# Optimal Depth-Three Circuits for Inner Product


Mohit Gurumukhani
Cornell University

Daniel Kleber
UC San Diego

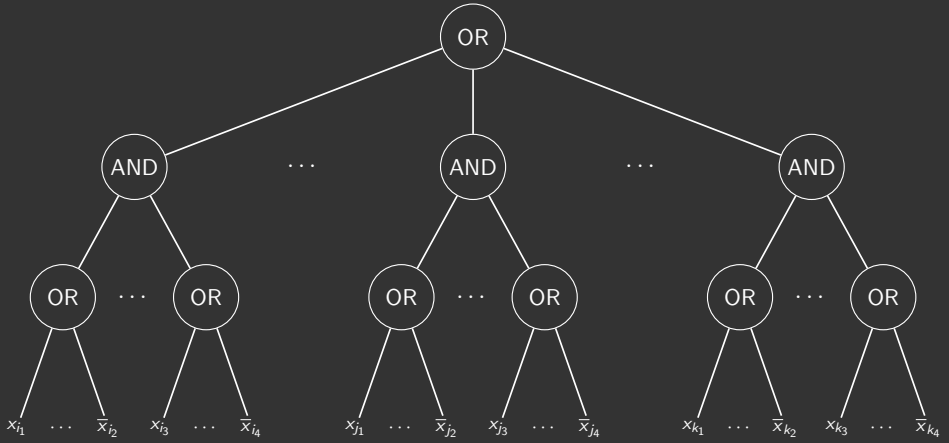
Ramamohan Paturi
UC San Diego

Christopher Rosin
Constructive Codes

Navid Talebanfard
University of Sheffield

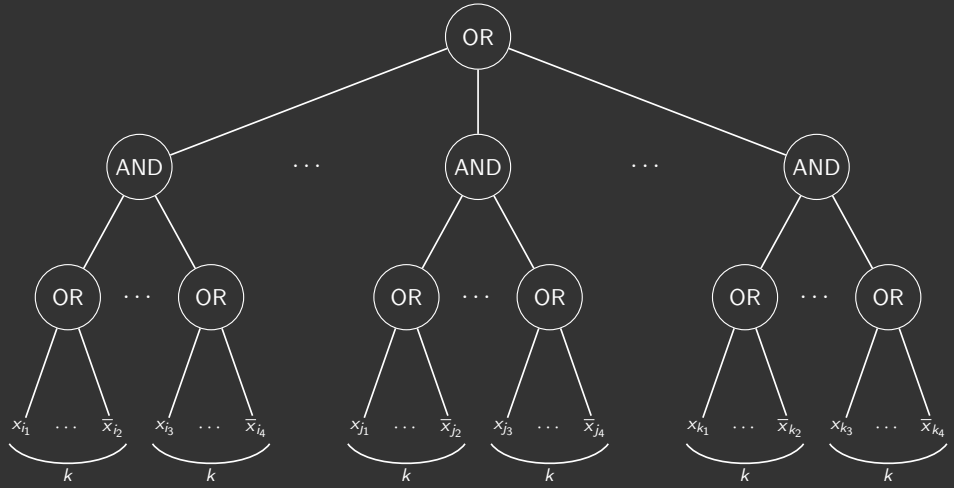
# Introduction

# Depth 3 Circuits




$\Sigma\Pi\Sigma$ circuit

# Depth 3 Circuits with fan in $k$




$\Sigma\Pi\Sigma_k$ circuit

## Depth 3 Circuits with fan in $k$

A $\Sigma\Pi\Sigma_k$ circuit $C$ is **OR** of ***k*-CNFs**

## Depth 3 Circuits with fan in $k$

A $\Sigma\Pi\Sigma_k$ circuit $C$ is **OR** of ***k*-CNFs**

$$C = \bigvee_{i=1}^{s} F_i$$

## Depth 3 Circuits with fan in $k$

A $\Sigma\Pi\Sigma_k$ circuit $C$ is **OR** of ***k*-CNFs**

$$C = \bigvee_{i=1}^{s} F_i$$

$$\text{Size}(C) = s$$

## Depth 3 Circuits with fan in $k$

A $\Sigma\Pi\Sigma_k$ circuit $C$ is **OR** of ***k*-CNFs**

$$C = \bigvee_{i=1}^{s} F_i$$

$$\text{Size}(C) = s$$

**$\Sigma\Pi\Sigma_k$ complexity of $f$ (upto $2^{o(n)}$ factors)**

For $f : \{0,1\}^n \to \{0,1\}$, what is **minimum** sized $\Sigma\Pi\Sigma_k$ circuit $C$ computing $f$?

# Why care about $\Sigma\Pi\Sigma_k$ circuits?

# Why care about $\Sigma\Pi\Sigma_k$ circuits?

**Rich Connections to other circuit models**

## Why care about $\Sigma\Pi\Sigma_k$ circuits?

**Rich Connections to other circuit models**

- Valiant'77 : If for all const $k$, $\Sigma\Pi\Sigma_k$ complexity of $f$ is $\geq 2^{0.01n}$, then $f$ requires **super linear** size, log depth series-parallel circuits

## Why care about $\Sigma\Pi\Sigma_k$ circuits?

**Rich Connections to other circuit models**

- Valiant'77 : If for all const $k$, $\Sigma\Pi\Sigma_{\boldsymbol{k}}$ complexity of $f$ is $\geq \mathbf{2^{0.01n}}$, then $f$ requires **super linear** size, log depth series-parallel circuits
- Golovnev-Kulikov-Williams'21 : If $\Sigma\Pi\Sigma_{\mathbf{16}}$ complexity of $f$ is $\mathbf{2^{n-o(n)}}$, then $f$ requires $\mathbf{3.9n}$ sized unrestricted circuits

## Why care about $\Sigma\Pi\Sigma_k$ circuits?

### Rich Connections to other circuit models

- Valiant'77 : If for all const $k$, $\Sigma\Pi\Sigma_k$ complexity of $f$ is $\geq 2^{0.01n}$, then $f$ requires **super linear** size, log depth series-parallel circuits
- Golovnev-Kulikov-Williams'21 : If $\Sigma\Pi\Sigma_{16}$ complexity of $f$ is $2^{n-o(n)}$, then $f$ requires $3.9n$ sized unrestricted circuits

### Clean model for circuit analysis

To analyze OR of $k$-CNFs, only need to analyze $k$-CNFs

## Why care about $\Sigma\Pi\Sigma_k$ circuits?

**Rich Connections to other circuit models**

- Valiant'77 : If for all const $k$, $\Sigma\Pi\Sigma_k$ complexity of $f$ is $\geq 2^{0.01n}$, then $f$ requires **super linear** size, log depth series-parallel circuits
- Golovnev-Kulikov-Williams'21 : If $\Sigma\Pi\Sigma_{16}$ complexity of $f$ is $2^{n-o(n)}$, then $f$ requires $3.9n$ sized unrestricted circuits

**Clean model for circuit analysis**

To analyze OR of $k$-CNFs, only need to analyze $k$-CNFs

**$k$-CNF Circuit analysis has proven useful**

Usually yields $k$-SAT algorithms e.g. Paturi-Pudlák-Saks-Zane'05

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

## An almost Exhaustive List

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

**An almost Exhaustive List**

- Paturi-Púdlak-Zane'99 : $\Sigma\Pi\Sigma_k$ complexity of Parity is $2^{n/k}$

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

**An almost Exhaustive List**

- Paturi-Púdlak-Zane'99 : $\Sigma\Pi\Sigma_k$ complexity of Parity is $2^{n/k}$
- Haståd-Jukna-Pudlák'95 : $\Sigma\Pi\Sigma_2$ complexity of Majority is $2^{n/2}$

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

**An almost Exhaustive List**

- Paturi-Púdlak-Zane'99 : $\Sigma\Pi\Sigma_k$ complexity of Parity is $2^{n/k}$
- Haståd-Jukna-Pudlák'95 : $\Sigma\Pi\Sigma_2$ complexity of Majority is $2^{n/2}$
- Paturi-Saks-Zane'00 : $\Sigma\Pi\Sigma_2$ complexity of 'affine dispersers' is $2^{n-o(n)}$

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

## An almost Exhaustive List

- Paturi-Púdlak-Zane'99 : $\Sigma\Pi\Sigma_k$ complexity of Parity is $2^{n/k}$
- Haståd-Jukna-Pudlák'95 : $\Sigma\Pi\Sigma_2$ complexity of Majority is $2^{n/2}$
- Paturi-Saks-Zane'00 : $\Sigma\Pi\Sigma_2$ complexity of 'affine dispersers' is $2^{n-o(n)}$

## Affine dispersers and friends

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

## An almost Exhaustive List

- Paturi-Púdlak-Zane'99 : $\Sigma\Pi\Sigma_k$ complexity of Parity is $2^{n/k}$
- Haståd-Jukna-Pudlák'95 : $\Sigma\Pi\Sigma_2$ complexity of Majority is $2^{n/2}$
- Paturi-Saks-Zane'00 : $\Sigma\Pi\Sigma_2$ complexity of 'affine dispersers' is $2^{n-o(n)}$

## Affine dispersers and friends

- Functions with pseudorandom properties

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

## An almost Exhaustive List

- Paturi-Púdlak-Zane'99 : $\Sigma\Pi\Sigma_k$ complexity of Parity is $2^{n/k}$
- Haståd-Jukna-Pudlák'95 : $\Sigma\Pi\Sigma_2$ complexity of Majority is $2^{n/2}$
- Paturi-Saks-Zane'00 : $\Sigma\Pi\Sigma_2$ complexity of 'affine dispersers' is $2^{n-o(n)}$

## Affine dispersers and friends

- Functions with pseudorandom properties
- Have asymptotically **optimal explicit** constructions Li'23

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

## An almost Exhaustive List

- Paturi-Púdlak-Zane'99 : $\Sigma\Pi\Sigma_k$ complexity of Parity is $2^{n/k}$
- Haståd-Jukna-Pudlák'95 : $\Sigma\Pi\Sigma_2$ complexity of Majority is $2^{n/2}$
- Paturi-Saks-Zane'00 : $\Sigma\Pi\Sigma_2$ complexity of 'affine dispersers' is $2^{n-o(n)}$

## Affine dispersers and friends

- Functions with pseudorandom properties
- Have asymptotically **optimal explicit** constructions Li'23
- Useful for circuit lower bounds Find-Golovnev-Hirsch-Kulikov'16, Li-Yang'22

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

## An almost Exhaustive List

- Paturi-Púdlak-Zane'99 : $\Sigma\Pi\Sigma_k$ complexity of Parity is $2^{n/k}$
- Haståd-Jukna-Pudlák'95 : $\Sigma\Pi\Sigma_2$ complexity of Majority is $2^{n/2}$
- Paturi-Saks-Zane'00 : $\Sigma\Pi\Sigma_2$ complexity of 'affine dispersers' is $2^{n-o(n)}$

## Affine dispersers and friends

- Functions with pseudorandom properties
- Have asymptotically **optimal explicit** constructions Li'23
- Useful for circuit lower bounds Find-Golovnev-Hirsch-Kulikov'16, Li-Yang'22
- Could have $2^{n-o(n)}$ $\Sigma\Pi\Sigma_k$ complexity

# $\Sigma\Pi\Sigma_k$-complexity of explicit functions

## An almost Exhaustive List

- Paturi-Púdlak-Zane'99 : $\Sigma\Pi\Sigma_k$ complexity of Parity is $2^{n/k}$
- Haståd-Jukna-Pudlák'95 : $\Sigma\Pi\Sigma_2$ complexity of Majority is $2^{n/2}$
- Paturi-Saks-Zane'00 : $\Sigma\Pi\Sigma_2$ complexity of 'affine dispersers' is $2^{n-o(n)}$

## Affine dispersers and friends

- Functions with pseudorandom properties
- Have asymptotically **optimal explicit** constructions Li'23
- Useful for circuit lower bounds Find-Golovnev-Hirsch-Kulikov'16, Li-Yang'22
- Could have $2^{n-o(n)}$ $\Sigma\Pi\Sigma_k$ complexity

## The best lower bound

Paturi-Púdlak-Saks-Zane'05 : $\Sigma\Pi\Sigma_k$ complexity of indicator of code is $\geq 2^{1.6 \cdot n/k}$


Inner Product

# $\Sigma\Pi\Sigma_k$-complexity of Inner Product

# $\Sigma\Pi\Sigma_k$-complexity of Inner Product

**Definition**

$\mathrm{IP} : \{0,1\}^{2n} \to \{0,1\}$ defined as

$$\mathrm{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigoplus_{i=1}^{n} (x_i \wedge y_i)$$

# $\Sigma\Pi\Sigma_k$-complexity of Inner Product

**Definition**

$\mathrm{IP} : \{0,1\}^{2n} \to \{0,1\}$ defined as

$$\mathrm{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigoplus_{i=1}^{n} (x_i \wedge y_i)$$

**Why IP?**

# $\Sigma\Pi\Sigma_k$-complexity of Inner Product

**Definition**

$\mathrm{IP} : \{0,1\}^{2n} \to \{0,1\}$ defined as

$$\mathrm{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigoplus_{i=1}^{n} (x_i \wedge y_i)$$

**Why IP?**

- IP ubiquitous in complexity, pseudorandomness, cryptography, coding theory

# $\Sigma\Pi\Sigma_k$-complexity of Inner Product

**Definition**

$\mathrm{IP} : \{0,1\}^{2n} \to \{0,1\}$ defined as

$$\mathrm{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigoplus_{i=1}^{n} (x_i \wedge y_i)$$

**Why IP?**

- IP ubiquitous in complexity, pseudorandomness, cryptography, coding theory
- IP is **decent** affine disperser; step towards $\Sigma\Pi\Sigma_k$ complexity of **affine dispersers**

# $\Sigma\Pi\Sigma_k$-complexity of Inner Product

**Definition**

$\mathrm{IP} : \{0,1\}^{2n} \to \{0,1\}$ defined as

$$\mathrm{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigoplus_{i=1}^{n} (x_i \wedge y_i)$$

**Why IP?**

- IP ubiquitous in complexity, pseudorandomness, cryptography, coding theory
- IP is **decent** affine disperser; step towards $\Sigma\Pi\Sigma_k$ complexity of **affine dispersers**

**History**

# $\Sigma\Pi\Sigma_k$-complexity of Inner Product

**Definition**

$\mathrm{IP} : \{0,1\}^{2n} \to \{0,1\}$ defined as

$$\mathrm{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigoplus_{i=1}^{n} (x_i \wedge y_i)$$

**Why IP?**

- IP ubiquitous in complexity, pseudorandomness, cryptography, coding theory
- IP is **decent** affine disperser; step towards $\Sigma\Pi\Sigma_k$ complexity of **affine dispersers**

**History**

- Golovnev-Kulikov-Williams'21 : What is the $\Sigma\Pi\Sigma_3$ complexity of IP?

# $\Sigma\Pi\Sigma_k$-complexity of Inner Product

**Definition**

$\mathrm{IP} : \{0,1\}^{2n} \to \{0,1\}$ defined as

$$\mathrm{IP}(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigoplus_{i=1}^{n} (x_i \wedge y_i)$$

**Why IP?**

- IP ubiquitous in complexity, pseudorandomness, cryptography, coding theory
- IP is **decent** affine disperser; step towards $\Sigma\Pi\Sigma_k$ complexity of **affine dispersers**

**History**

- Golovnev-Kulikov-Williams'21 : What is the $\Sigma\Pi\Sigma_3$ complexity of IP?
- Frankl-Gryaznov-Talebanfard'21 : What is the $\Sigma\Pi\Sigma_2$ complexity of IP?

# Problem at Hand

# $\Sigma\Pi\Sigma_2$-complexity of Inner Product

## $\Sigma\Pi\Sigma_2$-complexity of Inner Product

Let $\Sigma\Pi\Sigma_2$ complexity of IP be $2^{\delta n}$

**Previous Works**

# $\Sigma\Pi\Sigma_2$-complexity of Inner Product

Let $\Sigma\Pi\Sigma_2$ complexity of IP be $2^{\delta n}$

**Previous Works**

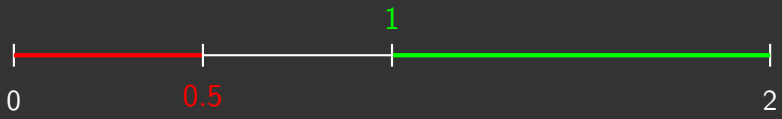
- Trivially, $0 \leq \delta \leq 2$

# $\Sigma\Pi\Sigma_2$-complexity of Inner Product

Let $\Sigma\Pi\Sigma_2$ complexity of IP be $2^{\delta n}$

**Previous Works**

- Trivially, $0 \leq \delta \leq 2$
- Golovnev-Kulikov-Williams'21 : $0.5 \leq \delta \leq 1$

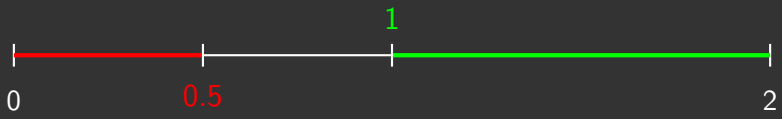


# $\Sigma\Pi\Sigma_2$-complexity of Inner Product

Let $\Sigma\Pi\Sigma_2$ complexity of IP be $2^{\delta n}$

**Previous Works**

- Trivially, $0 \leq \delta \leq 2$
- Golovnev-Kulikov-Williams'21 : $0.5 \leq \delta \leq 1$
- Frankl-Gryaznov-Talebanfard'21 : $0.4 \leq \delta$

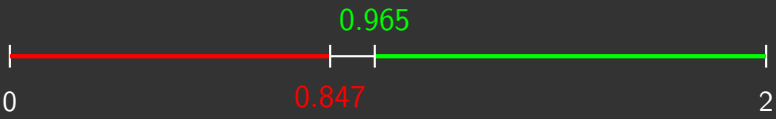


## $\Sigma\Pi\Sigma_2$-complexity of Inner Product

Let $\Sigma\Pi\Sigma_2$ complexity of IP be $2^{\delta n}$

**Previous Works**

- Trivially, $0 \leq \delta \leq 2$
- Golovnev-Kulikov-Williams'21 : $0.5 \leq \delta \leq 1$
- Frankl-Gryaznov-Talebanfard'21 : $0.4 \leq \delta$
- Göös-Guan-Mosnoi'24 : $0.847... \leq \delta \leq 0.965$

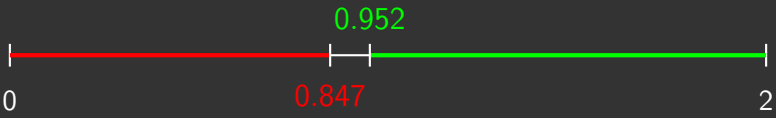


# $\Sigma\Pi\Sigma_2$-complexity of Inner Product

Let $\Sigma\Pi\Sigma_2$ complexity of IP be $2^{\delta n}$

**Previous Works**

- Trivially, $0 \leq \delta \leq 2$
- Golovnev-Kulikov-Williams'21 : $0.5 \leq \delta \leq 1$
- Frankl-Gryaznov-Talebanfard'21 : $0.4 \leq \delta$
- Göös-Guan-Mosnoi'24 : $0.847... \leq \delta \leq 0.965$
- Amano'23 : $\delta \leq 0.952$



## $\Sigma\Pi\Sigma_2$-complexity of Inner Product

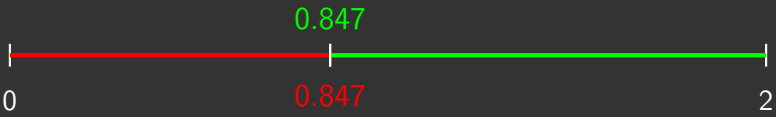
Let $\Sigma\Pi\Sigma_2$ complexity of IP be $2^{\delta n}$

**Previous Works**

- Trivially, $0 \leq \delta \leq 2$
- Golovnev-Kulikov-Williams'21 : $0.5 \leq \delta \leq 1$
- Frankl-Gryaznov-Talebanfard'21 : $0.4 \leq \delta$
- Göös-Guan-Mosnoi'24 : $0.847... \leq \delta \leq 0.965$
- Amano'23 : $\delta \leq 0.952$

**Our Result**

$\Sigma\Pi\Sigma_2$ complexity of IP is $2^{0.847...n} = (1.8)^n$ (upto poly($n$) factors)



# Constructing Optimal Circuits for IP

## Goal

Construct $\Sigma\Pi\Sigma_2$ circuit of size $(1.8)^n$ for IP

$$\mathrm{IP} = \bigvee_{i=1}^{s=(1.8)^n} F_i$$

## Goal

Construct $\Sigma\Pi\Sigma_2$ circuit of size $(1.8)^n$ for IP

$$\mathrm{IP} = \bigvee_{i=1}^{s=(1.8)^n} F_i$$

**Observe (Consistent 2-CNFs)**

For each 2-CNF $F_i$, $F_i(x) = 1 \implies \mathrm{IP}(x) = 1$

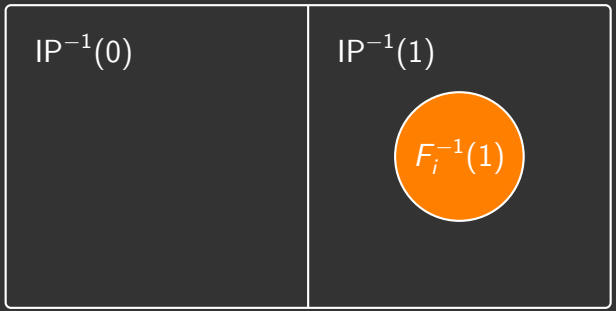
## Goal

Construct $\Sigma\Pi\Sigma_2$ circuit of size $(1.8)^n$ for IP

$$\mathrm{IP} = \bigvee_{i=1}^{s=(1.8)^n} F_i$$

**Observe (Consistent 2-CNFs)**

For each 2-CNF $F_i$, $F_i(x) = 1 \implies \mathrm{IP}(x) = 1$



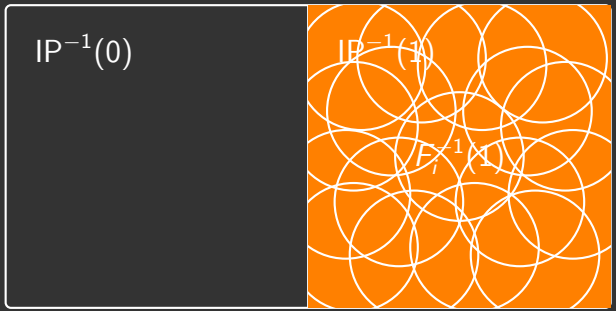
## Goal

Construct $\Sigma\Pi\Sigma_2$ circuit of size $(1.8)^n$ for IP

$$\mathrm{IP} = \bigvee_{i=1}^{s=(1.8)^n} F_i$$

**Observe (Consistent 2-CNFs)**

For each 2-CNF $F_i$, $F_i(x) = 1 \implies \mathrm{IP}(x) = 1$



# Recipe to Consistently Cover Slice function

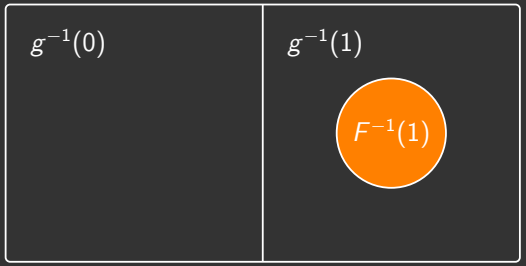
## Recipe to Consistently Cover Slice function

**For** $g(x) = 1 \iff$ **Hamming Weight**$(x) = n/2$

# Recipe to Consistently Cover Slice function

**For $g(x) = 1 \iff$ Hamming Weight$(x) = n/2$**

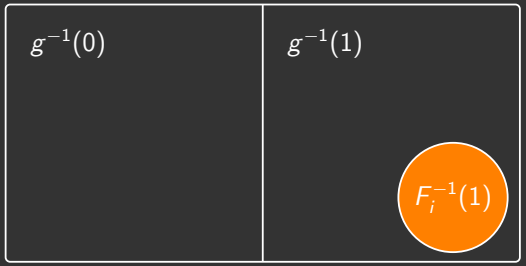
- Construct **one** consistent $F$ covering **large fraction** of $g^{-1}(1)$



## Recipe to Consistently Cover Slice function

**For** $g(x) = 1 \iff$ **Hamming Weight**$(x) = n/2$

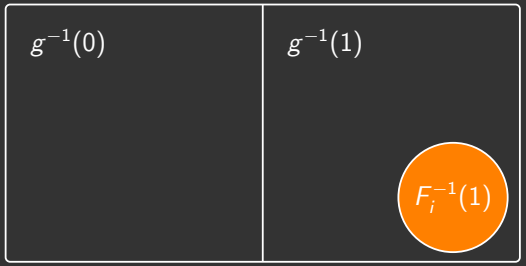
- Construct **one** consistent $F$ covering **large fraction** of $g^{-1}(1)$
- Define $\boldsymbol{F_i = F(\pi_i(x_1, \ldots, x_n))}$ for **random permutation** $\pi_i$



## Recipe to Consistently Cover Slice function

**For $g(x) = 1 \iff$ Hamming Weight$(x) = n/2$**

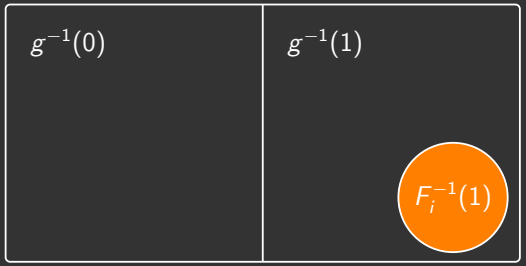
- Construct **one** consistent $F$ covering **large fraction** of $g^{-1}(1)$
- Define $\boldsymbol{F_i = F(\pi_i(x_1, \ldots, x_n))}$ for **random permutation** $\pi_i$
- Each $F_i$ is **consistent**



## Recipe to Consistently Cover Slice function

**For $g(x) = 1 \iff$ Hamming Weight$(x) = n/2$**

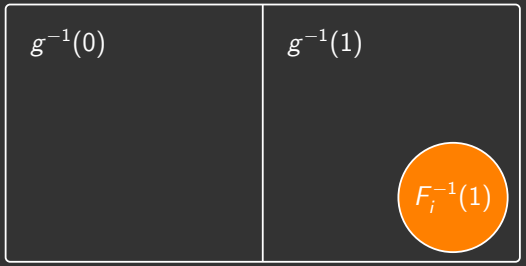
- Construct **one** consistent $F$ covering **large fraction** of $g^{-1}(1)$
- Define $\boldsymbol{F_i = F(\pi_i(x_1, \ldots, x_n))}$ for **random permutation** $\pi_i$
- Each $F_i$ is **consistent**
- For $\alpha \in g^{-1}(1)$, $\mathbf{Pr[\alpha \in F_i^{-1}(1)] = \frac{|F^{-1}(1)|}{|g^{-1}(1)|}}$



## Recipe to Consistently Cover Slice function

**For $g(x) = 1 \iff$ Hamming Weight$(x) = n/2$**

- Construct **one** consistent $F$ covering **large fraction** of $g^{-1}(1)$
- Define $\boldsymbol{F_i = F(\pi_i(x_1, \ldots, x_n))}$ for **random permutation** $\pi_i$
- Each $F_i$ is **consistent**
- For $\alpha \in g^{-1}(1)$, $\mathbf{Pr[\alpha \in F_i^{-1}(1)] = \frac{|F^{-1}(1)|}{|g^{-1}(1)|}}$
- Final circuit $\bigvee_{i=1}^{s} F_i$, where $\boldsymbol{s = \frac{|g^{-1}(1)|}{|F^{-1}(1)|} \cdot \mathrm{poly}(n)}$



# Recipe to Consistently Cover $\mathbb{P}^{-1}(1)$

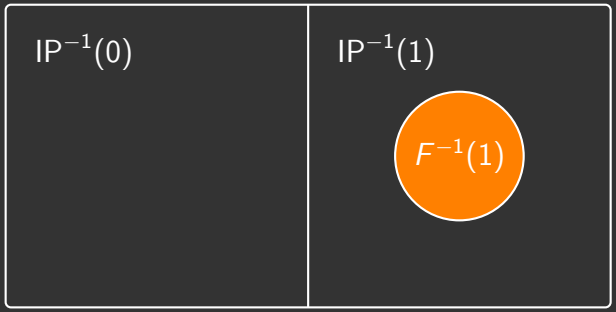
# Recipe to Consistently Cover $\mathbb{P}^{-1}(1)$

## An Attempt

# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## An Attempt

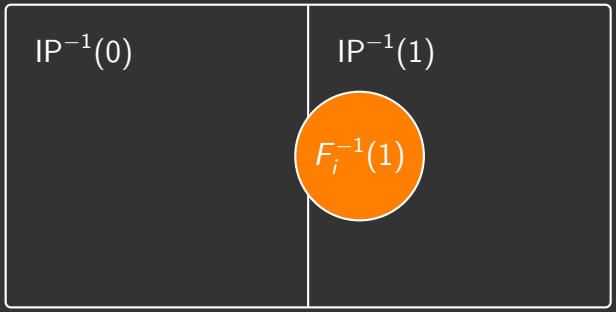
- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$



# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## An Attempt

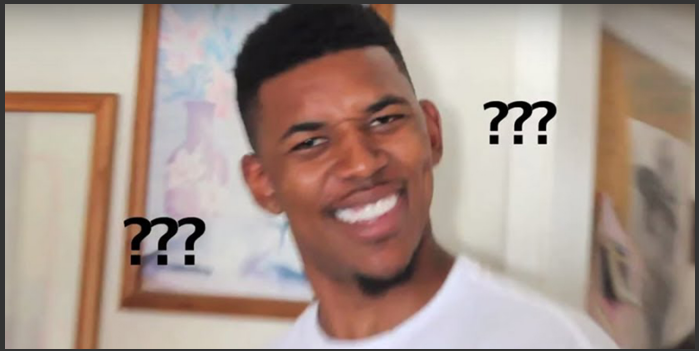
- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$



# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## An Attempt

- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$
- $F_i$ MIGHT NOT be consistent with IP



## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

### An Attempt

- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$
- $F_i$ MIGHT NOT be consistent with IP

| $\alpha$ | $\mathbf{IP}(\alpha)$ |
|---|---|
| $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad y = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ | 1 |

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

### An Attempt

- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$
- $F_i$ MIGHT NOT be consistent with IP

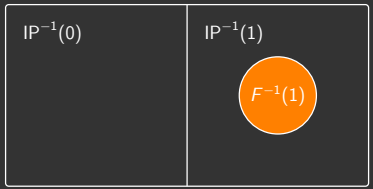
| $\alpha$ | $\mathbf{IP}(\alpha)$ |
|---|---|
| $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad y = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ | 1 |

| $\pi(\alpha)$ | $\mathbf{IP}(\pi(\alpha))$ |
|---|---|
| $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad y = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ | 0 |

# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## Another Attempt

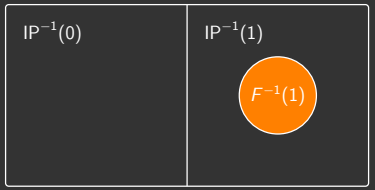
- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$



# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## Another Attempt

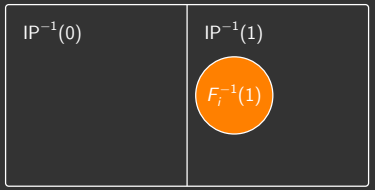
- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Let $\mathsf{Aut}_{\mathsf{IP}}$ be permutations $\pi$ that can be generated by:
  1. Swapping $(x_a, y_a)$ with $(x_b, y_b)$
  2. Swapping $(x_a, y_a)$ with $(y_a, x_a)$



# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## Another Attempt

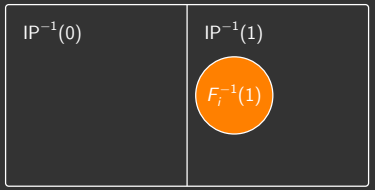
- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Let $\mathsf{Aut}_{\mathsf{IP}}$ be permutations $\pi$ that can be generated by:
  1. Swapping $(x_a, y_a)$ with $(x_b, y_b)$
  2. Swapping $(x_a, y_a)$ with $(y_a, x_a)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$ from $\mathsf{Aut}_{\mathsf{IP}}$



# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## Another Attempt

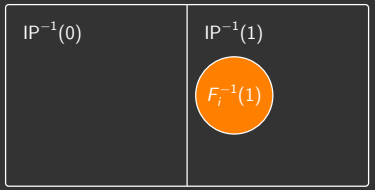
- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Let $\mathsf{Aut}_{\mathsf{IP}}$ be permutations $\pi$ that can be generated by:
  1. Swapping $(x_a, y_a)$ with $(x_b, y_b)$
  2. Swapping $(x_a, y_a)$ with $(y_a, x_a)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$ from $\mathsf{Aut}_{\mathsf{IP}}$
- Each $F_i$ is **consistent** ☺



# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## Another Attempt

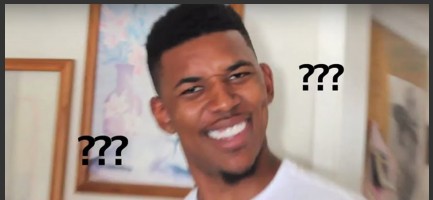
- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Let $\mathsf{Aut}_{\mathsf{IP}}$ be permutations $\pi$ that can be generated by:
  1. Swapping $(x_a, y_a)$ with $(x_b, y_b)$
  2. Swapping $(x_a, y_a)$ with $(y_a, x_a)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$ from $\mathsf{Aut}_{\mathsf{IP}}$
- Each $F_i$ is **consistent** ☺
- Given $\alpha \in \mathsf{IP}^{-1}(1)$, $\mathbf{Pr}[\alpha \in F_i^{-1}(1)] =$



# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## Another Attempt

- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Let $\mathsf{Aut}_{\mathsf{IP}}$ be permutations $\pi$ that can be generated by:
  1. Swapping $(x_a, y_a)$ with $(x_b, y_b)$
  2. Swapping $(x_a, y_a)$ with $(y_a, x_a)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$ from $\mathsf{Aut}_{\mathsf{IP}}$
- Each $F_i$ is **consistent** ☺
- Given $\alpha \in \mathsf{IP}^{-1}(1)$, $\Pr[\alpha \in F_i^{-1}(1)] =$



# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## Another Attempt

- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Let $\mathsf{Aut}_{\mathsf{IP}}$ be permutations $\pi$ that can be generated by:
  1. Swapping $(x_a, y_a)$ with $(x_b, y_b)$
  2. Swapping $(x_a, y_a)$ with $(y_a, x_a)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$ from $\mathsf{Aut}_{\mathsf{IP}}$

| $\alpha_0$ | $\mathsf{IP}(\alpha_0)$ |
|---|---|
| $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad y = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ | 1 |

# Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

## Another Attempt

- Construct **one** consistent $F$ covering **large fraction** of $\mathsf{IP}^{-1}(1)$
- Let $\mathsf{Aut}_{\mathsf{IP}}$ be permutations $\pi$ that can be generated by:
  1. Swapping $(x_a, y_a)$ with $(x_b, y_b)$
  2. Swapping $(x_a, y_a)$ with $(y_a, x_a)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random permutation** $\pi_i$ from $\mathsf{Aut}_{\mathsf{IP}}$

| $\alpha_0$ | $\mathsf{IP}(\alpha_0)$ |
|---|---|
| $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad y = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ | 1 |

| $\alpha_1$ | $\mathsf{IP}(\alpha_1)$ |
|---|---|
| $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad y = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ | 1 |

# Recipe to Consistently Cover $\mathbb{P}^{-1}(1)$

## Orbits

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Orbits**

For $\alpha \in \mathsf{IP}^{-1}(1)$, define

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Orbits**

For $\alpha \in \mathsf{IP}^{-1}(1)$, define

- $d_2(\alpha) = \#i : x_i = y_i = 1$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Orbits**

For $\alpha \in \mathsf{IP}^{-1}(1)$, define

- $d_2(\alpha) = \#i : x_i = y_i = 1$
- $d_1(\alpha) = \#i : x_i = 1, y_i = 0 / x_i = 0, y_i = 1$

## Recipe to Consistently Cover $\mathrm{IP}^{-1}(1)$

**Orbits**

For $\alpha \in \mathrm{IP}^{-1}(1)$, define

- $d_2(\alpha) = \#i : x_i = y_i = 1$
- $d_1(\alpha) = \#i : x_i = 1, y_i = 0 / x_i = 0, y_i = 1$
- $d_0(\alpha) = \#i : x_i = y_i = 0$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Orbits**

For $\alpha \in \mathsf{IP}^{-1}(1)$, define

- $d_2(\alpha) = \#i : x_i = y_i = 1$
- $d_1(\alpha) = \#i : x_i = 1, y_i = 0 / x_i = 0, y_i = 1$
- $d_0(\alpha) = \#i : x_i = y_i = 0$

**Basic observations**

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Orbits**

For $\alpha \in \mathsf{IP}^{-1}(1)$, define

- $d_2(\alpha) = \#i : x_i = y_i = 1$
- $d_1(\alpha) = \#i : x_i = 1, y_i = 0 / x_i = 0, y_i = 1$
- $d_0(\alpha) = \#i : x_i = y_i = 0$

**Basic observations**

- $d_2(\alpha) + d_1(\alpha) + d_0(\alpha) = n$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Orbits**

For $\alpha \in \mathsf{IP}^{-1}(1)$, define

- $d_2(\alpha) = \#i : x_i = y_i = 1$
- $d_1(\alpha) = \#i : x_i = 1, y_i = 0 / x_i = 0, y_i = 1$
- $d_0(\alpha) = \#i : x_i = y_i = 0$

**Basic observations**

- $d_2(\alpha) + d_1(\alpha) + d_0(\alpha) = n$
- $\alpha \in \mathsf{IP}^{-1}(1)$ iff $d_2$ is **odd**

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Orbits**

For $\alpha \in \mathsf{IP}^{-1}(1)$, define

- $d_2(\alpha) = \#i : x_i = y_i = 1$
- $d_1(\alpha) = \#i : x_i = 1, y_i = 0 / x_i = 0, y_i = 1$
- $d_0(\alpha) = \#i : x_i = y_i = 0$

**Basic observations**

- $d_2(\alpha) + d_1(\alpha) + d_0(\alpha) = n$
- $\alpha \in \mathsf{IP}^{-1}(1)$ iff $d_2$ is **odd**
- For $\pi \in \mathsf{Aut}_{\mathsf{IP}}$:
  1. $d_2(\alpha) = d_2(\pi(\alpha))$
  2. $d_1(\alpha) = d_1(\pi(\alpha))$
  3. $d_0(\alpha) = d_0(\pi(\alpha))$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

### Orbits

For $\alpha \in \mathsf{IP}^{-1}(1)$, define

- $d_2(\alpha) = \#i : x_i = y_i = 1$
- $d_1(\alpha) = \#i : x_i = 1, y_i = 0 / x_i = 0, y_i = 1$
- $d_0(\alpha) = \#i : x_i = y_i = 0$

### Basic observations

- $d_2(\alpha) + d_1(\alpha) + d_0(\alpha) = n$
- $\alpha \in \mathsf{IP}^{-1}(1)$ iff $d_2$ is **odd**
- For $\pi \in \mathsf{Aut}_{\mathsf{IP}}$:
  1. $d_2(\alpha) = d_2(\pi(\alpha))$
  2. $d_1(\alpha) = d_1(\pi(\alpha))$
  3. $d_0(\alpha) = d_0(\pi(\alpha))$

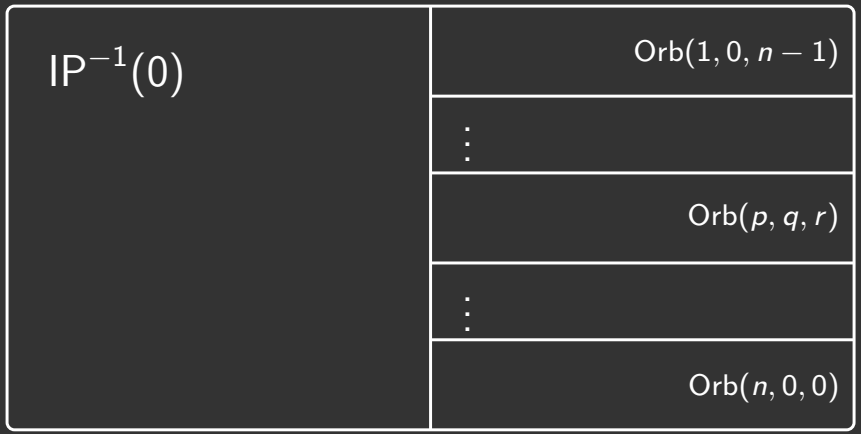
Define $\mathbf{Orb}(p, q, r) = \{\alpha : d_2(\alpha) = p, d_1(\alpha) = q, d_0(\alpha) = r\}$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

Pictorially:

## Recipe to Consistently Cover $\mathrm{IP}^{-1}(1)$

Pictorially:



# Recipe to Consistently Cover $\mathbb{P}^{-1}(1)$

**Third time's the charm Attempt**

## Recipe to Consistently Cover $\mathbb{P}^{-1}(1)$

**Third time's the charm Attempt**

For **all** $p, q, r$ with $p + q + r = \boldsymbol{n}$ and **odd** $\boldsymbol{p}$:

## Recipe to Consistently Cover $\mathbb{P}^{-1}(1)$

**Third time's the charm Attempt**

For **all** $p, q, r$ with $p + q + r = n$ and **odd** $p$:

- Construct **one** consistent $F$ covering **large fraction** of $\mathrm{Orb}(p, q, r)$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Third time's the charm Attempt**

For **all** $p, q, r$ with $p + q + r = n$ and **odd** $p$:

- Construct **one** consistent $F$ covering **large fraction** of $\mathrm{Orb}(p, q, r)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random** $\pi_i \in \mathbf{Aut}_{\mathsf{IP}}$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Third time's the charm Attempt**

For **all** $p, q, r$ with $p + q + r = n$ and **odd** $p$:

- Construct **one** consistent $F$ covering **large fraction** of $\mathrm{Orb}(p, q, r)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random** $\pi_i \in \mathbf{Aut}_{\mathsf{IP}}$
- Each $F_i$ is **consistent**

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Third time's the charm Attempt**

For **all** $p, q, r$ with $p + q + r = n$ and **odd** $p$:

- Construct **one** consistent $F$ covering **large fraction** of $\mathrm{Orb}(p, q, r)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random** $\pi_i \in \mathbf{Aut}_{\mathsf{IP}}$
- Each $F_i$ is **consistent**
- For $\alpha \in \mathrm{Orb}(p, q, r)$, $\Pr[\alpha \in F_i^{-1}(1)] = \frac{|F^{-1}(1) \cap \mathrm{Orb}(p,q,r)|}{|\mathrm{Orb}(p,q,r)|}$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Third time's the charm Attempt**

For **all** $p, q, r$ with $p + q + r = n$ and **odd** $p$:

- Construct **one** consistent $F$ covering **large fraction** of $\mathrm{Orb}(p, q, r)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random** $\pi_i \in \mathbf{Aut}_{\mathsf{IP}}$
- Each $F_i$ is **consistent**
- For $\alpha \in \mathrm{Orb}(p, q, r)$, $\mathbf{Pr}[\alpha \in F_i^{-1}(1)] = \frac{|F^{-1}(1) \cap \mathrm{Orb}(p,q,r)|}{|\mathrm{Orb}(p,q,r)|}$

Final circuit is **OR** of **all** $F_i$ across **all** $p, q, r$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Third time's the charm Attempt**

For **all** $p, q, r$ with $p + q + r = n$ and **odd** $p$:

- Construct **one** consistent $F$ covering **large fraction** of $\mathrm{Orb}(p, q, r)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random** $\pi_i \in \mathbf{Aut}_{\mathsf{IP}}$
- Each $F_i$ is **consistent**
- For $\alpha \in \mathrm{Orb}(p, q, r)$, $\Pr[\alpha \in F_i^{-1}(1)] = \frac{|F^{-1}(1) \cap \mathrm{Orb}(p,q,r)|}{|\mathrm{Orb}(p,q,r)|}$

Final circuit is **OR** of **all** $F_i$ across **all** $p, q, r$

Final circuit size is

$$\sum_{\substack{p+q+r=n, \\ p \text{ is odd}}} \frac{|\mathrm{Orb}(p, q, r)|}{|F^{-1}(1) \cap \mathrm{Orb}(p, q, r)|} \cdot \mathrm{poly}(n)$$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

**Third time's the charm Attempt**

For **all** $p, q, r$ with $p + q + r = n$ and **odd** $p$:

- Construct **one** consistent $F$ covering **large fraction** of $\mathrm{Orb}(p, q, r)$
- Define $F_i = F(\pi_i(x_1, \ldots, x_n, y_1, \ldots, y_n))$ for **random** $\pi_i \in \mathbf{Aut}_{\mathsf{IP}}$
- Each $F_i$ is **consistent**
- For $\alpha \in \mathrm{Orb}(p, q, r)$, $\Pr[\alpha \in F_i^{-1}(1)] = \frac{|F^{-1}(1) \cap \mathrm{Orb}(p,q,r)|}{|\mathrm{Orb}(p,q,r)|}$

Final circuit is **OR** of **all** $F_i$ across **all** $p, q, r$

Final circuit size is

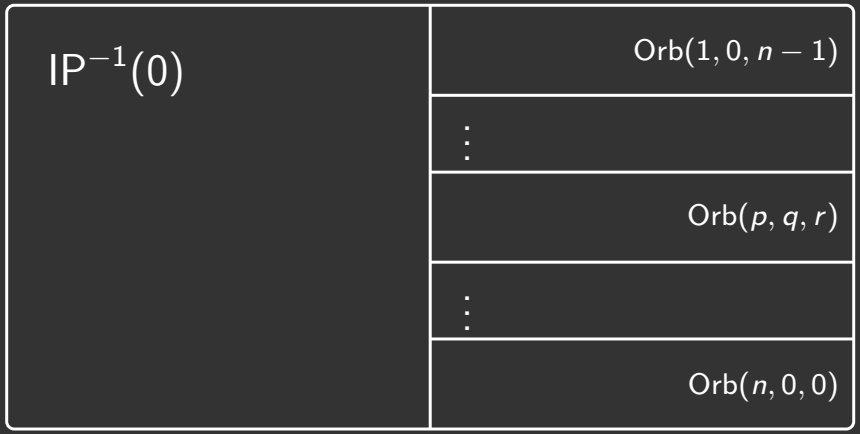
$$\max_{\substack{p+q+r=n, \\ p \text{ is odd}}} \frac{|\mathrm{Orb}(p, q, r)|}{|F^{-1}(1) \cap \mathrm{Orb}(p, q, r)|} \cdot \mathrm{poly}(n)$$

## Recipe to Consistently Cover $\mathsf{IP}^{-1}(1)$

Pictorially:

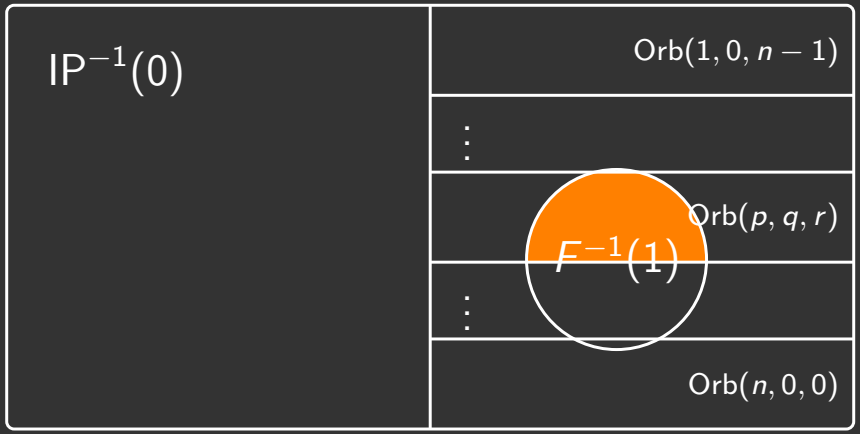
## Recipe to Consistently Cover $\mathrm{IP}^{-1}(1)$

Pictorially:



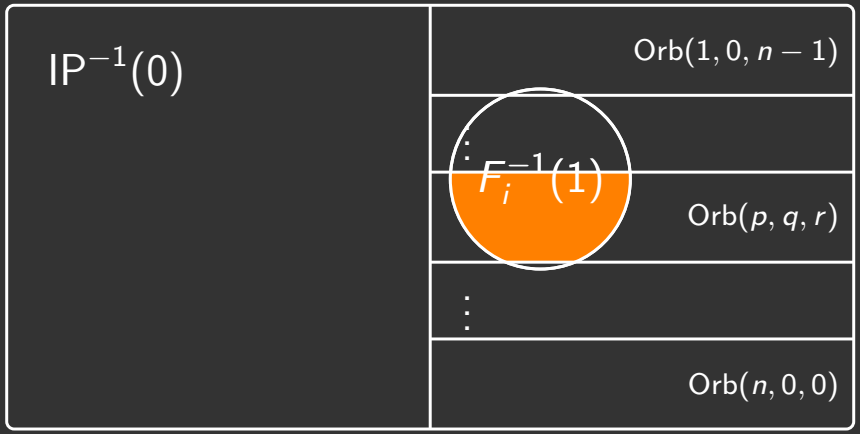
## Recipe to Consistently Cover $\mathrm{IP}^{-1}(1)$

Pictorially:



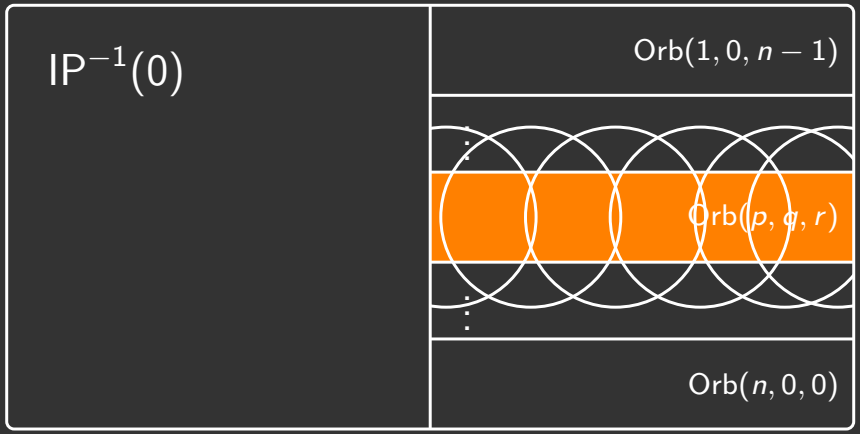
## Recipe to Consistently Cover $\mathrm{IP}^{-1}(1)$

Pictorially:



## Recipe to Consistently Cover $\mathrm{IP}^{-1}(1)$

Pictorially:



## Goal: Consistently Capturing Orbits for IP

Given $p, q, r$, construct **consistent** 2-CNF $F$ such that $|F^{-1}(1) \cap \mathbf{Orb}(p, q, r)|$ is **maximized**

## Goal: Consistently Capturing Orbits for IP

Given $p, q, r$, construct **consistent** 2-CNF $F$ such that $|F^{-1}(1) \cap \mathbf{Orb}(p, q, r)|$ is **maximized**

**Our approach**

## Goal: Consistently Capturing Orbits for IP

Given $p, q, r$, construct **consistent** 2-CNF $F$ such that $|F^{-1}(1) \cap \mathbf{Orb}(p, q, r)|$ is **maximized**

**Our approach**

- Find small number of **consistent building block** 2-CNFs $B_1, B_2, \ldots$, each over $t = O(1)$ variables

## Goal: Consistently Capturing Orbits for IP

Given $p, q, r$, construct **consistent** 2-CNF $F$ such that $|F^{-1}(1) \cap \mathbf{Orb}(p, q, r)|$ is **maximized**

**Our approach**

- Find small number of **consistent building block** 2-CNFs $B_1, B_2, \ldots$, each over $t = O(1)$ variables
- Let $F$ be **AND** of **carefully chosen** number of **disjoint copies** of $B_i$s

## Goal: Consistently Capturing Orbits for IP

Given $p, q, r$, construct **consistent** 2-CNF $F$ such that $|F^{-1}(1) \cap \mathbf{Orb}(p, q, r)|$ is **maximized**

**Our approach**

- Find small number of **consistent building block** 2-CNFs $B_1, B_2, \ldots,$ each over $t = O(1)$ variables
- Let $F$ be **AND** of **carefully chosen** number of **disjoint copies** of $B_i$s

**Same finite** set of building blocks for **all infinitely** many orbits

## Goal: Consistently Capturing Orbits for IP

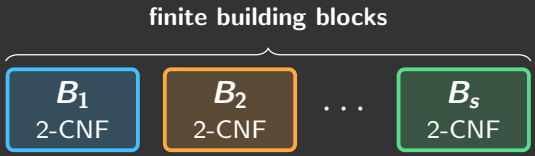
Given $p, q, r$, construct **consistent** 2-CNF $F$ such that $|F^{-1}(1) \cap \mathbf{Orb}(p, q, r)|$ is **maximized**

**Our approach**

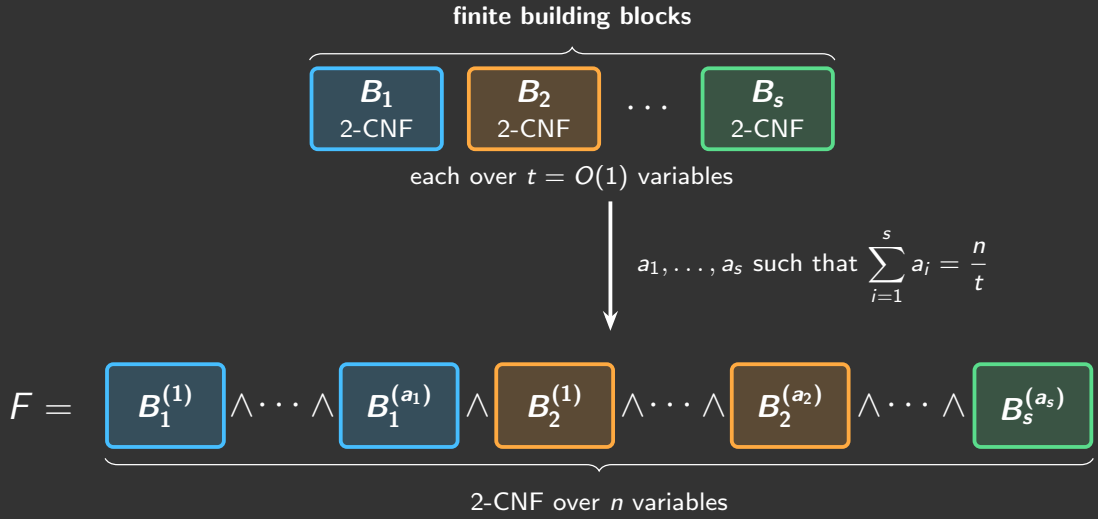
- Find small number of **consistent building block** 2-CNFs $B_1, B_2, \ldots,$ each over $t = O(1)$ variables
- Let $F$ be **AND** of **carefully chosen** number of **disjoint copies** of $B_i$s

**Same finite** set of building blocks for **all infinitely** many orbits

# Disjoint Copies of Building Blocks



# Disjoint Copies of Building Blocks



# Constructing Finite Set of Building Blocks

# Constructing Finite Set of Building Blocks

## Discovery Process

# Constructing Finite Set of Building Blocks

## Discovery Process

- **Brute force** over $t = 3, 4, 5, \ldots$ and obtain consistent building block 2-CNFs over $t$ variables

# Constructing Finite Set of Building Blocks

## Discovery Process

- **Brute force** over $t = 3, 4, 5, \ldots$ and obtain consistent building block 2-CNFs over $t$ variables
- Fix $n = 200$, and use **Dynamic Programming** to find optimal ways to **combine** building blocks

# Constructing Finite Set of Building Blocks

## Discovery Process

- **Brute force** over $t = 3, 4, 5, \ldots$ and obtain consistent building block 2-CNFs over $t$ variables
- Fix $n = 200$, and use **Dynamic Programming** to find optimal ways to **combine** building blocks
- Find **minimal** set of **optimal** building blocks

# Constructing Finite Set of Building Blocks

## Discovery Process

- **Brute force** over $t = 3, 4, 5, \ldots$ and obtain consistent building block 2-CNFs over $t$ variables
- Fix $n = 200$, and use **Dynamic Programming** to find optimal ways to **combine** building blocks
- Find **minimal** set of **optimal** building blocks
- Visit `https://constructive.codes/` by Christopher Rosin for automated LLM based discovery of combinatorial structures

# Constructing Finite Set of Building Blocks

## Discovery Process

- **Brute force** over $t = 3, 4, 5, \ldots$ and obtain consistent building block 2-CNFs over $t$ variables
- Fix $n = 200$, and use **Dynamic Programming** to find optimal ways to **combine** building blocks
- Find **minimal** set of **optimal** building blocks
- Visit `https://constructive.codes/` by Christopher Rosin for automated LLM based discovery of combinatorial structures

## Outcome

## Constructing Finite Set of Building Blocks

### Discovery Process

- **Brute force** over $t = 3, 4, 5, \ldots$ and obtain consistent building block 2-CNFs over $t$ variables
- Fix $n = 200$, and use **Dynamic Programming** to find optimal ways to **combine** building blocks
- Find **minimal** set of **optimal** building blocks
- Visit `https://constructive.codes/` by Christopher Rosin for automated LLM based discovery of combinatorial structures

### Outcome

With **3** building blocks over $t = 4$ variables, and $n = 200$, the circuit size was $(1.8)^n$

# Meet The Building Blocks

## Meet The Building Blocks

- **NAND**$(x_1, y_1)$: $\neg(x_1 \wedge y_1)$

## Meet The Building Blocks

- **NAND**$(x_1, y_1)$: $\neg(x_1 \wedge y_1)$
- **Matching**$(x_1, y_1, x_2, y_2)$: $(x_1 = x_2) \wedge (y_1 = y_2)$

## Meet The Building Blocks

- **NAND**$(x_1, y_1)$: $\neg(x_1 \wedge y_1)$
- **Matching**$(x_1, y_1, x_2, y_2)$: $(x_1 = x_2) \wedge (y_1 = y_2)$
- **2Imp**$(x_1, y_1, x_2, y_2)$: $(x_1 = x_2) \wedge (x_1 \implies y_1) \wedge (x_2 \implies y_2)$

# Combining The Building Blocks

## Combining The Building Blocks

Given $p, q, r$, find $A, B, C \in \mathbb{N}$ with $\boldsymbol{A + B + C = n/2}$ s.t. for
$F = \textbf{(Matching)}^{A}\textbf{(2Imp)}^{B}\textbf{(NAND)}^{2C}$,
$|\boldsymbol{F}^{-1}\textbf{(1)} \cap \textbf{Orb}(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r})|$ is **maximized**

## Combining The Building Blocks

Given $p, q, r$, find $A, B, C \in \mathbb{N}$ with $\boldsymbol{A + B + C = n/2}$ s.t. for
$F = \textbf{(Matching)}^{A}\textbf{(2Imp)}^{B}\textbf{(NAND)}^{2C}$,
$|\boldsymbol{F}^{-1}(1) \cap \textbf{Orb}(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r})|$ is **maximized**

**Generating Function Lens**

## Combining The Building Blocks

Given $p, q, r$, find $A, B, C \in \mathbb{N}$ with $\boldsymbol{A + B + C = n/2}$ s.t. for
$F = \textbf{(Matching)}^A\textbf{(2Imp)}^B\textbf{(NAND)}^{2C}$,
$|\boldsymbol{F^{-1}(1) \cap \mathrm{Orb}(p, q, r)}|$ is **maximized**

**Generating Function Lens**

- For 2-CNF $F$, define

$$\boldsymbol{G(x, y, z) = \sum_{p,q,r} |\mathrm{Orb}(p, q, r) \cap F^{-1}(1)| \cdot x^p y^q z^r}$$

## Combining The Building Blocks

Given $p, q, r$, find $A, B, C \in \mathbb{N}$ with $\boldsymbol{A + B + C = n/2}$ s.t. for
$F = \textbf{(Matching)}^A\textbf{(2Imp)}^B\textbf{(NAND)}^{2C}$,
$|\boldsymbol{F^{-1}(1) \cap \mathbf{Orb}(p, q, r)}|$ is **maximized**

**Generating Function Lens**

- For 2-CNF $F$, define

$$\boldsymbol{G(x, y, z) = \sum_{p,q,r} |\mathbf{Orb}(p, q, r) \cap F^{-1}(1)| \cdot x^p y^q z^r}$$

- **Crucial Observation**: If $F$ is obtained by taking **disjoint copies** of $F_1$ and $F_2$, then

$$\boldsymbol{G(x, y, z) = G_1(x, y, z) \cdot G_2(x, y, z)}$$

# Combining The Generating Functions

## Combining The Generating Functions

Given $p, q, r$, find $A, B, C \in \mathbb{N}$ with $\boldsymbol{A + B + C = n/2}$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

## Combining The Generating Functions

Given $p, q, r$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**Multivariate Analytical Combinatorics**

## Combining The Generating Functions

Given $p, q, r$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for
$G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**Multivariate Analytical Combinatorics**

- Using **standard** ideas in **analytical combinatorics**, we can **'take derivative'** and **set it to 0**

## Combining The Generating Functions

Given $p, q, r$, find $A, B, C \in \mathbb{N}$ with $\boldsymbol{A + B + C = n/2}$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**Multivariate Analytical Combinatorics**

- Using **standard** ideas in **analytical combinatorics**, we can **'take derivative'** and **set it to 0**
- Doing so, we obtain

$$A = \frac{5}{4}n - \frac{25}{32}p - \frac{25}{4}r$$
$$B = \frac{-5}{4}n + \frac{25}{16}p + \frac{25}{4}r$$
$$C = \frac{1}{2}n - \frac{25}{32}p$$

## Combining The Generating Functions

Given $p, q, r$, find $A, B, C \in \mathbb{N}$ with $\boldsymbol{A + B + C = n/2}$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**Multivariate Analytical Combinatorics**

- Using **standard** ideas in **analytical combinatorics**, we can **'take derivative'** and **set it to 0**
- Doing so, we obtain

$$A = \frac{5}{4}n - \frac{25}{32}p - \frac{25}{4}r$$
$$B = \frac{-5}{4}n + \frac{25}{16}p + \frac{25}{4}r$$
$$C = \frac{1}{2}n - \frac{25}{32}p$$

- And that circuit size is $\max_{p,q,r} \frac{|\text{Orb}(p,q,r)|}{\text{coefficient of } x^p y^q z^r} \leq (1.8)^n$

# However...

## However...

$$A = \frac{5}{4}n - \frac{25}{32}p - \frac{25}{4}r$$
$$B = \frac{-5}{4}n + \frac{25}{16}p + \frac{25}{4}r$$
$$C = \frac{1}{2}n - \frac{25}{32}p$$

## However...

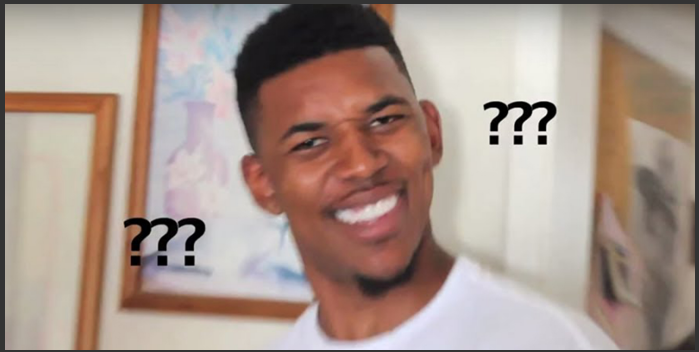
$$A = \frac{5}{4}n - \frac{25}{32}p - \frac{25}{4}r$$
$$B = \frac{-5}{4}n + \frac{25}{16}p + \frac{25}{4}r$$
$$C = \frac{1}{2}n - \frac{25}{32}p$$

$A, B, C$ can be **negative**

## However...

$$A = \frac{5}{4}n - \frac{25}{32}p - \frac{25}{4}r$$
$$B = \frac{-5}{4}n + \frac{25}{16}p + \frac{25}{4}r$$
$$C = \frac{1}{2}n - \frac{25}{32}p$$

$A, B, C$ can be **negative**



## Regions of Orbits

Let $\mathcal{R}_i(n) \subseteq \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ be $(p, q, r) \in \mathbb{N}$ such that $p + q + r = n$ and

## Regions of Orbits

Let $\mathcal{R}_i(n) \subseteq \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ be $(p, q, r) \in \mathbb{N}$ such that $p + q + r = n$ and

$$\mathcal{R}_1(n) : \frac{1}{2}n - \frac{25}{32}p \geq 0, \frac{5}{4}n - \frac{25}{32}p - \frac{25}{4}r \geq 0, -\frac{5}{4}n + \frac{25}{16}p + \frac{25}{4}r \geq 0$$

$$\mathcal{R}_2(n) : \frac{1}{2}n - \frac{25}{32}p \leq 0$$

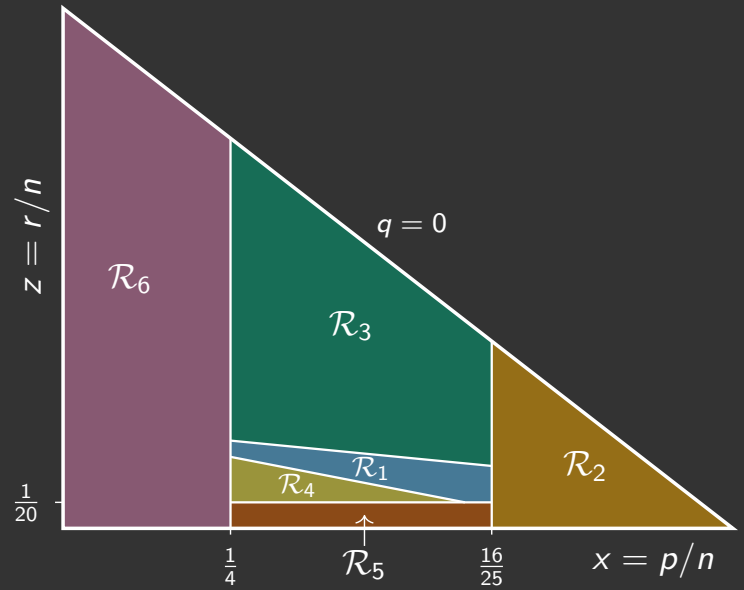
$$\mathcal{R}_3(n) : \frac{1}{2}n - \frac{25}{32}p \geq 0, \frac{5}{4}n - \frac{25}{32}p - \frac{25}{4}r \leq 0, \frac{1}{4}n - p \leq 0$$

$$\mathcal{R}_4(n) : \frac{1}{2}n - \frac{25}{32}p \geq 0, -\frac{5}{4}n + \frac{25}{16}p + \frac{25}{4}r \leq 0, -\frac{1}{20}n + r \geq 0, \frac{1}{4}n - p \leq 0$$

$$\mathcal{R}_5(n) : \frac{1}{2}n - \frac{25}{32}p \geq 0, -\frac{1}{20}n + r \leq 0$$

$$\mathcal{R}_6(n) : \frac{1}{4}n - p \geq 0$$

## Six regions




$$x = \frac{p}{n},$$
$$z = \frac{r}{n},$$
$$\frac{q}{n} = 1 - x - z$$

# Combining Building Blocks for Orbits in Region 1

## Combining Building Blocks for Orbits in Region 1

Given $p, q, r \in \mathcal{R}_1(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

## Combining Building Blocks for Orbits in Region 1

Given $p, q, r \in \mathcal{R}_1(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

$$\mathcal{R}_1(n) : \frac{1}{2}n - \frac{25}{32}p \geq 0, \frac{5}{4}n - \frac{25}{32}p - \frac{25}{4}r \geq 0, -\frac{5}{4}n + \frac{25}{16}p + \frac{25}{4}r \geq 0$$

## Combining Building Blocks for Orbits in Region 1

Given $p, q, r \in \mathcal{R}_1(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

$$\mathcal{R}_1(n) : \frac{1}{2}n - \frac{25}{32}p \geq 0, \frac{5}{4}n - \frac{25}{32}p - \frac{25}{4}r \geq 0, -\frac{5}{4}n + \frac{25}{16}p + \frac{25}{4}r \geq 0$$

Solved earlier using **Analytical Combinatorics** :)

## Combining Building Blocks for Orbits in Region 2/5/6

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

# Combining Building Blocks for Orbits in Region 2/5/6

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**Simple** constructions and **straightforward** calculation

## Combining Building Blocks for Orbits in Region 3/4

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

# Combining Building Blocks for Orbits in Region 3/4

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**For region 3:**

## Combining Building Blocks for Orbits in Region 3/4

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**For region 3:**

- Inspired by $n = 200$, set $B_1 = 0.34n, C_1 = 0.16n, B_2 = 0.465n, C_2 = 0.035n$

# Combining Building Blocks for Orbits in Region 3/4

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**For region 3:**

- Inspired by $n = 200$, set $B_1 = 0.34n, C_1 = 0.16n, B_2 = 0.465n, C_2 = 0.035n$
- Let $G_1 = (G_{\text{2Imp}})^{B_1} (G_{\text{NAND}})^{2C_1}$
  Let $G_2 = (G_{\text{2Imp}})^{B_2} (G_{\text{NAND}})^{2C_2}$

# Combining Building Blocks for Orbits in Region 3/4

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**For region 3:**

- Inspired by $n = 200$, set $B_1 = 0.34n, C_1 = 0.16n, B_2 = 0.465n, C_2 = 0.035n$
- Let $G_1 = (G_{\text{2Imp}})^{B_1} (G_{\text{NAND}})^{2C_1}$
  Let $G_2 = (G_{\text{2Imp}})^{B_2} (G_{\text{NAND}})^{2C_2}$
- Given $p, q, r \in \mathcal{R}_3(n)$, pick $i \in [2]$ s.t. $G_i$ has larger **coefficient** of $x^p y^q z^r$

# Combining Building Blocks for Orbits in Region 3/4

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**For region 3:**

- Inspired by $n = 200$, set $B_1 = 0.34n, C_1 = 0.16n, B_2 = 0.465n, C_2 = 0.035n$
- Let $G_1 = (G_{\text{2Imp}})^{B_1} (G_{\text{NAND}})^{2C_1}$
  Let $G_2 = (G_{\text{2Imp}})^{B_2} (G_{\text{NAND}})^{2C_2}$
- Given $p, q, r \in \mathcal{R}_3(n)$, pick $i \in [2]$ s.t. $G_i$ has larger **coefficient** of $x^p y^q z^r$
- Want **strong lower bound** on
  $\min_{p,q,r \in \mathcal{R}_3(n)} \max_{i \in [2]} \frac{\text{coefficient of } x^p y^q z^r \text{ in } (G_{\text{2Imp}})^{B_i} (G_{\text{NAND}})^{2C_i}}{|\text{Orb}(p,q,r)|}$

# Combining Building Blocks for Orbits in Region 3/4

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**For region 3:**

- Inspired by $n = 200$, set $B_1 = 0.34n, C_1 = 0.16n, B_2 = 0.465n, C_2 = 0.035n$
- Let $G_1 = (G_{\text{2Imp}})^{B_1} (G_{\text{NAND}})^{2C_1}$
  Let $G_2 = (G_{\text{2Imp}})^{B_2} (G_{\text{NAND}})^{2C_2}$
- Given $p, q, r \in \mathcal{R}_3(n)$, pick $i \in [2]$ s.t. $G_i$ has larger **coefficient** of $x^p y^q z^r$
- Want **strong lower bound** on
  $\min_{p,q,r \in \mathcal{R}_3(n)} \max_{i \in [2]} \frac{\text{coefficient of } x^p y^q z^r \text{ in } (G_{\text{2Imp}})^{B_i} (G_{\text{NAND}})^{2C_i}}{|\text{Orb}(p,q,r)|}$
- Directly analyzing — highly unclear and complicated

# Combining Building Blocks for Orbits in Region 3/4

Given $p, q, r \in \mathcal{R}_i(n)$, find $A, B, C \in \mathbb{N}$ with $A + B + C = n/2$ s.t. for $G = (G_{\text{Matching}})^A (G_{\text{2Imp}})^B (G_{\text{NAND}})^{2C}$,
the **coefficient** of $x^p y^q z^r$ in $G$ is **maximized**

**For region 3:**

- Inspired by $n = 200$, set $B_1 = 0.34n, C_1 = 0.16n, B_2 = 0.465n, C_2 = 0.035n$
- Let $G_1 = (G_{\text{2Imp}})^{B_1} (G_{\text{NAND}})^{2C_1}$
  Let $G_2 = (G_{\text{2Imp}})^{B_2} (G_{\text{NAND}})^{2C_2}$
- Given $p, q, r \in \mathcal{R}_3(n)$, pick $i \in [2]$ s.t. $G_i$ has larger **coefficient** of $x^p y^q z^r$
- Want **strong lower bound** on
  $\min_{p,q,r \in \mathcal{R}_3(n)} \max_{i \in [2]} \frac{\text{coefficient of } x^p y^q z^r \text{ in } (G_{\text{2Imp}})^{B_i} (G_{\text{NAND}})^{2C_i}}{|\text{Orb}(p,q,r)|}$
- Directly analyzing — highly unclear and complicated
- Used IbexOpt, **optimization tool** part of interval arithmetic library IBEX in C++

# Conclusion

# To Summarize

## To Summarize

- Wanted to construct **optimal** $\Sigma\Pi\Sigma_2$ circuits for IP

## To Summarize

- Wanted to construct **optimal** $\Sigma\Pi\Sigma_2$ circuits for IP
- Reduced to constructing **consistent** 2-CNFs for **orbits** of IP

## To Summarize

- Wanted to construct **optimal** $\Sigma\Pi\Sigma_2$ circuits for IP
- Reduced to constructing **consistent** 2-CNFs for **orbits** of IP
- Used **computer search** to find building blocks for this task

## To Summarize

- Wanted to construct **optimal** $\Sigma\Pi\Sigma_2$ circuits for IP
- Reduced to constructing **consistent** 2-CNFs for **orbits** of IP
- Used **computer search** to find building blocks for this task
- Used **dynamic programming** to find minimal set of building blocks that can attain $\mathbf{(1.8)^n}$ bound

## To Summarize

- Wanted to construct **optimal** $\Sigma\Pi\Sigma_2$ circuits for IP
- Reduced to constructing **consistent** 2-CNFs for **orbits** of IP
- Used **computer search** to find building blocks for this task
- Used **dynamic programming** to find minimal set of building blocks that can attain $\mathbf{(1.8)^n}$ bound
- Used **generating functions**, **analytical combinatorics**, **direct analysis**, and **Ibex optimization solver** to optimally **combine** building blocks


$\Sigma\Pi\Sigma_2$ Circuit for IP
Brute Force
Dynamic Programming
Generating Functions
Analytical Combinatorics
Direct Analysis
Ibex Optimization Solver

# Open Problems

# Open Problems

## Complexity of IP

- Figure out $\Sigma\Pi\Sigma_k$ complexity of IP
- Open even for $k = 3$ - complexity between $2^{0.33n}$ and $2^{0.692n}$

# Open Problems

## Complexity of IP

- Figure out $\Sigma\Pi\Sigma_k$ complexity of IP
- Open even for $k = 3$ - complexity between $2^{0.33n}$ and $2^{0.692n}$

## Degree 2 polynomial with large $\Sigma\Pi\Sigma_k$ complexity

- Construct **explicit** degree **2** polynomial with $\Sigma\Pi\Sigma_k$ complexity $2^{n-o(n)}$
- Open even for $k = 2$
- **Existentially**, shown by Impagliazzo-Paturi-Zane'01

# Thank you