# Local Enumeration: The Not-All-Equal Case



Mohit Gurumukhani
Cornell University

Ramamohan Paturi
University of California, San Diego

Michael Saks
Rutgers University

Navid Talebanfard
University of Sheffield

# Depth $3$ Circuits

# Depth $3$ Circuits

**Definition($\Sigma\Pi\Sigma$ circuits)**

# Depth $3$ Circuits

## Definition($\Sigma\Pi\Sigma$ circuits)

- Layers of *OR*, *AND*, *OR* gates.

# Depth $3$ Circuits

**Definition($\Sigma\Pi\Sigma$ circuits)**

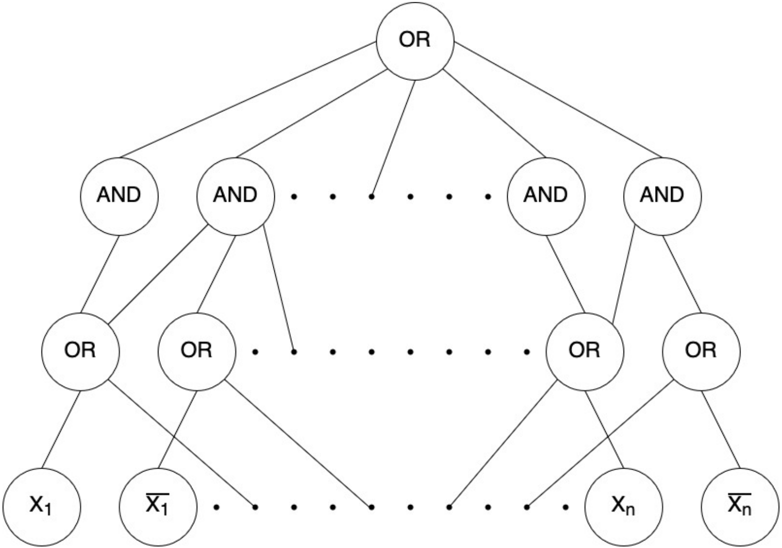
- Layers of *OR*, *AND*, *OR* gates.
- Bottom layer connected to input variables.

# Depth $3$ Circuits

**Definition($\Sigma\Pi\Sigma$ circuits)**

- Layers of *OR*, *AND*, *OR* gates.
- Bottom layer connected to input variables.
- Output gate - single *OR* gate at top.

# Depth $3$ Circuits



# Depth $3$ Circuits

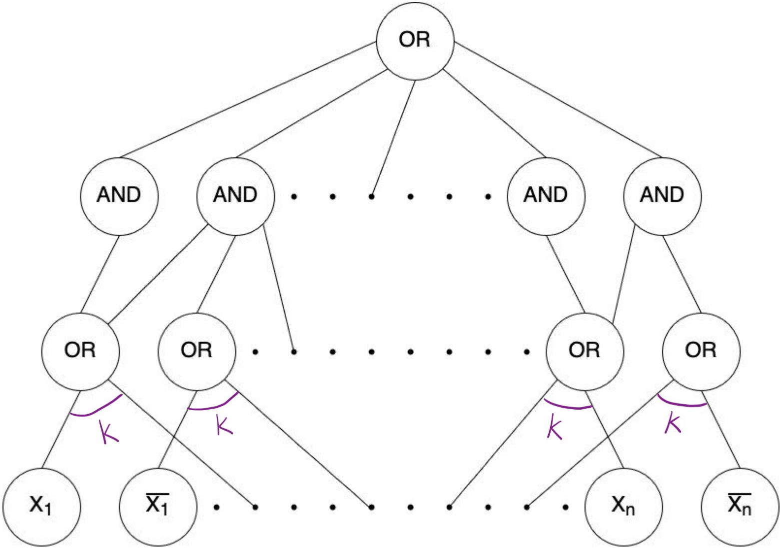
## Definition($\Sigma\Pi\Sigma$ circuits)

- Layers of *OR*, *AND*, *OR* gates.
- Bottom layer connected to input variables.
- Output gate - single *OR* gate at top.

## Definition($\Sigma\Pi\Sigma_k$ circuits)

$\Sigma\Pi\Sigma$ circuits where bottom layer *OR* gates have fan in $k$.

# Depth $3$ Circuits



# Why Depth $3$ Circuits?

Strong depth $3$ circuit lower bounds imply:

# Why Depth $3$ Circuits?

Strong depth $3$ circuit lower bounds imply:

- Super-linear lower bounds for log-depth circuits [ Valiant'77 ].

# Why Depth $3$ Circuits?

Strong depth $3$ circuit lower bounds imply:

- Super-linear lower bounds for log-depth circuits [ Valiant'77 ].
- Improved $(3.9n)$ general circuit lower bounds [ Golovnev-Kulikov-Williams'21 ].

# Depth $3$ Circuit Lower Bounds

# Depth $3$ Circuit Lower Bounds

- $\Sigma\Pi\Sigma$ Lower bound of $\mathbf{2^{\Omega(\sqrt{n})}}$ known for PARITY, BCH code, and MAJORITY function.

# Depth $3$ Circuit Lower Bounds

- $\Sigma\Pi\Sigma$ Lower bound of $2^{\Omega(\sqrt{n})}$ known for PARITY, BCH code, and MAJORITY function.
- $\Sigma\Pi\Sigma_k$ Lower bound of $2^{\Omega(n/k)}$ known for PARITY, BCH code, and MAJORITY function [ Håstad'1986, Paturi-Pudlák-Zane'1999, Paturi-Pudlák-Saks-Zane'2005, Håstad-Jukna-Pudlák'1995, … ].

# Depth $3$ Circuit Lower Bounds

- $\Sigma\Pi\Sigma$ Lower bound of $2^{\Omega(\sqrt{n})}$ known for PARITY, BCH code, and MAJORITY function.
- $\Sigma\Pi\Sigma_k$ Lower bound of $2^{\Omega(n/k)}$ known for PARITY, BCH code, and MAJORITY function [ Håstad'1986, Paturi-Pudlák-Zane'1999, Paturi-Pudlák-Saks-Zane'2005, Håstad-Jukna-Pudlák'1995, ... ].

## Upper bounds for MAJORITY

# Depth $3$ Circuit Lower Bounds

- $\Sigma\Pi\Sigma$ Lower bound of $2^{\Omega(\sqrt{n})}$ known for PARITY, BCH code, and MAJORITY function.
- $\Sigma\Pi\Sigma_k$ Lower bound of $2^{\Omega(n/k)}$ known for PARITY, BCH code, and MAJORITY function [ Håstad'1986, Paturi-Pudlák-Zane'1999, Paturi-Pudlák-Saks-Zane'2005, Håstad-Jukna-Pudlák'1995, ... ].

## Upper bounds for MAJORITY

- Can construct $\Sigma\Pi\Sigma$ circuit for MAJORITY of size $2^{O(\sqrt{n \log n})}$.

# Depth $3$ Circuit Lower Bounds

- $\Sigma\Pi\Sigma$ Lower bound of $2^{\Omega(\sqrt{n})}$ known for PARITY, BCH code, and MAJORITY function.
- $\Sigma\Pi\Sigma_k$ Lower bound of $2^{\Omega(n/k)}$ known for PARITY, BCH code, and MAJORITY function [ Håstad'1986, Paturi-Pudlák-Zane'1999, Paturi-Pudlák-Saks-Zane'2005, Håstad-Jukna-Pudlák'1995, ... ].

## Upper bounds for MAJORITY

- Can construct $\Sigma\Pi\Sigma$ circuit for MAJORITY of size $2^{O(\sqrt{n \log n})}$.
- Can construct $\Sigma\Pi\Sigma_k$ circuit for MAJORITY of size $2^{O(\log(k)/k)\cdot n}$.

# k-SAT

# k-SAT

**Definition ($k$-SAT)**

Given $k$-CNF $F$, decide if it is satisfiable.

# k-SAT

## Definition ($k$-SAT)

Given $k$-CNF $F$, decide if it is satisfiable.

## Current best algorithms

Algorithms using various techniques, all running in time $2^{n-O(n/k)}$. [ Paturi-Pudlák-Zane'1999, Schöning'1999, Paturi-Pudlák-Saks-Zane'2005, Chan-Williams'2019, Brakensiek-Guruswami'2019, ... ]

# k-SAT

## Definition ($k$-SAT)

Given $k$-CNF $F$, decide if it is satisfiable.

## Current best algorithms

Algorithms using various techniques, all running in time $2^{n-O(n/k)}$. [ Paturi-Pudlák-Zane'1999, Schöning'1999, Paturi-Pudlák-Saks-Zane'2005, Chan-Williams'2019, Brakensiek-Guruswami'2019, ... ]

## Super Strong Exponential Time Hypothesis (SSETH)

There does not exist a $2^{n-\omega(n/k)}$ time algorithm to solve $k$-SAT [ Vyas-Williams'21 ].

# **Enum**$(k, r)$

# Enum$(k, r)$

**Definition (Local Enumeration Problem)**

Enum$(k, r)$: Given $k$-CNF $F$ and $r \geq 0$, s.t. every satisfying assignment of $F$ has Hamming weight $\geq r$, **output all satisfying assignments** of Hamming weight $r$.

# **Enum**$(k, r)$

## Definition (Local Enumeration Problem)

Enum$(k, r)$: Given $k$-CNF $F$ and $r \geq 0$, s.t. every satisfying assignment of $F$ has Hamming weight $\geq r$, **output all satisfying assignments** of Hamming weight $r$.

## Lower Bound via Construction

# Enum$(k, r)$

## Definition (Local Enumeration Problem)

Enum$(k, r)$: Given $k$-CNF $F$ and $r \geq 0$, s.t. every satisfying assignment of $F$ has Hamming weight $\geq r$, **output all satisfying assignments** of Hamming weight $r$.

## Lower Bound via Construction

- Can construct $k$-CNF with $\boldsymbol{2^{n-O(\log(k)/k)\cdot n}}$ satisfying assignments of Hamming weight $n/2$ and no satisfying assignments of Hamming weight $< n/2$.

# $\mathsf{Enum}(k, r)$

## Definition (Local Enumeration Problem)

$\mathsf{Enum}(k, r)$: Given $k$-CNF $F$ and $r \geq 0$, s.t. every satisfying assignment of $F$ has Hamming weight $\geq r$, **output all satisfying assignments** of Hamming weight $r$.

## Lower Bound via Construction

- Can construct $k$-CNF with $2^{n - O(\log(k)/k) \cdot n}$ satisfying assignments of Hamming weight $n/2$ and no satisfying assignments of Hamming weight $< n/2$.
- $\mathsf{Enum}(k, n/2) \geq 2^{n - O(\log(k)/k) \cdot n}$.

# Local Enumeration Applications

**Lemma**

*If* $Enum(k, r)$ *can be solved in time* $T(r)$*:*

# Local Enumeration Applications

**Lemma**

*If Enum$(k, r)$ can be solved in time $T(r)$:*

1. *$k$-SAT can be solved in time $T(n/2)$.*

# Local Enumeration Applications

**Lemma**

*If $Enum(k, r)$ can be solved in time $T(r)$:*

1. *$k$-SAT can be solved in time $T(n/2)$.*
2. *MAJORITY requires $\Sigma\Pi\Sigma_k$ circuits of size $2^n/T(n/2)$.*

# Local Enumeration Applications

## Lemma

*If Enum$(k, r)$ can be solved in time $T(r)$:*

1. *$k$-SAT can be solved in time $T(n/2)$.*
2. *MAJORITY requires $\Sigma\Pi\Sigma_k$ circuits of size $2^n/T(n/2)$.*

## Corollary

*If Enum$(k, n/2) \leq \mathbf{2^{n-O(\log(k)/k)\cdot n}}$:*

# Local Enumeration Applications

## Lemma

*If Enum$(k, r)$ can be solved in time $T(r)$:*

1. *$k$-SAT can be solved in time $T(n/2)$.*
2. *MAJORITY requires $\Sigma\Pi\Sigma_k$ circuits of size $2^n/T(n/2)$.*

## Corollary

*If Enum$(k, n/2) \leq \mathbf{2^{n-O(\log(k)/k)\cdot n}}$:*

1. *Can solve $k$-SAT in time $\mathbf{2^{n-O(\log(k)/k)\cdot n}}$, showing SSETH is False.*

# Local Enumeration Applications

## Lemma

*If Enum$(k, r)$ can be solved in time $T(r)$:*

1. *$k$-SAT can be solved in time $T(n/2)$.*
2. *MAJORITY requires $\Sigma\Pi\Sigma_k$ circuits of size $2^n/T(n/2)$.*

## Corollary

*If Enum$(k, n/2) \leq \mathbf{2^{n-O(\log(k)/k)\cdot n}}$:*

1. *Can solve $k$-SAT in time $\mathbf{2^{n-O(\log(k)/k)\cdot n}}$, showing SSETH is False.*
2. *MAJORITY requires $\Sigma\Pi\Sigma_k$ circuits of size $\mathbf{2^{\Omega(\log(k)/k)\cdot n}}$ and $\Sigma\Pi\Sigma$ circuits of size $\mathbf{2^{\Omega(\sqrt{n\log n})}}$.*

# Previous Work

G–Paturi–Pudlák–Saks–Talebanfard'2024

# Previous Work

## G-Paturi-Pudlák-Saks-Talebanfard'2024

- Enum$(3, n/2)$ can be solved in time $\mathbf{1.598^n}$.

# Previous Work

G-Paturi-Pudlák-Saks-Talebanfard'2024

- Enum$(3, n/2)$ can be solved in time $\mathbf{1.598^n}$.
- MAJORITY requires $\Sigma\Pi\Sigma_3$ circuits of size $\geq \mathbf{1.251^n}$.

# Previous Work

## G-Paturi-Pudlák-Saks-Talebanfard'2024

- Enum$(3, n/2)$ can be solved in time $\mathbf{1.598^n}$.
- MAJORITY requires $\Sigma\Pi\Sigma_3$ circuits of size $\geq \mathbf{1.251^n}$.

## This is Not Tight

By construction, Enum$(3, n/2)$ requires time $\geq \mathbf{6^{n/4} \approx 1.565^n}$.

# NAE Case

# NAE Case

**Definition (Not-All-Equal solution)**

For $k$-CNF $F$, $\alpha$ is NAE solution if both $\alpha$ and $\overline{\alpha}$ satisfy $F$.

# NAE Case

## Definition (Not-All-Equal solution)

For $k$-CNF $F$, $\alpha$ is NAE solution if both $\alpha$ and $\overline{\alpha}$ satisfy $F$.

## Definition (NAE Local Enumeration)

NAE-Enum$(k, r)$: Given $k$-CNF $F$ and $r \geq 0$, s.t. every **NAE** satisfying assignment of $F$ has Hamming weight $\geq r$, **output all NAE satisfying assignments** of Hamming weight $r$.

# NAE Local Enumeration Applications

**Lemma**

*If NAE-Enum$(k, r)$ can be solved in time $T(r)$:*

# NAE Local Enumeration Applications

**Lemma**

*If NAE-Enum$(k, r)$ can be solved in time $T(r)$:*

1. *$(k-1)$-SAT can be solved in time $T(n/2)$.*

# NAE Local Enumeration Applications

**Lemma**

*If NAE-Enum$(k, r)$ can be solved in time $T(r)$:*

1. $(k-1)$*-SAT can be solved in time* $T(n/2)$.
2. *Middle slice function requires* $\Sigma\Pi\Sigma_{k-1}$ *circuits of size* $2^n/T(n/2)$.

# NAE Local Enumeration Applications

## Lemma

*If NAE-Enum$(k, r)$ can be solved in time $T(r)$:*

1. $(k-1)$*-SAT can be solved in time* $T(n/2)$.
2. *Middle slice function requires* $\Sigma\Pi\Sigma_{k-1}$ *circuits of size* $2^n/T(n/2)$.

## Corollary

*If NAE-Enum*$(k, n/2) \leq \boldsymbol{2^{n-O(\log(k)/k)\cdot n}}$:

# NAE Local Enumeration Applications

## Lemma

*If NAE-Enum$(k, r)$ can be solved in time $T(r)$:*

1. $(k-1)$*-SAT can be solved in time* $T(n/2)$.
2. *Middle slice function requires* $\Sigma\Pi\Sigma_{k-1}$ *circuits of size* $2^n/T(n/2)$.

## Corollary

*If NAE-Enum$(k, n/2) \leq \mathbf{2^{n-O(\log(k)/k)\cdot n}}$:*

1. *Can solve k-SAT in time* $\mathbf{2^{n-O(\log(k)/k)\cdot n}}$*, showing SSETH is False.*

# NAE Local Enumeration Applications

## Lemma

*If NAE-Enum*$(k, r)$ *can be solved in time* $T(r)$*:*

1. $(k-1)$*-SAT can be solved in time* $T(n/2)$.
2. *Middle slice function requires* $\Sigma\Pi\Sigma_{k-1}$ *circuits of size* $2^n / T(n/2)$.

## Corollary

*If NAE-Enum*$(k, n/2) \leq \mathbf{2^{n - O(\log(k)/k) \cdot n}}$*:*

1. *Can solve k-SAT in time* $\mathbf{2^{n - O(\log(k)/k) \cdot n}}$*, showing SSETH is False.*
2. *Middle slice function requires* $\Sigma\Pi\Sigma_k$ *circuits of size* $\mathbf{2^{\Omega(\log(k)/k) \cdot n}}$ *and* $\Sigma\Pi\Sigma$ *circuits of size* $\mathbf{2^{\Omega(\sqrt{n \log n})}}$.

# Our Results

# Our Results

**Theorem**

*NAE-Enum*$(3, n/2)$ *can be solved in time* $\mathbf{6^{n/4} \approx 1.565^n}$.

# Our Results

**Theorem**

*NAE-Enum*$(3, n/2)$ *can be solved in time* $\mathbf{6^{n/4} \approx 1.565^n}$.

**This is Tight**

By construction, NAE-Enum$(3, n/2)$ requires time $\geq \mathbf{6^{n/4} \approx 1.565^n}$.

# Solving NAE-Enum$(k, r)$

# Solving NAE-Enum$(k, r)$

0. Given $k$-CNF $G$, let $F(x) = G(x) \wedge G(\overline{x})$. Every satisfying assignment of $F$ is NAE-satisfying assignment of $G$ and $F$.

# Solving NAE-Enum$(k, r)$

0. Given $k$-CNF $G$, let $F(x) = G(x) \wedge G(\overline{x})$. Every satisfying assignment of $F$ is NAE-satisfying assignment of $G$ and $F$.
1. Associate "Transversal Tree" with $F$ that captures satisfying assignments of Hamming weight $r$ - "transversals".

# Solving NAE-Enum$(k, r)$

0. Given $k$-CNF $G$, let $F(x) = G(x) \wedge G(\overline{x})$. Every satisfying assignment of $F$ is NAE-satisfying assignment of $G$ and $F$.
1. Associate "Transversal Tree" with $F$ that captures satisfying assignments of Hamming weight $r$ - "transversals".
2. Randomly prune Transversal Tree s.t. every transversal corresponds to unique leaf.

# Solving NAE-Enum$(k, r)$

0. Given $k$-CNF $G$, let $F(x) = G(x) \wedge G(\overline{x})$. Every satisfying assignment of $F$ is NAE-satisfying assignment of $G$ and $F$.
1. Associate "Transversal Tree" with $F$ that captures satisfying assignments of Hamming weight $r$ - "transversals".
2. Randomly prune Transversal Tree s.t. every transversal corresponds to unique leaf.
3. Visit all leaves of this pruned Transversal Tree.

# Transversal Tree

# Transversal Tree

## Definition (Transversal Tree)

Given $k$-CNF $F$:

# Transversal Tree

**Definition (Transversal Tree)**

Given $k$-CNF $F$:

- If $F$ doesn't contain monotone clause, create leaf node.

# Transversal Tree

## Definition (Transversal Tree)

Given $k$-CNF $F$:

- If $F$ doesn't contain monotone clause, create leaf node.
- Else, pick monotone clause $C$ (carefully), and for each $x \in C$:

# Transversal Tree

## Definition (Transversal Tree)

Given $k$-CNF $F$:

- If $F$ doesn't contain monotone clause, create leaf node.
- Else, pick monotone clause $C$ (carefully), and for each $x \in C$:
    1. Recursively construct tree $T_x$ for $F_{x \leftarrow 1}$

# Transversal Tree

## Definition (Transversal Tree)

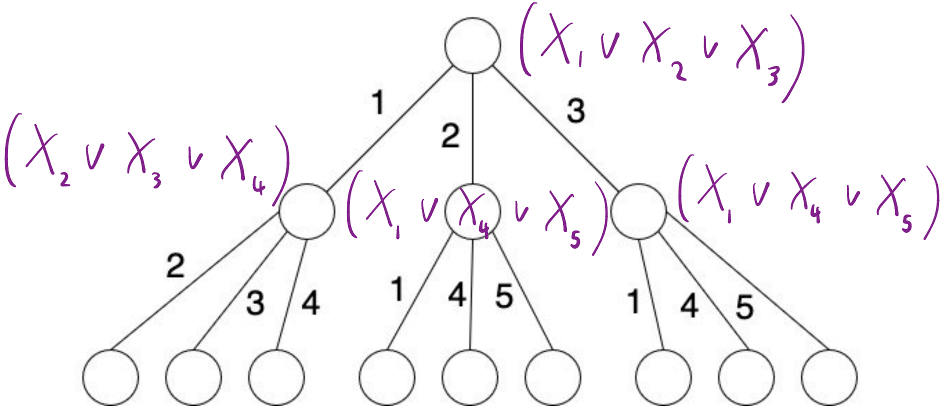
Given $k$-CNF $F$:

- If $F$ doesn't contain monotone clause, create leaf node.
- Else, pick monotone clause $C$ (carefully), and for each $x \in C$:
  1. Recursively construct tree $T_x$ for $F_{x \leftarrow 1}$
  2. Add edge from current node to $T_x$ labelled $x$.

# Transversal Tree

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

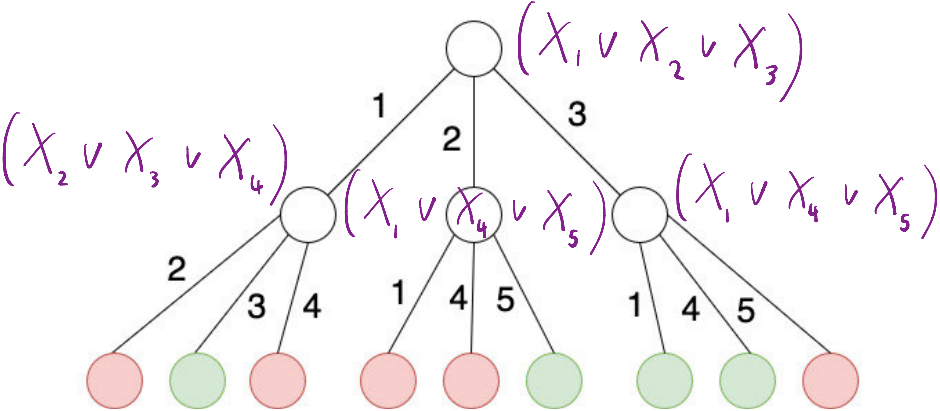
AND (negations of these clauses)



# Transversal Tree

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

AND (negations of these clauses)



# Transversal Tree Properties

# Transversal Tree Properties

**Lemma**

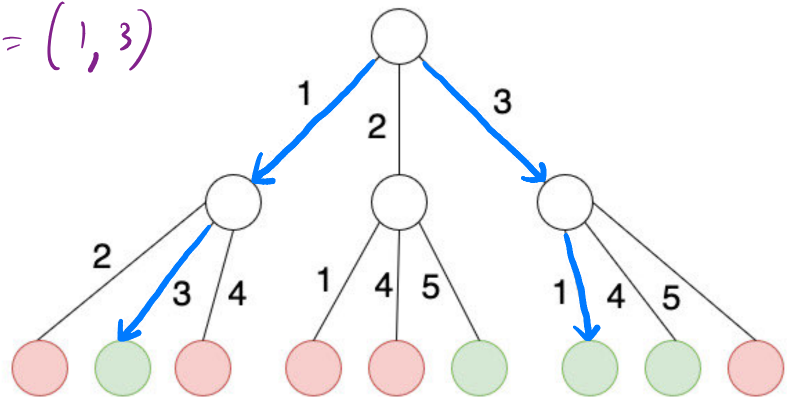
*Every "transversal" of $F$ corresponds to some (maybe many) valid leaves of $T$.*

# Transversal Tree Properties

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

AND (negations of these clauses)

$\alpha = (1, 3)$



# Pruning Transversal Tree

## Visiting all Transversals Once

# Pruning Transversal Tree

## Visiting all Transversals Once

1. At a node, let outgoing edges be ordered as $x_1, \ldots, x_k$.

# Pruning Transversal Tree

## Visiting all Transversals Once

1. At a node, let outgoing edges be ordered as $x_1, \ldots, x_k$.
2. For $1 \leq i \leq k$:

# Pruning Transversal Tree

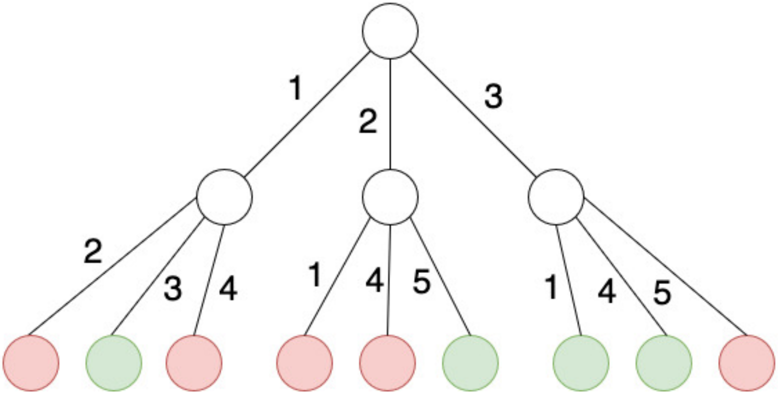
## Visiting all Transversals Once

1. At a node, let outgoing edges be ordered as $x_1, \ldots, x_k$.
2. For $1 \leq i \leq k$:
   2.1 Prune (delete) all edges in $T_{x_i}$ labelled with any of $x_1, \ldots, x_{i-1}$ & search $T_{x_i}$.

# Pruning: Left to Right Ordering

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

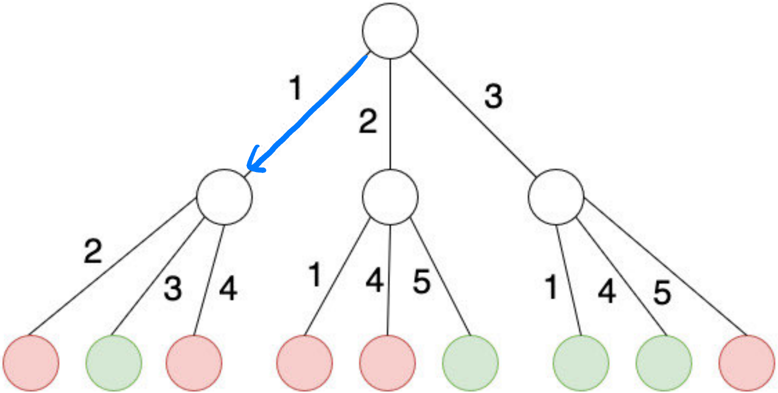
AND (negations of these clauses)



# Pruning: Left to Right Ordering

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

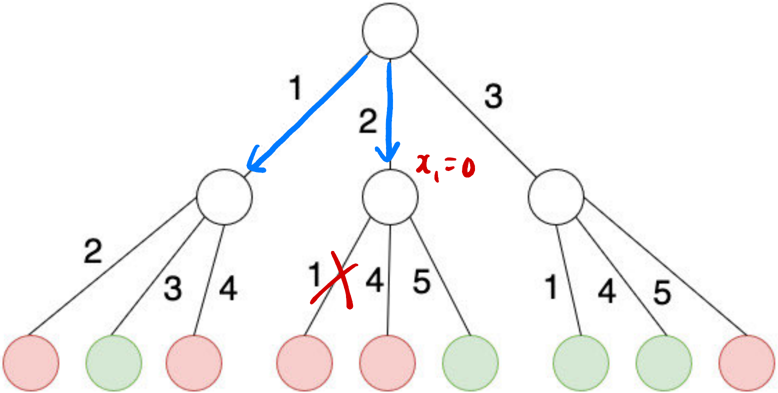
AND (negations of these clauses)



# Pruning: Left to Right Ordering

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

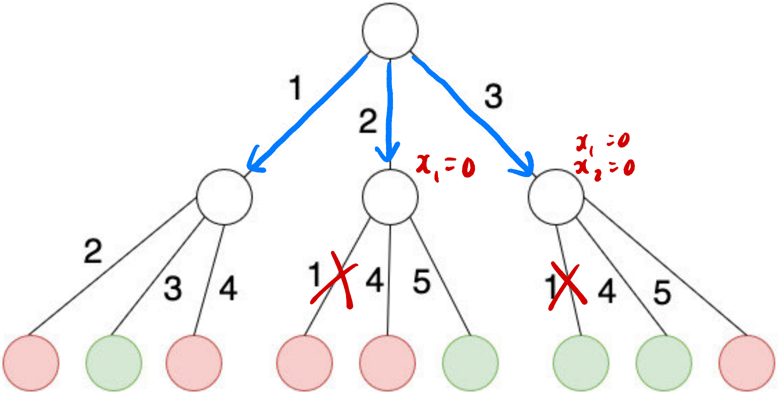
AND (negations of these clauses)



# Pruning: Left to Right Ordering

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

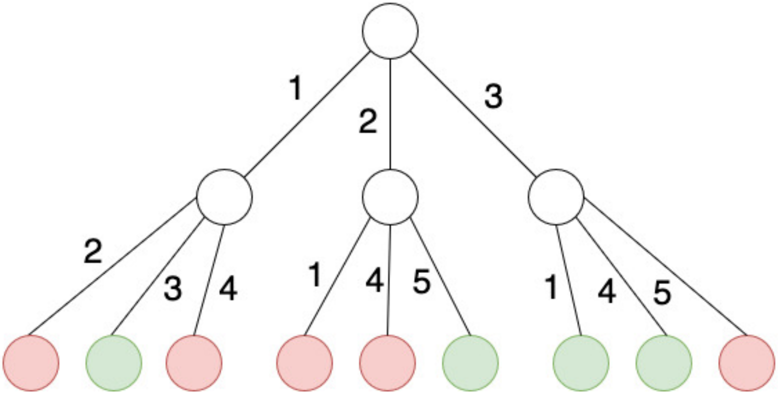
AND (negations of these clauses)



# Pruning: Right to Left Ordering

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

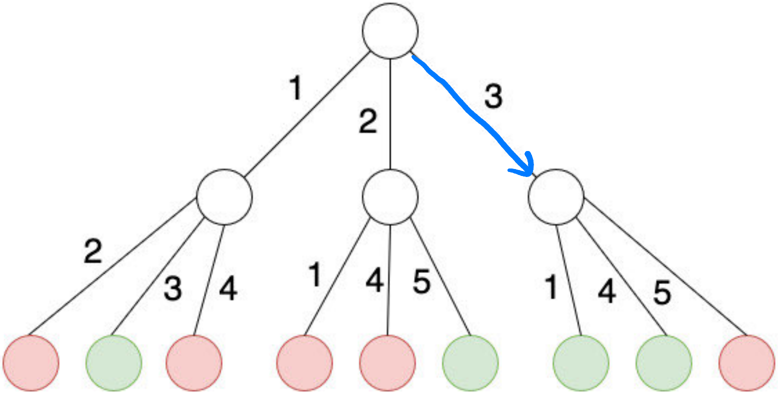
AND (negations of these clauses)



# Pruning: Right to Left Ordering

$$F = (X_1 \lor X_2 \lor X_3)(X_1 \lor X_4 \lor X_5)(X_2 \lor X_3 \lor X_4)(X_3 \lor X_5)$$

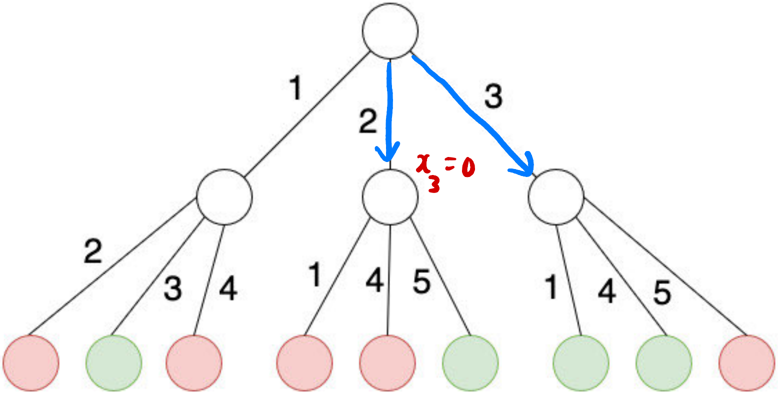
AND (negations of these clauses)



# Pruning: Right to Left Ordering

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

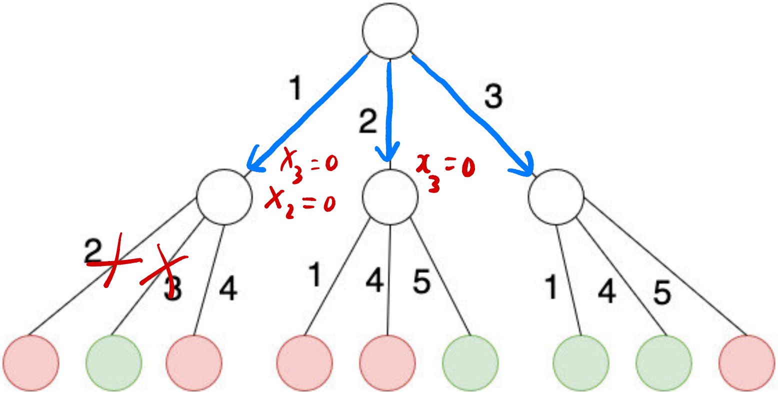
AND (negations of these clauses)



# Pruning: Right to Left Ordering

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

AND (negations of these clauses)



# Which Ordering?

# Which Ordering?

## Canonical Ordering

Same as $k$-SAT algorithm by Monien-Speckenmeyer [ Monien-Speckenmeyer'85 ].

# Which Ordering?

## Canonical Ordering

Same as $k$-SAT algorithm by Monien-Speckenmeyer [ Monien-Speckenmeyer'85 ].

## Previous and This work: Randomized Ordering

Randomize order of outgoing edges at every node.

# Survival Probability

# Survival Probability

## Survival Probability of Edge

For edge $e$ let $\sigma(e) = \Pr[\text{e not pruned under randomized ordering}]$.

# Survival Probability

## Survival Probability of Edge

For edge $e$ let $\sigma(e) = \Pr[\text{e not pruned under randomized ordering}]$.

## Survival Probability of Path

For path $P$:
$\sigma(P) = \Pr[\text{all edges on path not pruned under randomized ordering}]$.

# Survival Probability

## Survival Probability of Edge

For edge $e$ let $\sigma(e) = \Pr[\text{e not pruned under randomized ordering}]$.

## Survival Probability of Path

For path $P$:
$\sigma(P) = \Pr[\text{all edges on path not pruned under randomized ordering}]$.

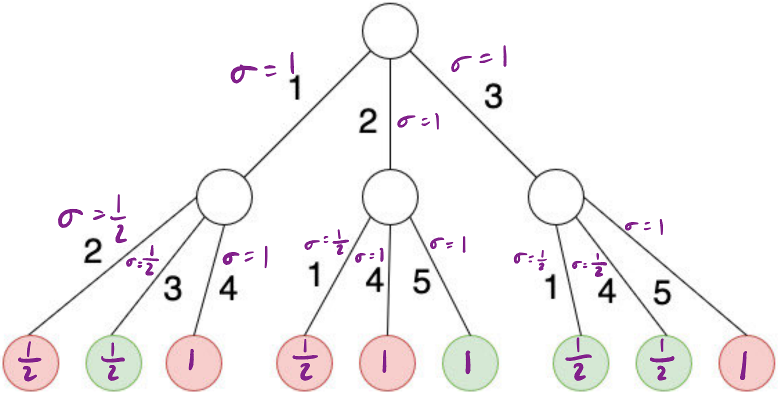
## Lemma (Expected Runtime of Our Algorithm)

*Expected runtime under randomized ordering is* $\sigma(T) = \sum_{\ell \in leaf(T)} \sigma(P_{root,\ell})$

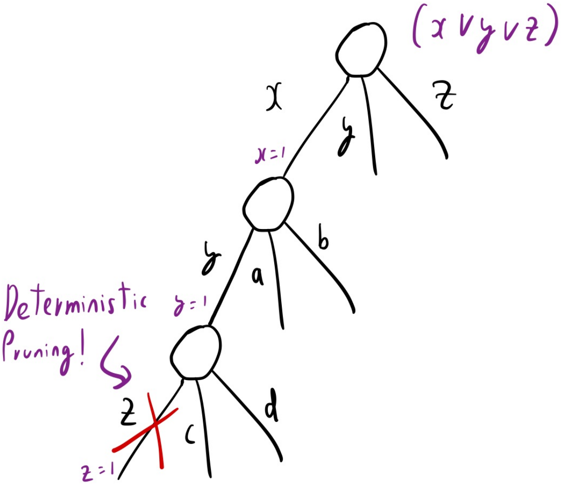
# Survival Probability

$$F = (X_1 \vee X_2 \vee X_3)(X_1 \vee X_4 \vee X_5)(X_2 \vee X_3 \vee X_4)(X_3 \vee X_5)$$

AND (negations of these clauses)



# Why NAE Helps - Deterministic Pruning



# Remaining Analysis

# Remaining Analysis

- Use NAE assumption to force deterministic pruning, guaranteeing certain clauses must exist (as well as not exist).

# Remaining Analysis

- Use NAE assumption to force deterministic pruning, guaranteeing certain clauses must exist (as well as not exist).
- Use and refine tools from GPPST'24 (NAE assumption simplifies this analysis).

# Remaining Analysis

- Use NAE assumption to force deterministic pruning, guaranteeing certain clauses must exist (as well as not exist).
- Use and refine tools from GPPST'24 (NAE assumption simplifies this analysis).
- With careful accounting, conclude

$$\sigma(T) \leq 6^{n/4}.$$

# Open Questions

# Open Questions

- Prove non-trivial bounds for NAE-Enum$(k, n/2)$ for $k > 3$.

# Open Questions

- Prove non-trivial bounds for NAE-Enum$(k, n/2)$ for $k > 3$.
- Prove that Enum$(3, n/2)$ can be solved in time $\mathbf{6^{n/4}}$.

# Open Questions

- Prove non-trivial bounds for NAE-Enum$(k, n/2)$ for $k > 3$.
- Prove that Enum$(3, n/2)$ can be solved in time $\mathbf{6^{n/4}}$.
- Prove that in every $3$-uniform hypergraph with transversal number $n/2$, number of transversals of size $n/2$ is $\leq \mathbf{6^{n/4}}$.