}(g).$$

Is this tight?

Technical Result Preliminaries

- Non-Deterministic Formulas
- One-sided Approximations
- Direct Sum Theorems

Preliminaries: Non-Deterministic Formulas

A non-deterministic (ND) formula Ψ specified by

- an integer m specifying the number of "non-deterministic inputs"
- (unrestricted) formula $\phi(x, y)$ on $(m + n)$-inputs

Non-deterministic input

Regular input

Computes n-bit function given by $\Psi(y) := \bigvee_x \phi(x, y)$

Size of non-det. formula $|\Psi| := |\phi|$

Def (Bounded non-det. formula complexity)
$L_{ND}(f) :=$ min size of ND formula for f with $m = n$ non-det. input bits

Preliminaries: One-Sided Approximation

Let $g, \tilde{g}: \{0,1\}^n \to \{0,1\}$.

Def

$\tilde{g}$ is an α-one sided approximation of g if

- $\tilde{g}$ rejects all NO instances of g
- $\tilde{g}$ accepts at least an α-fraction of the YES instances of g
  - i.e. $|\tilde{g}^{-1}(1)| \geq \alpha \cdot |g^{-1}(1)|$

Def

$L_{ND,\alpha}(g) := \min L_{ND}(\tilde{g})$ over all α-one sided approx $\tilde{g}$ of g

Preliminaries: Direct Sum Theorem

Recall: $H(x, y) = f(x) \wedge g(y)$

Thm (Folklore?):

Let f, g be non-constant functions. Then

$$L_d^{OR}\big(H(x, y)\big) \geq L_d^{OR}(f) + L_d^{OR}(g).$$

Proof

Suppose

- $\phi(x, y) = f(x) \wedge g(y)$
- $g(y^\star) = 1.$

ϕ has $\geq L_d^{OR}(f)$ many x-leaves.

Then restriction $\phi(x, y^\star)$ computes f.

Similarly, ϕ has $\geq L_d^{OR}(g)$ many y-leaves.

What is “expensive”?

$H(x, y) = f(x) \wedge g(y)$

Theorem (Informal): If g is “expensive” compared to f, then

$$L_{d+1}^{OR}(H) \geq L_d^{AND}(f) + L_{d+1}^{OR}(g).$$

g is expensive compared to f if

g takes more inputs than f,

and both

- $L_{ND}(g) + L_{ND,\gamma}(g)$ ⟵ “ND complexity of g and a weak approx. to g”
- $2 \cdot L_{ND,.73}(g)$ ⟵ “ND complexity of computing strong approx. to g twice”

are greater than $L_d^{AND}(f) + L_{d+1}^{OR}(g)$ ⟵ Our desired lower bound

γ = “some small number” = 10^{-4}

Formal Theorem

$H(x, y) = f(x) \wedge g(y)$

Theorem: $L_{d+1}^{OR}(H) \geq L_d^{AND}(f) + L_{d+1}^{OR}(g)$

when $\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,\,.73}(g)\} > L_d^{AND}(f) + L_{d+1}^{OR}(g)$

and f and g are non-constant and g takes more inputs than f.

Is this tight?

$H(x, y) = f(x) \wedge g(y)$

Theorem: $L_{d+1}^{OR}(H) \geq L_{d}^{AND}(f) + L_{d+1}^{OR}(g)$

when $\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,\,.73}(g)\} > L_{d}^{AND}(f) + L_{d+1}^{OR}(g)$

and f and g are non-constant and g takes more inputs than f.

Trivial Lower Bound: $L_{d+1}^{OR}(H) \geq L_{d+1}^{OR}(f) + L_{d+1}^{OR}(g)$

Trivial Upper Bound: $L_{d+1}^{OR}(H) \leq L_{d}^{AND}(H) = L_{d}^{AND}(f) + L_{d}^{AND}(g)$

Best Bounds: $L_{d}^{AND}(f) + L_{d+1}^{OR}(g) \leq L_{d+1}^{OR}(H) \leq L_{d}^{AND}(f) + L_{d}^{AND}(g)$

Tight if: $L_{d+1}^{OR}(g) = L_{d}^{AND}(g)$

Is this tight?

$H(x, y) = f(x) \wedge g(y)$

Theorem: $L^{OR}_{d+1}(H) \geq L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$

when $\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,\,.73}(g)\} > L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$

and f and g are non-constant and g takes more inputs than f.

Best Bounds : $L^{AND}_{d}(f) + L^{OR}_{d+1}(g) \leq L^{OR}_{d+1}(H) \leq L^{AND}_{d}(f) + L^{AND}_{d}(g)$

$$L^{AND}_{d}(f) \leq \underbrace{L^{OR}_{d+1}(H) - L^{OR}_{d+1}(g)}_{(1)} \leq L^{AND}_{d}(f) + \underbrace{[L^{AND}_{d}(g) - L^{OR}_{d+1}(g)]}_{(2)}$$

So $L^{AND}_{d}(f) \approx$ (1) up to additive error (2)

Can build on this to give the desired reduction between depth-d and depth-(d+1)

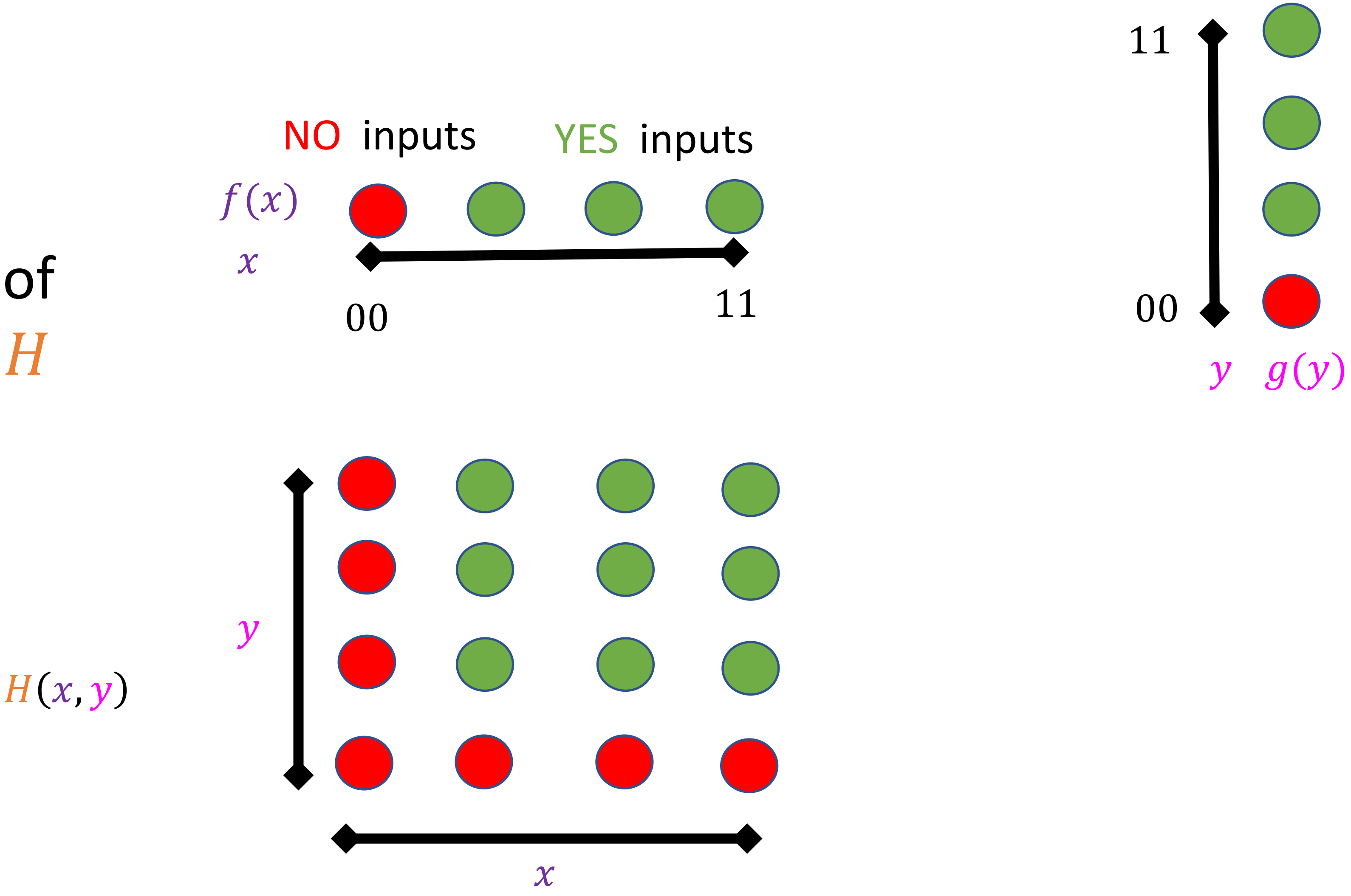
Proof!

Theorem: $L_{d+1}^{OR}(H) \geq L_d^{AND}(f) + L_{d+1}^{OR}(g)$

when $\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,.73}(g)\} > L_d^{AND}(f) + L_{d+1}^{OR}(g)$

and f and g are non-constant and g takes more inputs than f.

Visualization of the f, g, and H functions



Proof!

Theorem: $L_{d+1}^{OR}(H) \geq L_d^{AND}(f) + L_{d+1}^{OR}(g)$

when $\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,.73}(g)\} > L_d^{AND}(f) + L_{d+1}^{OR}(g)$

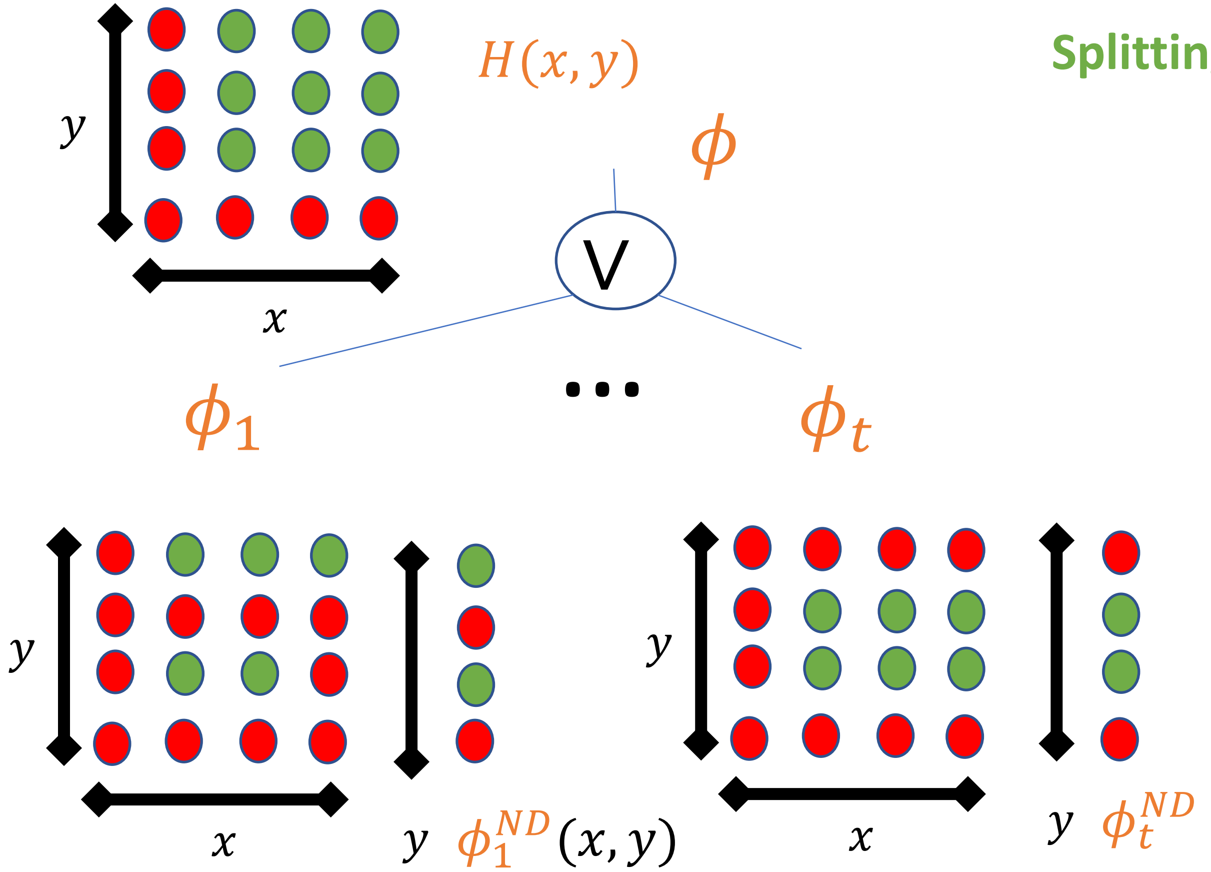
and f and g are non-constant and g takes more inputs than f.

Suppose ϕ computing $H(x,y) = f(x) \wedge g(y)$ contradicted this

Splitting Claim:
Can split ϕ^{ND} into two disjoint subformulas Ψ_L^{ND} and Ψ_R^{ND} that are both (.73)-one sided non-det. approxs of g.

Splitting Claim ⇒ done!:

$$\begin{aligned} |\phi| &= |\phi^{ND}| \\ &\geq |\Psi_L^{ND}| + |\Psi_R^{ND}| \\ &\geq 2 \cdot L_{ND,.73}(g) \\ &> L_d^{AND}(f) + L_{d+1}^{OR}(g) \end{aligned}$$



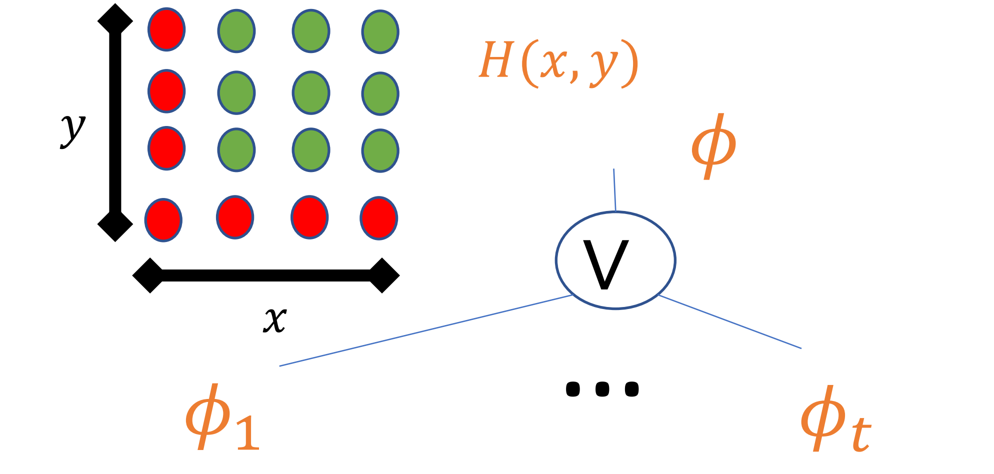
Proof!

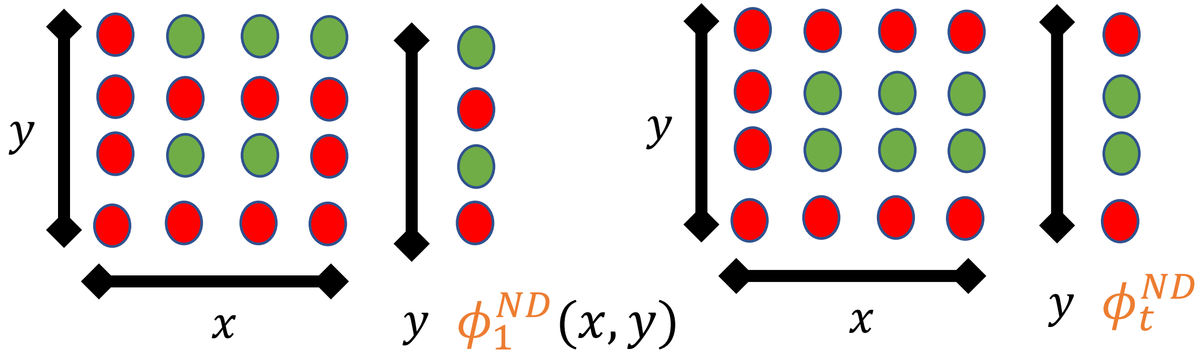
Theorem: $L^{OR}_{d+1}(H) \geq L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$

when $\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,.73}(g)\} > L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$

and f and g are non-constant and g takes more inputs than f.

Suppose ϕ computing $H(x,y) = f(x) \wedge g(y)$ contradicted this





Splitting Claim:
Can split ϕ^{ND} into two disjoint subformulas Ψ_L^{ND} and Ψ_R^{ND} that are both (.73)-one sided non-det. approxs of g.

Redundancy Claim: Every YES instances $y^\star$ of g is non-det. accepted by at least two of $\phi_1^{ND}, \ldots, \phi_t^{ND}$.

Pf: Suppose $y^\star$ is only non-det. accepted by ϕ_1^{ND}

Then $\phi_i(x, y^\star) = 0$ for all x and $i \geq 2$.

But then $\phi_1(x, y^\star)$ computes $f(x)$:

$$f(x) = H(x, y^\star) = \phi(x, y^\star) = \vee_i \, \phi_i(x, y^\star) = \phi_1(x, y^\star)$$

Then **depth-d sub formula** ϕ_1 has $\geq L^{AND}_{d}(f)$ many x-leaves!

But ϕ has $\geq L^{OR}_{d+1}(g)$ many y-leaves, by setting x to a YES instance of f!

So $|\phi| \geq L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$

Proof!

Theorem: $L_{d+1}^{OR}(H) \geq L_d^{AND}(f) + L_{d+1}^{OR}(g)$

when $\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,.73}(g)\} > L_d^{AND}(f) + L_{d+1}^{OR}(g)$

and f and g are non-constant and g takes more inputs than f.

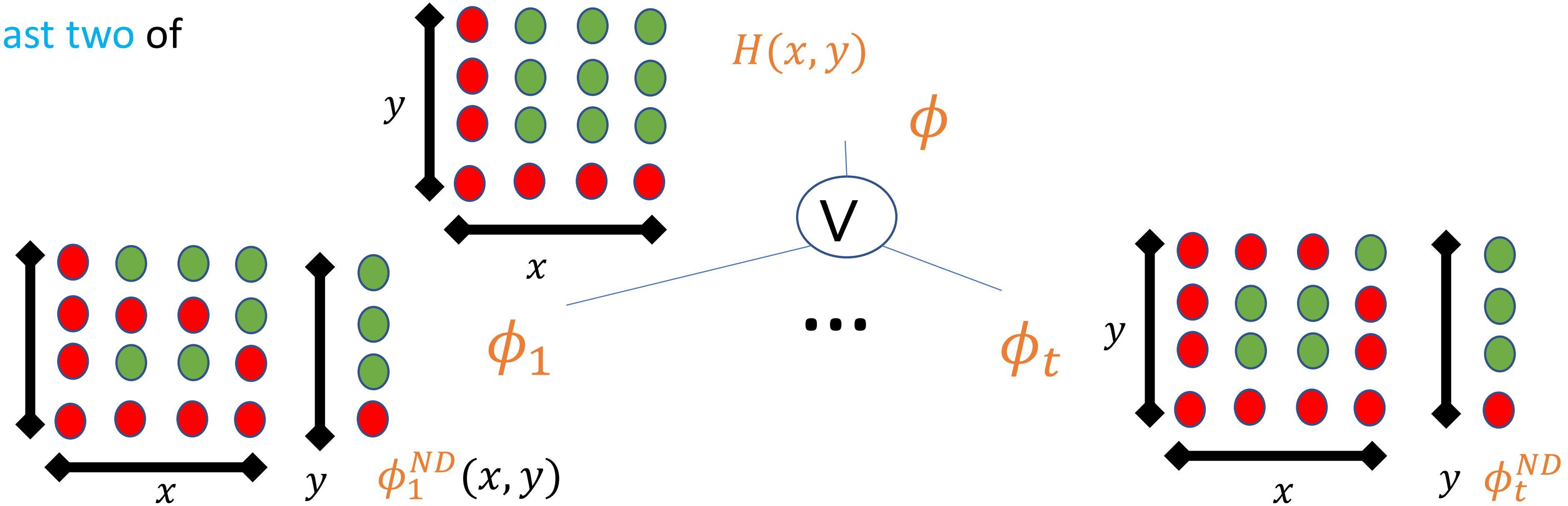
Splitting Claim:
Can split ϕ^{ND} into two disjoint subformulas Ψ_L^{ND} and Ψ_R^{ND} that are both (.73)-one sided non-det. approxs of g.

Redundancy Claim: Every YES instances $y^\star$ of g is non-det. accepted by at least two of $\phi_1^{ND}, \dots, \phi_t^{ND}$.

Pf of Splitting Claim:
Pick L and R to be a **uniformly random** partition of [t].
Let $\Psi_L^{ND}(x,y) = \vee_{i \in L}\, \phi_i^{ND}(x,y)$. Let $\Psi_R^{ND} = \vee_{i \in R}\, \phi_i^{ND}(x,y)$.
In expectation Ψ_L^{ND} and Ψ_R^{ND} are .75 one-sided non-det. approx of g.

Why? Because **Linearity of Expectation**:

- Redundancy ⇒ any YES instance $y^\star$ of g has $\geq$ 2 chances to get a $i \in L$ s.t. ϕ_i non-det. accepts $y^\star$



Proof!

Theorem: $L^{OR}_{d+1}(H) \geq L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$

when $\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,.73}(g)\} > L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$

and f and g are non-constant and g takes more inputs than f.

Splitting Claim:
Can split ϕ^{ND} into two disjoint subformulas Ψ^{ND}_L and Ψ^{ND}_R that are both (.73)-one sided non-det. approxs of g.

Redundancy Claim: Every YES instances $y^\star$ of g is non-det. accepted by at least two of $\phi^{ND}_1, \dots, \phi^{ND}_t$.

Pf of Splitting Claim:
Pick L and R to be a **uniformly random** partition of [t].
Let $\Psi^{ND}_L(x,y) = \bigvee_{i \in L} \phi^{ND}_i(x,y)$. Let $\Psi^{ND}_R = \bigvee_{i \in R} \phi^{ND}_i(x,y)$.
In expectation Ψ^{ND}_L and Ψ^{ND}_R are .75 one-sided non-det. approx of g.

Why? Because **Linearity of Expectation**:

- Redundancy ⇒ any YES instance $y^\star$ of g has $\geq$ 2 chances to get a $i \in L$ s.t. ϕ_i non-det. accepts $y^\star$

But **expectation not enough**... Need to hold simultaneously

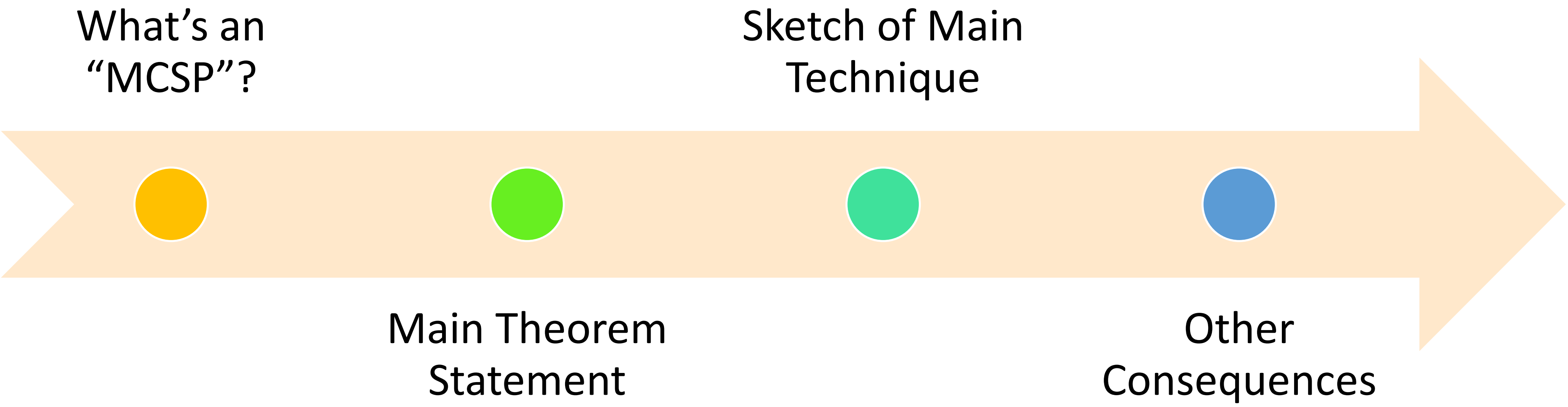
So prove **concentration**! Chebyshev works if one can show:

Each ϕ^{ND}_i accepts $\leq \gamma$-fraction of g's YES instances

If not, then $|\phi^{ND}_i| \geq L_{ND,\gamma}(g)$

OTOH: Redundancy ⇒ $\bigvee_{j \neq i} \phi^{ND}_j$ computes g non-det. ⇒ $|\bigvee_{j \neq i} \phi^{ND}_j| \geq L_{ND}(g)$

But then $|\phi| = |\phi^{ND}_i| + |\bigvee_{j \neq i} \phi^{ND}_j| \geq L_{ND}(g) + L_{ND,\gamma}(g) \geq L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$



Other Consequences

Gaps in Formula Complexity Between Depths

Theorem

There exists an $\epsilon > 0$ s.t. for all $d \geq 2$ there exists a function f such that $L_d(f) - L_{d+1}(f) \geq 2^{\Omega_d(n^\epsilon)}$

$\boldsymbol{d = 2, 3}$ **cases:** Use existing depth hierarchy theorems [Hastad '89] that shows $2^{n^{\Omega\left(\frac{1}{d}\right)}}$ seperation

$\boldsymbol{d \geq 4}$ **case:** Use "Lifting-esque Theorem" to "lift" a $L_d(f) - L_{d+1}(f)$ separation into a $L_{d+1}(H) - L_{d+2}(H)$ (cost is a constant in the exponent)

$$H(x, y) = f(x) \wedge g(y)$$

Thanks!

Questions?

Finding good g

Suppose you have a f on n-inputs of size s

One can sample a g such that

Affects # of inputs to $H(x, y) = f(x) \wedge g(y)$

Hypothesis of Lifting-esque Lower Bound

$L^{OR}_{d+1}(H) \approx L^{OR}_{d+1}(g) + L^{AND}_{d}(f)$

Use	Inputs to **g**	$\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,.73}(g)\}$ $> L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$	Inequality Slack $L_d(f) - L_{d+1}(f)$	How to Sample

Finding good g

Suppose you have a f on n-inputs of size s

One can sample a g such that

Affects # of inputs to $H(x, y) = f(x) \wedge g(y)$

↓

Hypothesis of Lifting-esque Lower Bound

↓

$L^{OR}_{d+1}(H) \approx L^{OR}_{d+1}(g) + L^{AND}_{d}(f)$

↓

Use	**Inputs to g**	$\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,.73}(g)\} > L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$	**Inequality Slack** $L_d(f) - L_{d+1}(f)$	**How to Sample**
Reduction	$poly(n)$	☑	$o(s)$ for $d \geq 2$	Depth-2 Subformula of Lupanov's formula for random function

Finding good g

Suppose you have a f on n-inputs of size s

One can sample a g such that

Affects # of inputs to $H(x, y) = f(x) \wedge g(y)$

Hypothesis of Lifting-esque Lower Bound

$L^{OR}_{d+1}(H) \approx L^{OR}_{d+1}(g) + L^{AND}_{d}(f)$

Use	Inputs to ***g***	$\min\{L_{ND}(g) + L_{ND,\gamma}(g), 2 \cdot L_{ND,.73}(g)\} > L^{AND}_{d}(f) + L^{OR}_{d+1}(g)$	**Inequality Slack** $L_d(f) - L_{d+1}(f)$	**How to Sample**
Reduction	$poly(n)$	☑	$o(s)$ for $d \geq 2$	Depth-2 Subformula of Lupanov's formula for random function
Gap Theorem	$O(n)$	☑	$o(s)$ if $d \geq 3$	Biased random function

Depth-2 Subformulas of Lupanov

- $m = n^{100}$
- For each $x \in \{0,1\}^n$, select a random subset $S_x \subseteq [m]$
- $g: \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$
- $g(x, y) = \bigvee_{\tilde{x} \in \{0,1\}^n} 1_{x=\tilde{x}}(x) \wedge 1_{weight(y)=1}(y) \wedge 1_{y \subseteq S_{\tilde{x}}}(y)$