Simple Circuit Extensions for XOR in PTIME

Marco Carmosino

IBM

Ngu (Nathan) Dang

Boston University

Tim Jackman

Boston University

Outline

Table of Contents

Circuits

- Circuits are a model for studying the complexity of Boolean functions

Circuits

- Circuits are a model for studying the complexity of Boolean functions
- DAGs where sources are variables and interior nodes are Boolean operators

Circuits

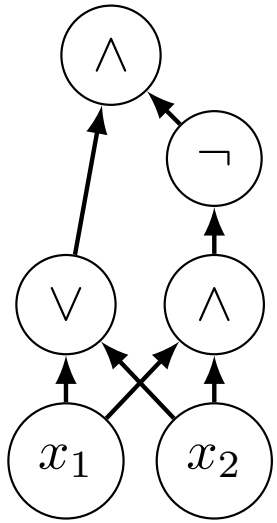
- Circuits are a model for studying the complexity of Boolean functions
- DAGs where sources are variables and interior nodes are Boolean operators
  - DeMorgan Basis: fan-in 2 $\wedge$ and $\vee$, fan-in 1 $\neg$

Circuits

- Circuits are a model for studying the complexity of Boolean functions
- DAGs where sources are variables and interior nodes are Boolean operators
  - DeMorgan Basis: fan-in 2 $\wedge$ and $\vee$, fan-in 1 $\neg$
  - No restriction on fan-out

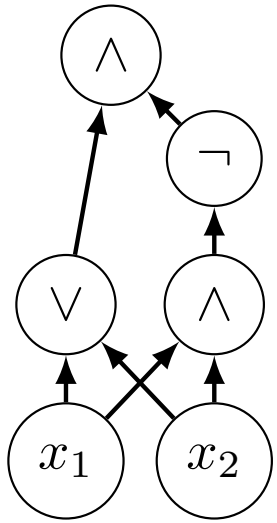
Circuits

- Circuits are a model for studying the complexity of Boolean functions
- DAGs where sources are variables and interior nodes are Boolean operators
  - DeMorgan Basis: fan-in 2 $\wedge$ and $\vee$, fan-in 1 $\neg$
  - No restriction on fan-out



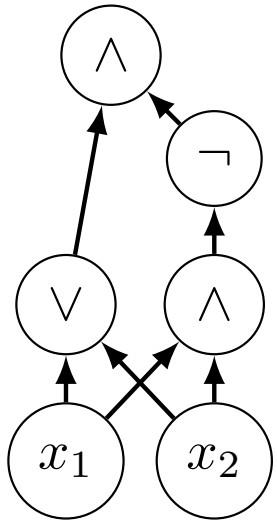
Circuits

- Circuits are a model for studying the complexity of Boolean functions
- DAGs where sources are variables and interior nodes are Boolean operators
  - DeMorgan Basis: fan-in 2 $\wedge$ and $\vee$, fan-in 1 $\neg$
  - No restriction on fan-out
- Size := number of *binary* gates



Circuits

- Circuits are a model for studying the complexity of Boolean functions
- DAGs where sources are variables and interior nodes are Boolean operators
  - DeMorgan Basis: fan-in 2 $\wedge$ and $\vee$, fan-in 1 $\neg$
  - No restriction on fan-out
- Size := number of *binary* gates
- $CC(f)$:= size of the smallest circuit computing f



The Minimum Circuit Size Problem

The Minimum Circuit Size Problem (MCSP)

Given a truth table of $f : \{0,1\}^n \to \{0,1\}$ (a string of length 2^n) and size parameter $s \in \mathbb{N}$

Output yes if $CC(f) \leq s$

The Minimum Circuit Size Problem

The Minimum Circuit Size Problem (MCSP)

Given a truth table of $f : \{0,1\}^n \to \{0,1\}$ (a string of length 2^n) and size parameter $s \in \mathbb{N}$
Output yes if $CC(f) \leq s$

- In NP, unknown whether it's NP-complete

The Minimum Circuit Size Problem

The Minimum Circuit Size Problem (MCSP)

Given a truth table of $f : \{0,1\}^n \to \{0,1\}$ (a string of length 2^n) and size parameter $s \in \mathbb{N}$
Output yes if $CC(f) \leq s$

- In NP, unknown whether it's NP-complete
- Resolving this has been a **major open question** for ≥ 75 years [Trakhtenbrot 1984]

The Minimum Circuit Size Problem

The Minimum Circuit Size Problem (MCSP)

Given a truth table of $f : \{0,1\}^n \to \{0,1\}$ (a string of length 2^n) and size parameter $s \in \mathbb{N}$
Output yes if $CC(f) \leq s$

- In NP, unknown whether it's NP-complete
- Resolving this has been a **major open question** for ≥ 75 years [Trakhtenbrot 1984]
- If in P, then one-way functions do not exist [Kabanets, Cai 2000]

The Minimum Circuit Size Problem

The Minimum Circuit Size Problem (MCSP)

Given a truth table of $f : \{0,1\}^n \to \{0,1\}$ (a string of length 2^n) and size parameter $s \in \mathbb{N}$
Output yes if $CC(f) \leq s$

- In NP, unknown whether it's NP-complete
- Resolving this has been a **major open question** for ≥ 75 years [Trakhtenbrot 1984]
- If in P, then one-way functions do not exist [Kabanets, Cai 2000]
- If it's NP-complete, then EXP $\neq$ ZPP [Murray, Williams 2015]

The Minimum Circuit Size Problem

The Minimum Circuit Size Problem (MCSP)

Given a truth table of $f : \{0,1\}^n \to \{0,1\}$ (a string of length 2^n) and size parameter $s \in \mathbb{N}$
Output yes if $CC(f) \leq s$

- In NP, unknown whether it's NP-complete
- Resolving this has been a **major open question** for ≥ 75 years [Trakhtenbrot 1984]
- If in P, then one-way functions do not exist [Kabanets, Cai 2000]
- If it's NP-complete, then EXP $\neq$ ZPP [Murray, Williams 2015]
- Some variants are known to be hard:

The Minimum Circuit Size Problem

The Minimum Circuit Size Problem (MCSP)

Given a truth table of $f : \{0,1\}^n \to \{0,1\}$ (a string of length 2^n) and size parameter $s \in \mathbb{N}$
Output yes if $CC(f) \leq s$

- In NP, unknown whether it's NP-complete
- Resolving this has been a **major open question** for ≥ 75 years [Trakhtenbrot 1984]
- If in P, then one-way functions do not exist [Kabanets, Cai 2000]
- If it's NP-complete, then EXP $\neq$ ZPP [Murray, Williams 2015]
- Some variants are known to be hard:
  - DNF-MCSP [Masek 1979]
  - OR-AND-MOD-MCSP [Hirahara, Oliveira, Santhanam 2018]
  - Multi-output MCSP [Ilango, Loff, Oliviera 2020]

MCSP*

The Partial Function Minimum Circuit Size Problem (MCSP*)

Given a partial truth table of $f : \{0,1,\star\}^n \to \{0,1\}$ (string of length 2^n) and a size parameter $s \in \mathbb{N}$

Output yes if there exists a completion h such that $CC(h) \leq s$?

MCSP*

The Partial Function Minimum Circuit Size Problem (MCSP*)

Given a partial truth table of $f : \{0, 1, \star\}^n \to \{0, 1\}$ (string of length 2^n) and a size parameter $s \in \mathbb{N}$

Output yes if there exists a completion h such that $CC(h) \leq s$?

Theorem (Ilango 2020)

MCSP* $\notin$ P *under the Exponential Time Hypothesis.*

MCSP*

The Partial Function Minimum Circuit Size Problem (MCSP*)

Given a partial truth table of $f : \{0, 1, \star\}^n \to \{0, 1\}$ (string of length 2^n) and a size parameter $s \in \mathbb{N}$

Output yes if there exists a completion h such that $CC(h) \leq s$?

Theorem (Ilango 2020)

MCSP* $\notin$ P *under the Exponential Time Hypothesis.*

Proof.

(1) OR-SEP* is ETH-hard and (2) OR-SEP* $\leq$ MCSP* $\square$

MCSP*

The Partial Function Minimum Circuit Size Problem (MCSP*)

Given a partial truth table of $f : \{0,1,\star\}^n \to \{0,1\}$ (string of length 2^n) and a size parameter $s \in \mathbb{N}$

Output yes if there exists a completion h such that $CC(h) \leq s$?

Theorem (Ilango 2020)

MCSP* $\notin$ P *under the Exponential Time Hypothesis.*

Proof.

(1) OR-SEP* is ETH-hard and (2) OR-SEP* $\leq$ MCSP* $\square$

- Can we extend the proof idea to MCSP?

MCSP*

The Partial Function Minimum Circuit Size Problem (MCSP*)

Given a partial truth table of $f : \{0,1,\star\}^n \to \{0,1\}$ (string of length 2^n) and a size parameter $s \in \mathbb{N}$

Output yes if there exists a completion h such that $CC(h) \leq s$?

Theorem (Ilango 2020)

MCSP* $\notin$ P *under the Exponential Time Hypothesis.*

Proof.

(1) OR-SEP* is ETH-hard and (2) OR-SEP* $\leq$ MCSP* $\square$

- Can we extend the proof idea to MCSP?
- MCSP* is now known to be NP-hard [Hirahara 2022]

MCSP*

The Partial Function Minimum Circuit Size Problem (MCSP*)

Given a partial truth table of $f : \{0, 1, \star\}^n \to \{0, 1\}$ (string of length 2^n) and a size parameter $s \in \mathbb{N}$

Output yes if there exists a completion h such that $CC(h) \leq s$?

Theorem (Ilango 2020)

MCSP* $\notin$ P *under the Exponential Time Hypothesis.*

Proof.

(1) OR-SEP* is ETH-hard and (2) OR-SEP* $\leq$ MCSP* $\square$

- Can we extend the proof idea to MCSP?
- MCSP* is now known to be NP-hard [Hirahara 2022]
  - Barriers to proving MCSP without added assumptions

$\mathcal{F}$-Simple Extensions

- A function $g : \{0,1\}^{n+m} \to \{0,1\}$ is a *simple extension* of $f : \{0,1\}^n \to \{0,1\}$ if:

$\mathcal{F}$-Simple Extensions

- A function $g : \{0,1\}^{n+m} \to \{0,1\}$ is a *simple extension* of $f : \{0,1\}^n \to \{0,1\}$ if:
  - g *extends* f*:* $\exists$ restriction $r \in \{0,1\}^m$ such that $g\restriction_r \equiv f$ (*key* substitution)

$\mathcal{F}$-Simple Extensions

- A function $g : \{0,1\}^{n+m} \to \{0,1\}$ is a *simple extension* of $f : \{0,1\}^n \to \{0,1\}$ if:
  - g *extends* f*:* $\exists$ restriction $r \in \{0,1\}^m$ such that $g\restriction_r \equiv f$ (*key* substitution)
  - g is *non-degenerate:* it depends on all its inputs

$\mathcal{F}$-Simple Extensions

- A function $g : \{0,1\}^{n+m} \to \{0,1\}$ is a *simple extension* of $f : \{0,1\}^n \to \{0,1\}$ if:
  - g *extends* f*:* $\exists$ restriction $r \in \{0,1\}^m$ such that $g\restriction_r \equiv f$ (*key* substitution)
  - g is *non-degenerate:* it depends on all its inputs
  - $CC(g) = CC(f) + m$

$\mathcal{F}$-Simple Extensions

- A function $g : \{0,1\}^{n+m} \to \{0,1\}$ is a *simple extension* of $f : \{0,1\}^n \to \{0,1\}$ if:
  - g *extends* f*:* $\exists$ restriction $r \in \{0,1\}^m$ such that $g\restriction_r \equiv f$ (*key* substitution)
  - g is *non-degenerate:* it depends on all its inputs
  - $CC(g) = CC(f) + m$
- For any non-degenerate extensions g, $CC(g) \geq CC(f) + m$

$\mathcal{F}$-Simple Extensions

- A function $g : \{0,1\}^{n+m} \to \{0,1\}$ is a *simple extension* of $f : \{0,1\}^n \to \{0,1\}$ if:
  - g *extends* f*:* $\exists$ restriction $r \in \{0,1\}^m$ such that $g\restriction_r \equiv f$ (*key* substitution)
  - g is *non-degenerate:* it depends on all its inputs
  - $CC(g) = CC(f) + m$
- For any non-degenerate extensions g, $CC(g) \geq CC(f) + m$
- For a family $\mathcal{F} = \{f_i\}_{i \in \mathbb{N}}$, define

$\mathcal{F}$-Simple Extensions

- A function $g : \{0,1\}^{n+m} \to \{0,1\}$ is a *simple extension* of $f : \{0,1\}^n \to \{0,1\}$ if:
  - g *extends* f*:* $\exists$ restriction $r \in \{0,1\}^m$ such that $g\restriction_r \equiv f$ (*key* substitution)
  - g is *non-degenerate:* it depends on all its inputs
  - $CC(g) = CC(f) + m$
- For any non-degenerate extensions g, $CC(g) \geq CC(f) + m$
- For a family $\mathcal{F} = \{f_i\}_{i \in \mathbb{N}}$, define

The $\mathcal{F}$-Simple Extension Problem ($\mathcal{F}$-SEP)

Given truth table of $g : \{0,1\}^{n+m} \to \{0,1\}$ and $n \in \mathbb{N}$
Output yes if g is a simple extension of f_n

$\mathcal{F}$-Simple Extensions

- A function $g : \{0,1\}^{n+m} \to \{0,1\}$ is a *simple extension* of $f : \{0,1\}^n \to \{0,1\}$ if:
  - g *extends* f*:* $\exists$ restriction $r \in \{0,1\}^m$ such that $g\restriction_r \equiv f$ (*key* substitution)
  - g is *non-degenerate:* it depends on all its inputs
  - $CC(g) = CC(f) + m$
- For any non-degenerate extensions g, $CC(g) \geq CC(f) + m$
- For a family $\mathcal{F} = \{f_i\}_{i \in \mathbb{N}}$, define

The $\mathcal{F}$-Simple Extension Problem ($\mathcal{F}$-SEP)

Given truth table of $g : \{0,1\}^{n+m} \to \{0,1\}$ and $n \in \mathbb{N}$
Output yes if g is a simple extension of f_n

- Testing if g is a non-degenerate extension is *easy*

$\mathcal{F}$-Simple Extensions

- A function $g : \{0,1\}^{n+m} \to \{0,1\}$ is a *simple extension* of $f : \{0,1\}^n \to \{0,1\}$ if:
  - g *extends* f*:* $\exists$ restriction $r \in \{0,1\}^m$ such that $g\restriction_r \equiv f$ (*key* substitution)
  - g is *non-degenerate:* it depends on all its inputs
  - $CC(g) = CC(f) + m$
- For any non-degenerate extensions g, $CC(g) \geq CC(f) + m$
- For a family $\mathcal{F} = \{f_i\}_{i \in \mathbb{N}}$, define

The $\mathcal{F}$-Simple Extension Problem ($\mathcal{F}$-SEP)

Given truth table of $g : \{0,1\}^{n+m} \to \{0,1\}$ and $n \in \mathbb{N}$
Output yes if g is a simple extension of f_n

- Testing if g is a non-degenerate extension is *easy*
- Reduces to a call to MCSP oracle

From Partial Functions to Total Functions

- When $\mathcal{F} = \text{OR}$ and g is a partial function, this problem is ETH-hard

From Partial Functions to Total Functions

- When $\mathcal{F} = \text{OR}$ and g is a partial function, this problem is ETH-hard
- Can we extend this to show hardness for MCSP?

From Partial Functions to Total Functions

- When $\mathcal{F} = \text{OR}$ and g is a partial function, this problem is ETH-hard
- Can we extend this to show hardness for MCSP?
- No. OR-SEP is *easy* when g is a total function.

From Partial Functions to Total Functions

- When $\mathcal{F} = \text{OR}$ and g is a partial function, this problem is ETH-hard
- Can we extend this to show hardness for MCSP?
- No. OR-SEP is *easy* when g is a total function.
  - Simple Extensions of Read-once Formulas are Read-once Formulas

From Partial Functions to Total Functions

- When $\mathcal{F} = \text{OR}$ and g is a partial function, this problem is ETH-hard
- Can we extend this to show hardness for MCSP?
- No. OR-SEP is *easy* when g is a total function.
  - Simple Extensions of Read-once Formulas are Read-once Formulas
  - "Is g computed by read-once formulas?" is in P [Golumbic, Mintz, Rotics 2006]

From Partial Functions to Total Functions

- When $\mathcal{F} = \text{OR}$ and g is a partial function, this problem is ETH-hard
- Can we extend this to show hardness for MCSP?
- No. OR-SEP is *easy* when g is a total function.
  - Simple Extensions of Read-once Formulas are Read-once Formulas
  - "Is g computed by read-once formulas?" is in P [Golumbic, Mintz, Rotics 2006]

From Partial Functions to Total Functions

- When $\mathcal{F} = \text{OR}$ and g is a partial function, this problem is ETH-hard
- Can we extend this to show hardness for MCSP?
- No. OR-SEP is *easy* when g is a total function.
  - Simple Extensions of Read-once Formulas are Read-once Formulas
  - "Is g computed by read-once formulas?" is in P [Golumbic, Mintz, Rotics 2006]

"Thus, the missing component in extending our results to MCSP *is finding some function f whose optimal circuits we can characterize but are also sufficiently complex."*

- Rahul Ilango, SIAM J. of Computing, 2022

From Partial Functions to Total Functions

- When $\mathcal{F} = \text{OR}$ and g is a partial function, this problem is ETH-hard
- Can we extend this to show hardness for MCSP?
- No. OR-SEP is *easy* when g is a total function.
  - Simple Extensions of Read-once Formulas are Read-once Formulas
  - "Is g computed by read-once formulas?" is in P [Golumbic, Mintz, Rotics 2006]

"Thus, the missing component in extending our results to MCSP *is finding some function f whose optimal circuits we can characterize but are also sufficiently complex."*

- Rahul Ilango, SIAM J. of Computing, 2022

- Can we use XOR?

Table of Contents

XOR-Simple Extension

Theorem (FPT Algorithm for $\mathcal{F}$-SEP)

$\mathcal{F}$-SEP $\in P$ whenever

1. $CC(\mathcal{F}) = O(n)$,
2. *the maximum fanout in any optimal circuit computing $\mathcal{F}$ is constant,*
3. *and the number of optimal circuits computing $\mathcal{F}$ is $2^{O(n)}$ (up to isomorphism and permutation of inputs)*

XOR-Simple Extension

Theorem (FPT Algorithm for $\mathcal{F}$-SEP)

$\mathcal{F}$-SEP $\in P$ *whenever*

1. $CC(\mathcal{F}) = O(n)$,
2. *the maximum fanout in any optimal circuit computing* $\mathcal{F}$ *is constant,*
3. *and the number of optimal circuits computing* $\mathcal{F}$ *is* $2^{O(n)}$ *(up to isomorphism and permutation of inputs)*

Lemma (XOR Structure Lemma)

In our setting, all optimal XOR_n *circuits are binary trees of* $n-1$ $(\neg)\mathrm{XOR}_2$ *widgets*

XOR-Simple Extension

Theorem (FPT Algorithm for $\mathcal{F}$-SEP)

$\mathcal{F}$-SEP $\in P$ *whenever*

1. $CC(\mathcal{F}) = O(n)$,
2. *the maximum fanout in any optimal circuit computing* $\mathcal{F}$ *is constant,*
3. *and the number of optimal circuits computing* $\mathcal{F}$ *is* $2^{O(n)}$ *(up to isomorphism and permutation of inputs)*

Lemma (XOR Structure Lemma)

In our setting, all optimal XOR_n *circuits are binary trees of* $n-1$ $(\neg)\mathrm{XOR}_2$ *widgets*

Corollary (Main Corollary)

XOR-SEP $\in$ P.

Interpreting Our Results

- FPT algorithm gives us a more concrete metric for "sufficiently complex":

Interpreting Our Results

- FPT algorithm gives us a more concrete metric for "sufficiently complex":
  1. Non-linear circuit size,

Interpreting Our Results

- FPT algorithm gives us a more concrete metric for "sufficiently complex":
  1. Non-linear circuit size,
  2. Non-constant fanout, or

Interpreting Our Results

- FPT algorithm gives us a more concrete metric for "sufficiently complex":
  1. Non-linear circuit size,
  2. Non-constant fanout, or
  3. Many diverse constructions

Interpreting Our Results

- FPT algorithm gives us a more concrete metric for "sufficiently complex":
  1. Non-linear circuit size,
  2. Non-constant fanout, or
  3. Many diverse constructions
- #2 seems the most achievable with current techniques

Interpreting Our Results

- FPT algorithm gives us a more concrete metric for "sufficiently complex":
  1. Non-linear circuit size,
  2. Non-constant fanout, or
  3. Many diverse constructions
- #2 seems the most achievable with current techniques
- Analyzing *structure* will help us apply/bypass our algorithm

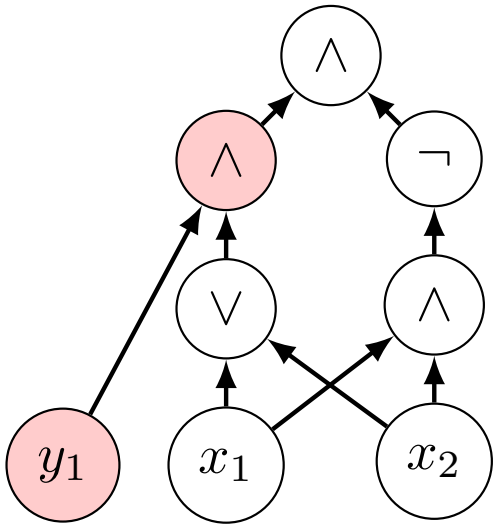
Table of Contents

Structure of Simple Extension Circuits

- Let's look at an example of a circuit for a XOR_2 simple extension

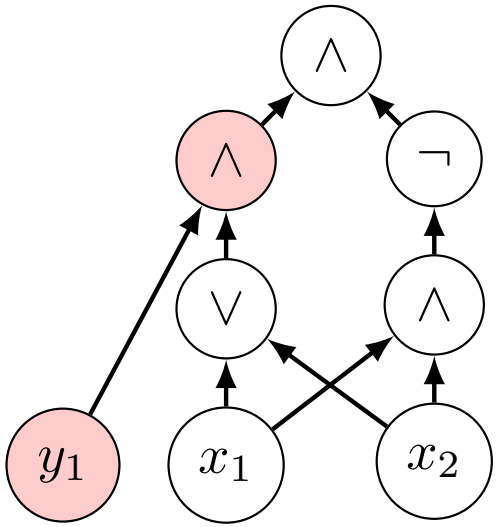
Structure of Simple Extension Circuits

- Let's look at an example of a circuit for a XOR_2 simple extension



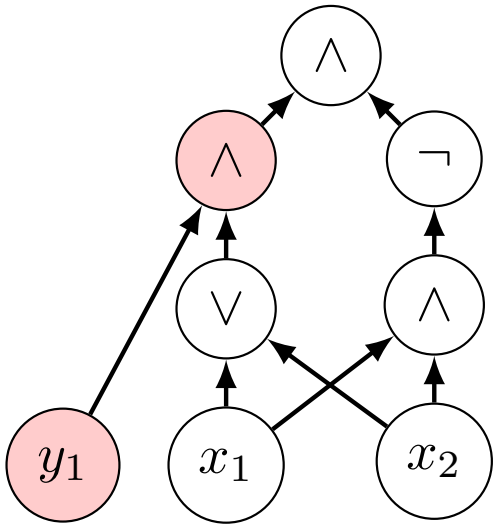
Structure of Simple Extension Circuits

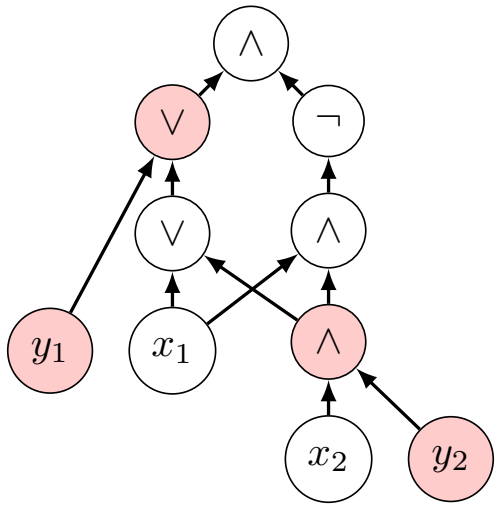
- Let's look at an example of a circuit for a XOR_2 simple extension
- When we substitute the *key*, we can simplify the optimal Simple Extension circuit and get an optimal base circuit

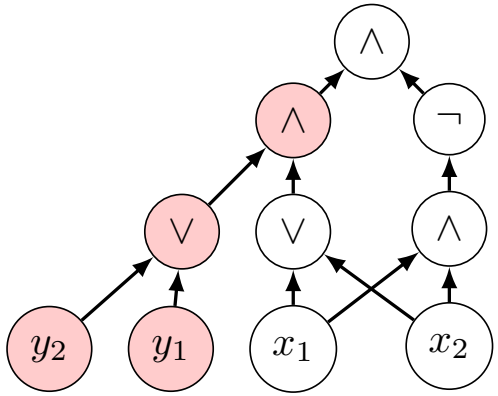


Structure of Simple Extension Circuits

- Let's look at an example of a circuit for a XOR_2 simple extension
- When we substitute the *key*, we can simplify the optimal Simple Extension circuit and get an optimal base circuit
- Extension variables can be "spliced" on new or original nodes

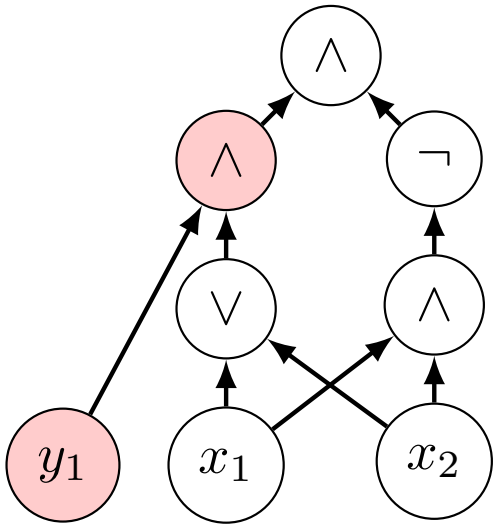


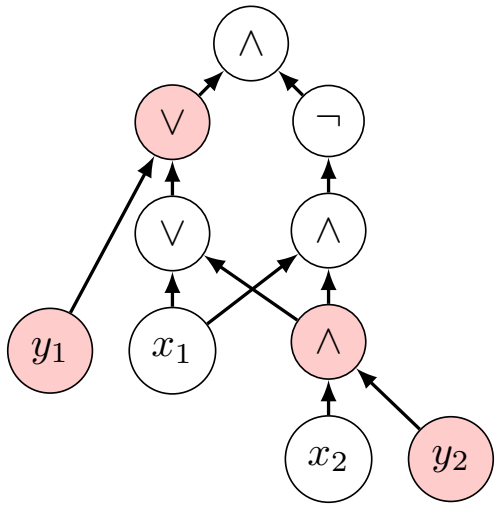


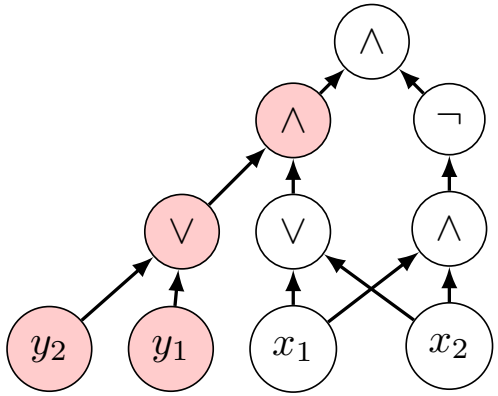


Structure of Simple Extension Circuits

- Let's look at an example of a circuit for a XOR_2 simple extension
- When we substitute the *key*, we can simplify the optimal Simple Extension circuit and get an optimal base circuit
- Extension variables can be "spliced" on new or original nodes
  - Idea: One binary gate per variable $\rightarrow$ added circuitry isn't complicated







Structure of Simple Extension Circuits

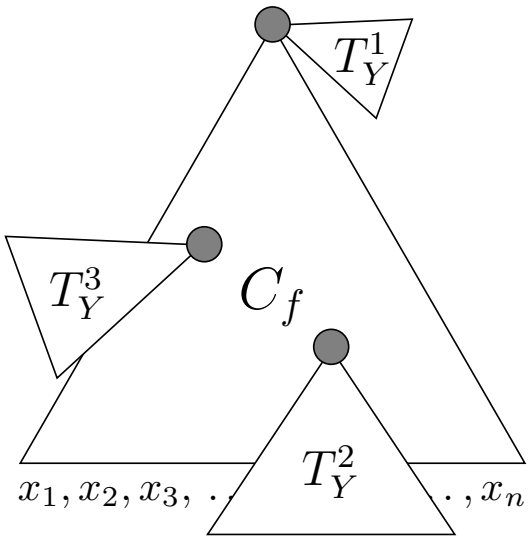
Theorem (Y-Tree Decomposition)

Any optimal circuit for g, a simple extension of f, can be decomposed into an optimal circuit computing f decorated with read-once subformulas that depend only on disjoint sets of extension variables (Y-trees).

Structure of Simple Extension Circuits

Theorem (Y-Tree Decomposition)

Any optimal circuit for g, a simple extension of f, can be decomposed into an optimal circuit computing f decorated with read-once subformulas that depend only on disjoint sets of extension variables (Y-trees).

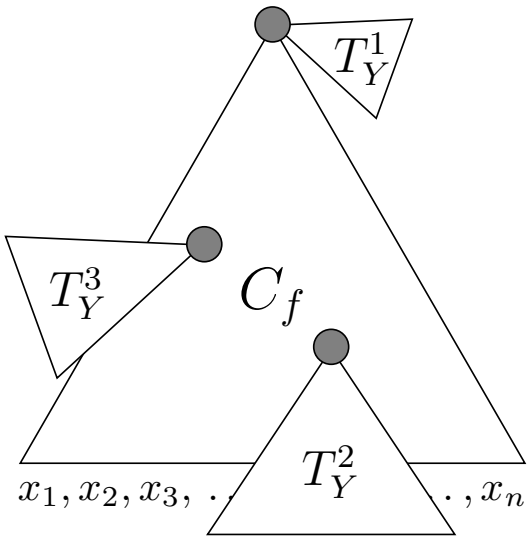


Structure of Simple Extension Circuits

Theorem (Y-Tree Decomposition)

Any optimal circuit for g, a simple extension of f, can be decomposed into an optimal circuit computing f decorated with read-once subformulas that depend only on disjoint sets of extension variables (Y-trees).

- # of options for where to add Y-trees depends on $CC(f)$ and the fanout of each gate

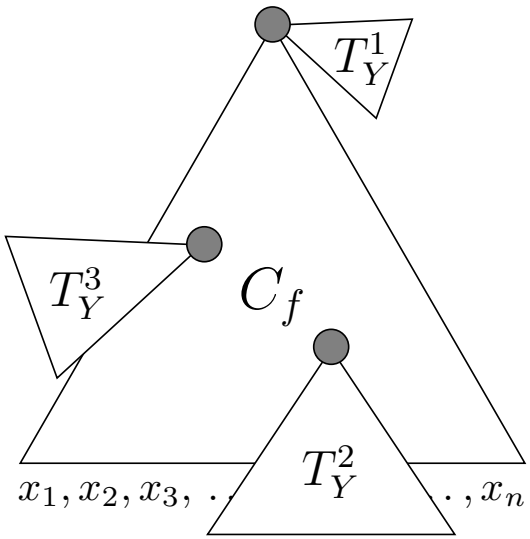


Structure of Simple Extension Circuits

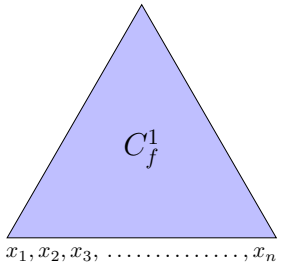
Theorem (Y-Tree Decomposition)

Any optimal circuit for g, a simple extension of f, can be decomposed into an optimal circuit computing f decorated with read-once subformulas that depend only on disjoint sets of extension variables (Y-trees).

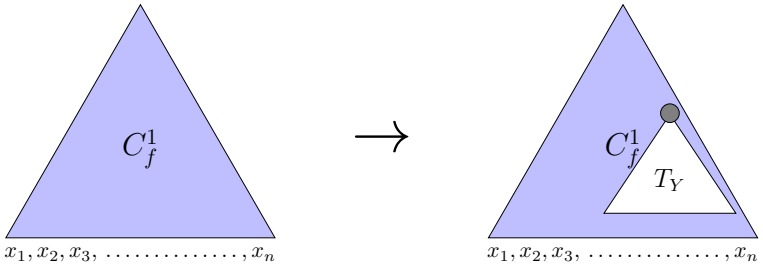
- # of options for where to add Y-trees depends on $CC(f)$ and the fanout of each gate
- If these are limited, then we could try to brute force over all possible ways to add the subformulas



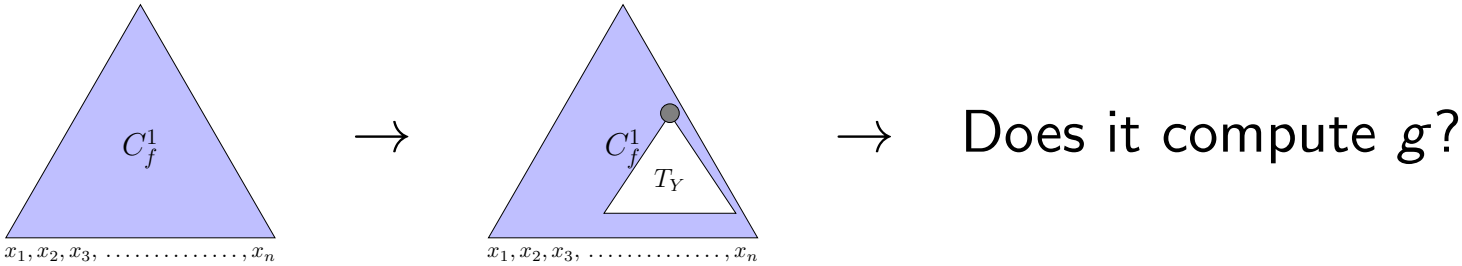
Better Brute Force



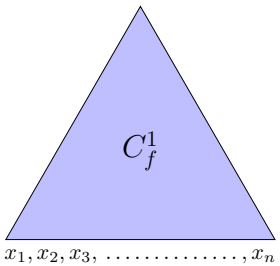
Better Brute Force



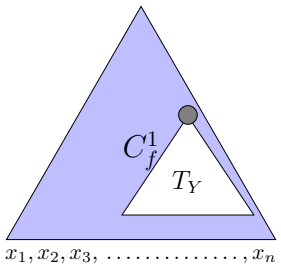
Better Brute Force



Better Brute Force

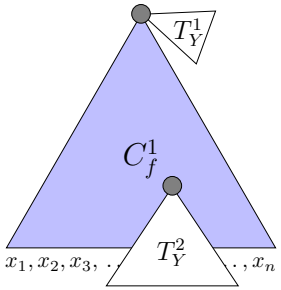


$\rightarrow$



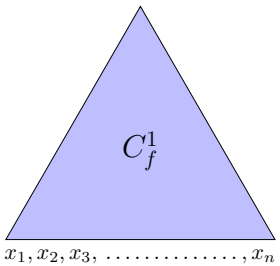
$\rightarrow$ Does it compute g?

$\rightarrow$

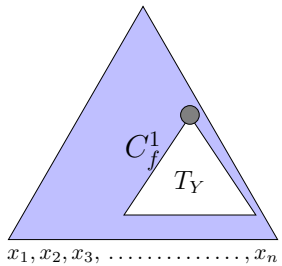


$\rightarrow$ Does it compute g?

Better Brute Force

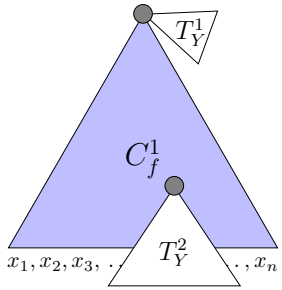


$\rightarrow$



$\rightarrow$ Does it compute g?

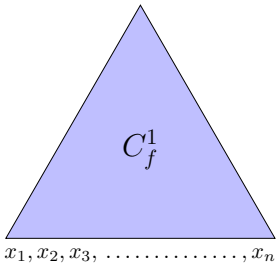
$\rightarrow$



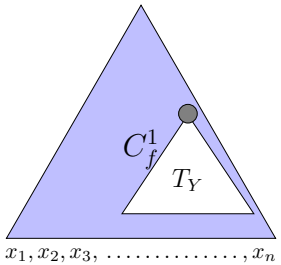
$\rightarrow$ Does it compute g?

$\vdots$

Better Brute Force

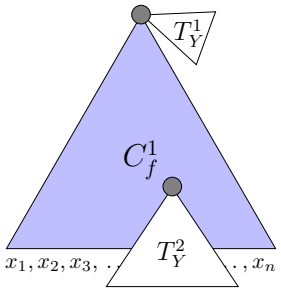


$\rightarrow$



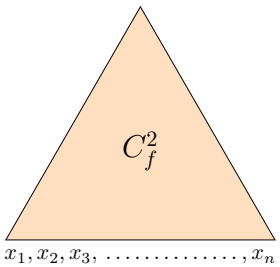
$\rightarrow$ Does it compute g?

$\rightarrow$

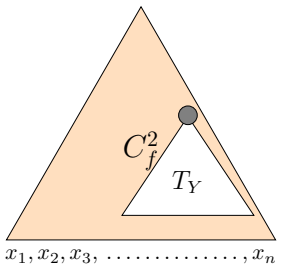


$\rightarrow$ Does it compute g?

$\vdots$

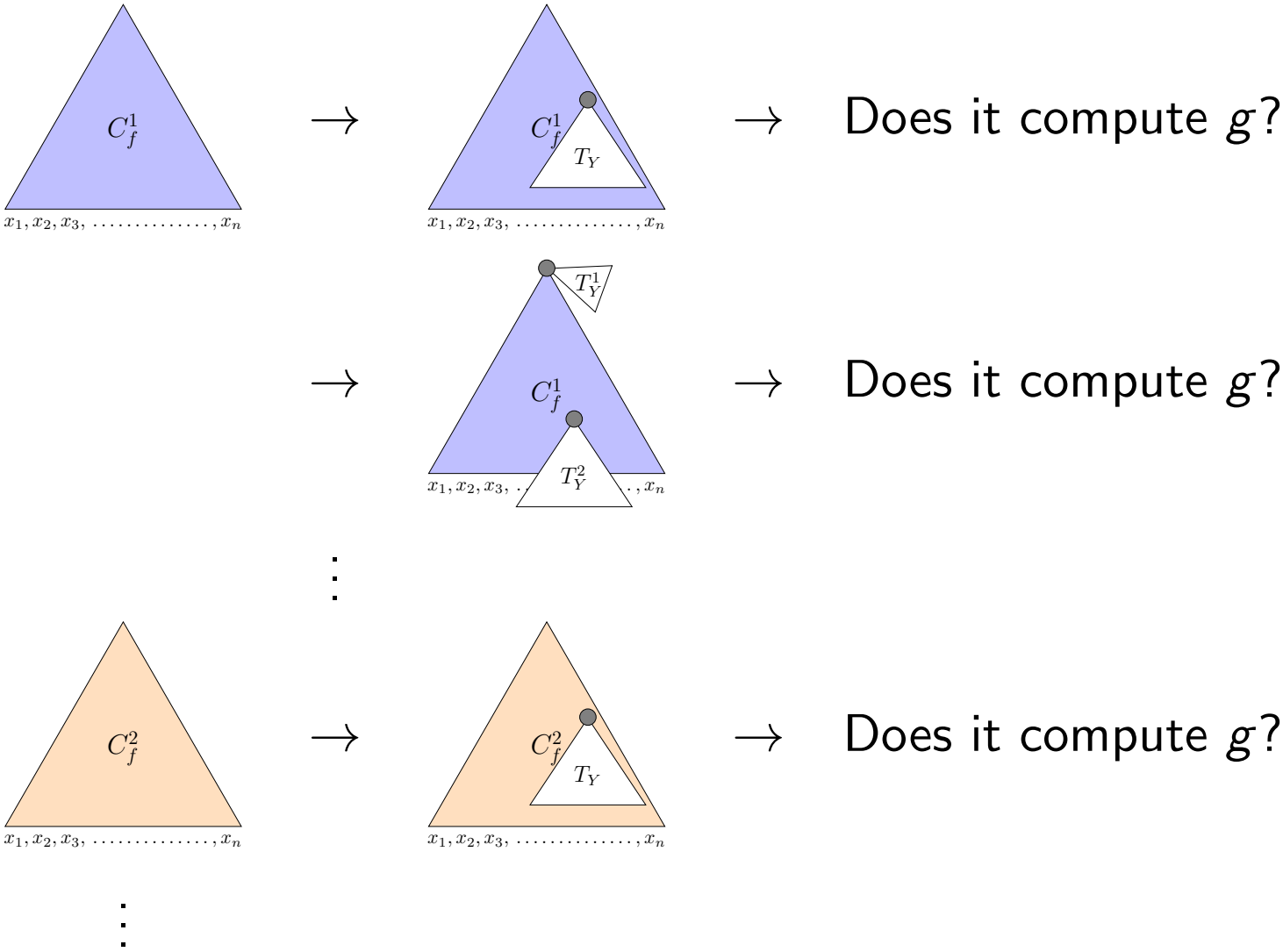


$\rightarrow$



$\rightarrow$ Does it compute g?

Better Brute Force

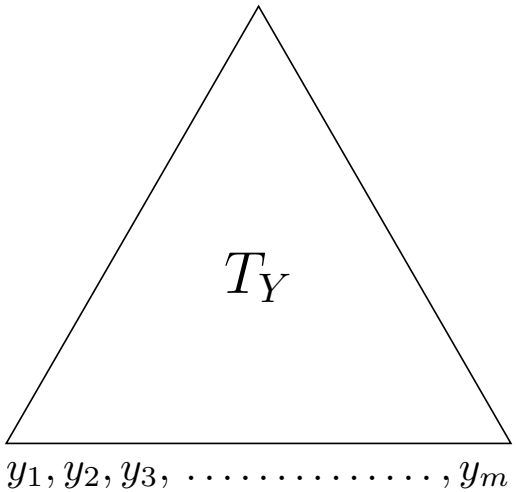


Truth-Table Isomorphism

- Immediate Issue: $m!$ ways to permute the extension variables in any "potential" extension

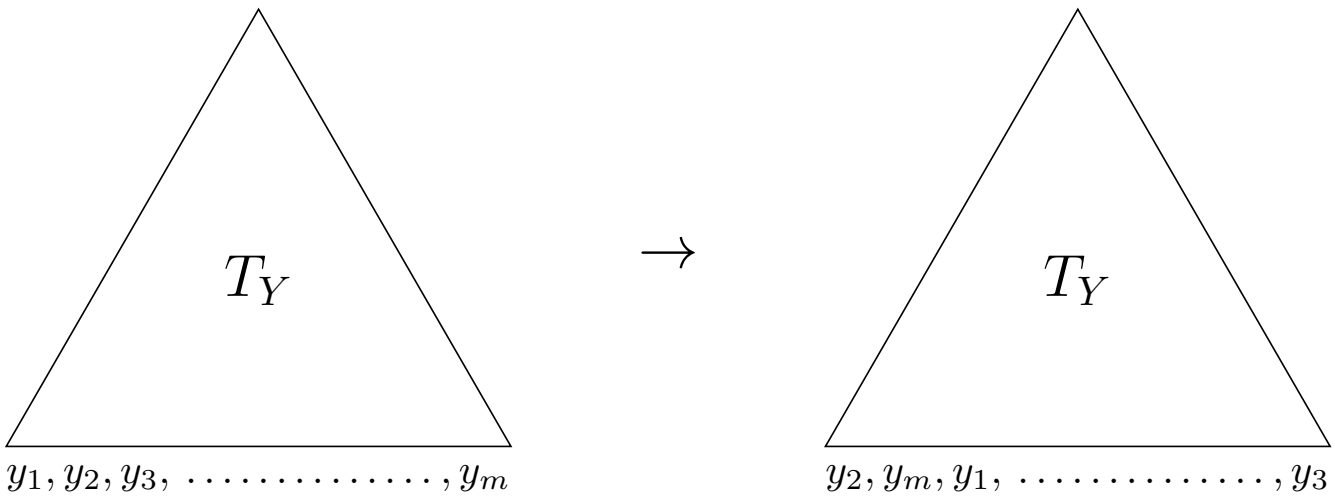
Truth-Table Isomorphism

- Immediate Issue: $m!$ ways to permute the extension variables in any "potential" extension



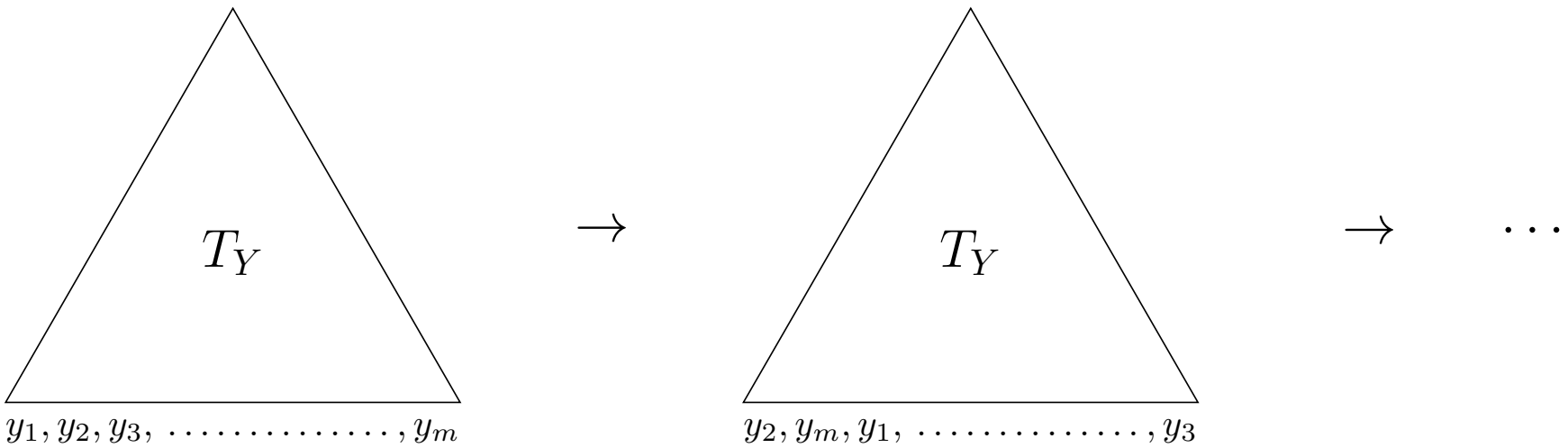
Truth-Table Isomorphism

- Immediate Issue: $m!$ ways to permute the extension variables in any "potential" extension



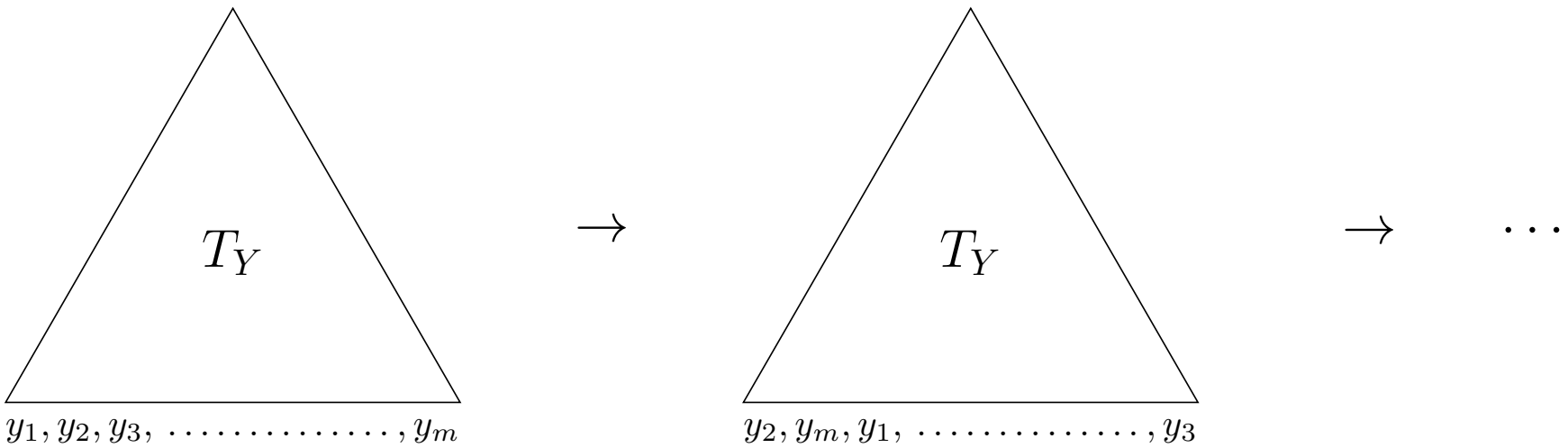
Truth-Table Isomorphism

- Immediate Issue: $m!$ ways to permute the extension variables in any "potential" extension



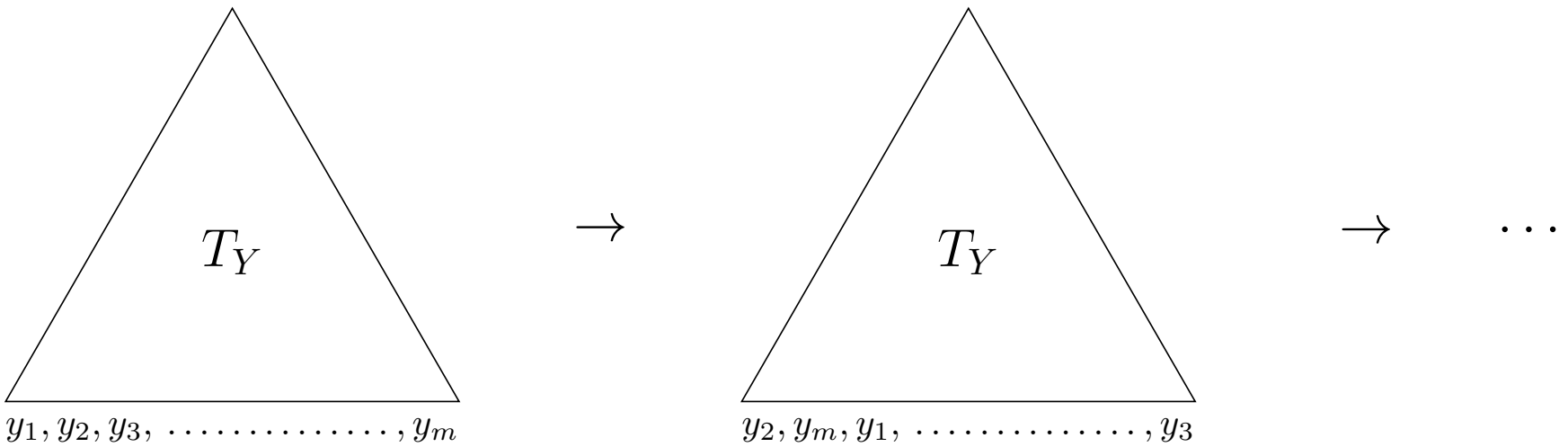
Truth-Table Isomorphism

- Immediate Issue: $m!$ ways to permute the extension variables in any "potential" extension
- Also occurs with # of base circuits for symmetric functions



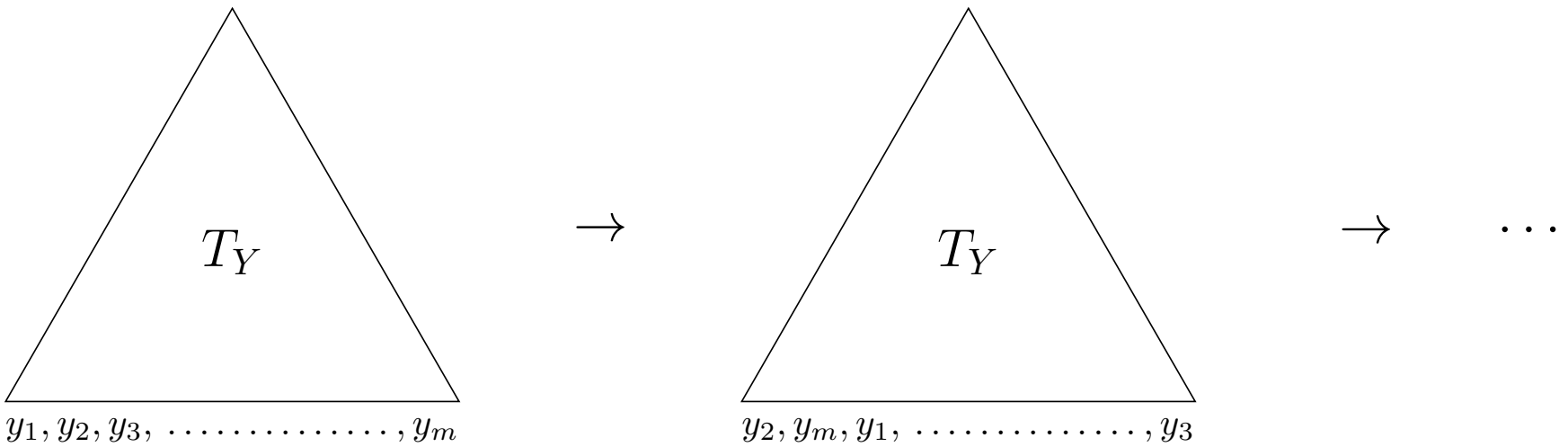
Truth-Table Isomorphism

- Immediate Issue: $m!$ ways to permute the extension variables in any "potential" extension
- Also occurs with # of base circuits for symmetric functions
- However, if two circuits have the same structure but permuted variables, they compute *truth-table isomorphic* functions



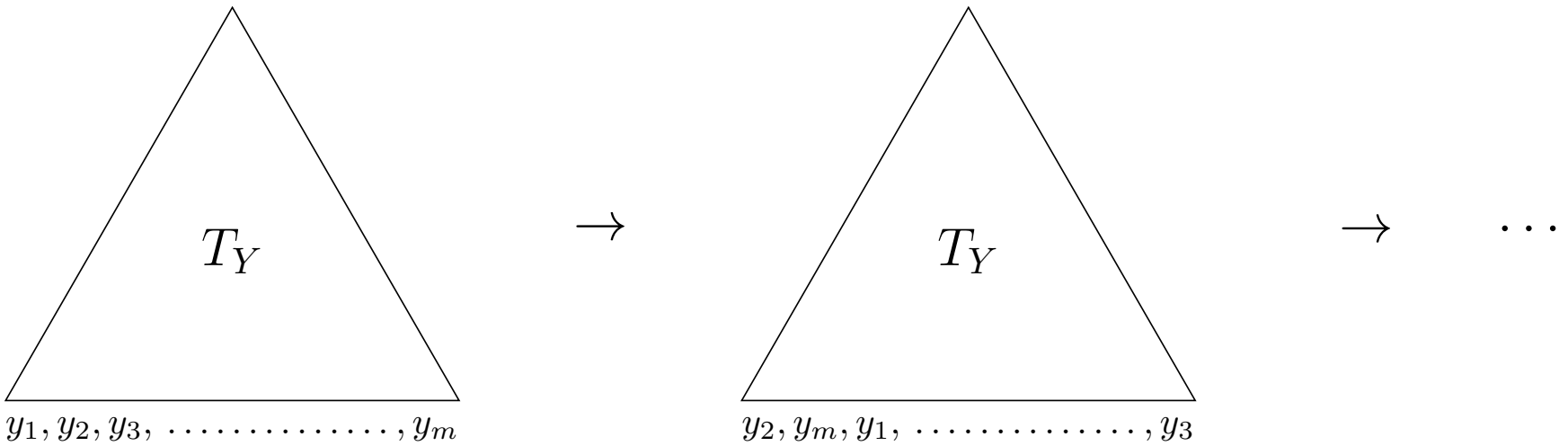
Truth-Table Isomorphism

- Immediate Issue: $m!$ ways to permute the extension variables in any "potential" extension
- Also occurs with # of base circuits for symmetric functions
- However, if two circuits have the same structure but permuted variables, they compute *truth-table isomorphic* functions
- Testing if two functions are truth-table isomorphic is in P [Luks 1999]

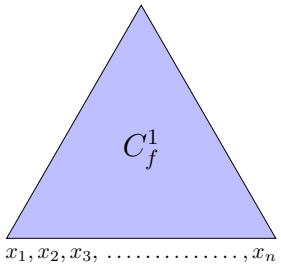


Truth-Table Isomorphism

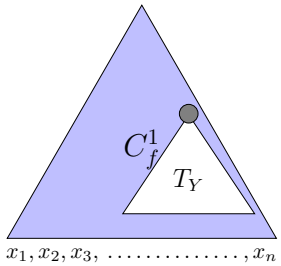
- Immediate Issue: $m!$ ways to permute the extension variables in any "potential" extension
- Also occurs with # of base circuits for symmetric functions
- However, if two circuits have the same structure but permuted variables, they compute *truth-table isomorphic* functions
- Testing if two functions are truth-table isomorphic is in P [Luks 1999]
- Only need to brute force up to permutations



Better Brute Force

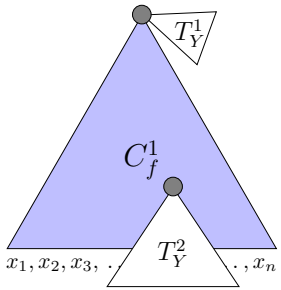


$\rightarrow$



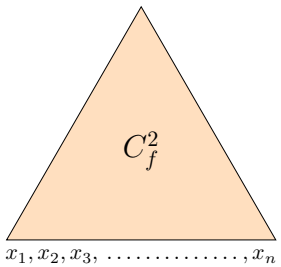
$\rightarrow$ Is it tt-isomorphic to g?

$\rightarrow$

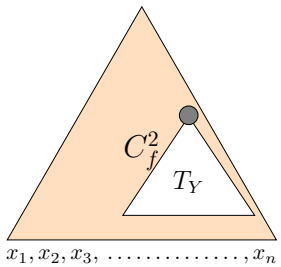


$\rightarrow$ Is it tt-isomorphic to g?

$\vdots$



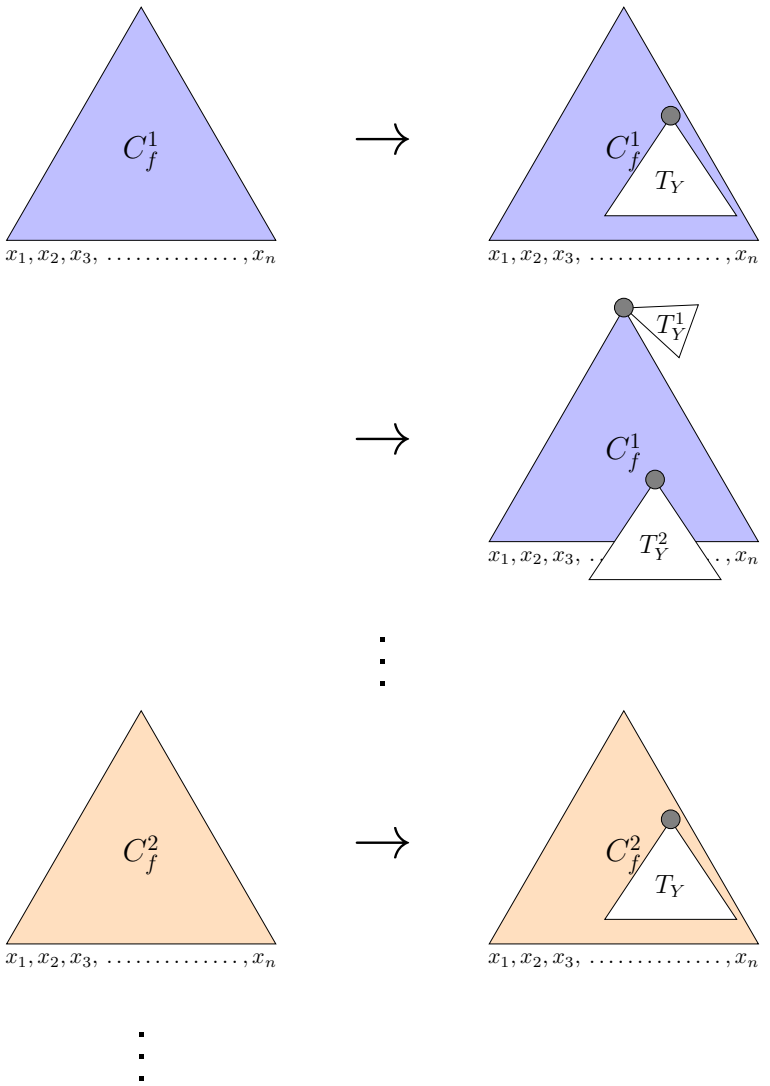
$\rightarrow$



$\rightarrow$ Is it tt-isomorphic to g?

$\vdots$

Better Brute Force

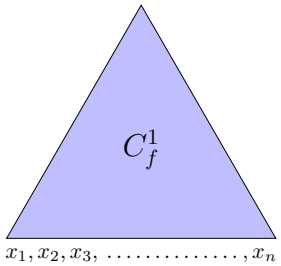


Inner Loops Feasible By:

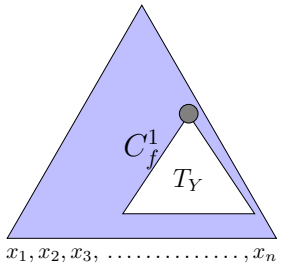
- Y-Tree Decomposition
- Truth-Table Isomorphism
- $|C_f|$ is linear
- Low Fan-Out

→ Is it tt-isomorphic to g?

Better Brute Force

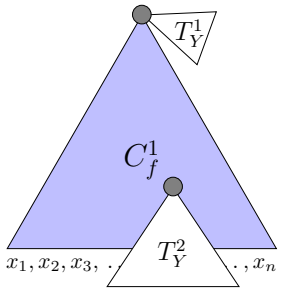


$\rightarrow$



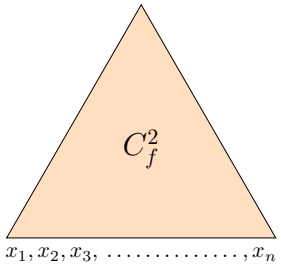
$\rightarrow$ Is it tt-isomorphic to g?

$\rightarrow$

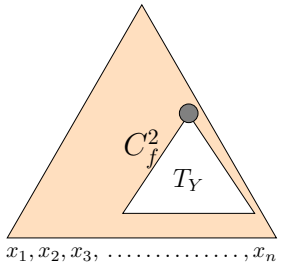


$\rightarrow$ Is it tt-isomorphic to g?

$\vdots$



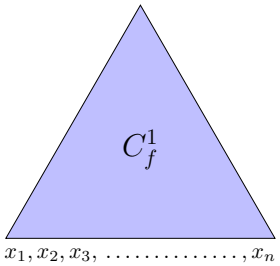
$\rightarrow$

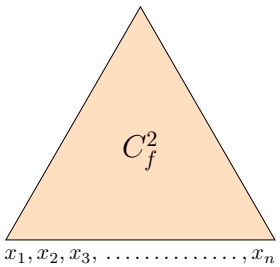


$\rightarrow$ Is it tt-isomorphic to g?

$\vdots$

Better Brute Force





Outer Loop Feasible By:

- Truth-Table Isomorphism
- Few Base Circuits

Characterizing XOR

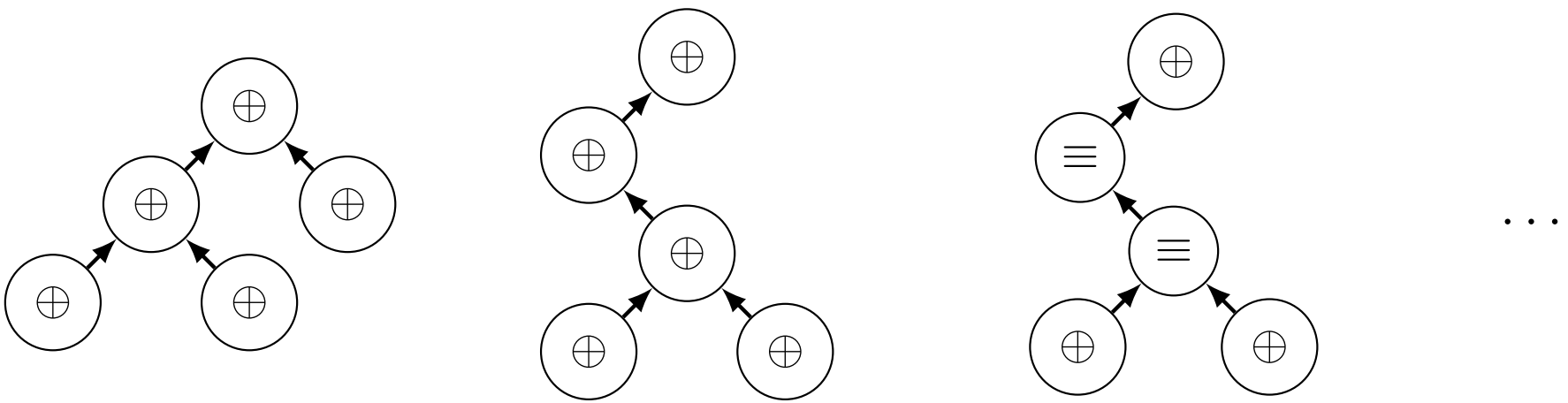
- When $\neg$ *count* towards size, structure of XOR_n is known

Characterizing XOR

- When $\neg$ *count* towards size, structure of XOR_n is known
  - Optimal XOR_n ckts are binary trees of XOR_2 widgets [Kombarov 2011]

Characterizing XOR

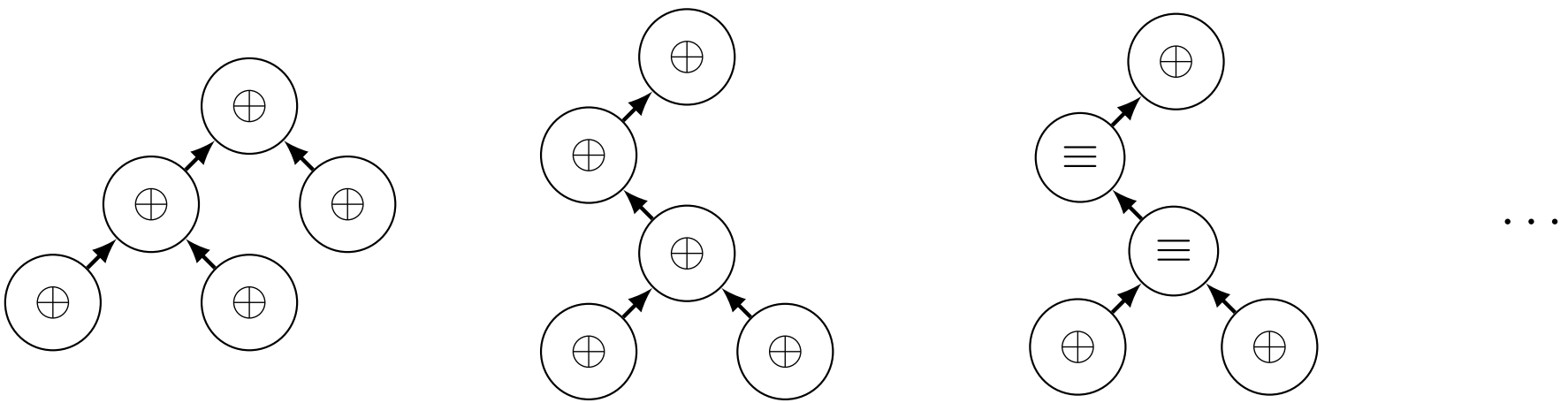
- When $\neg$ *count* towards size, structure of XOR_n is known
  - Optimal XOR_n ckts are binary trees of XOR_2 widgets [Kombarov 2011]
  - Max fanout is 2 and the # of optimal circuits is $2^{O(n)}$ as required



XOR_6 **circuits are binary trees of 5** $\mathrm{XOR}_2(\oplus)$ **or** $\neg\mathrm{XOR}_2(\equiv)$ **widgets**

Characterizing XOR

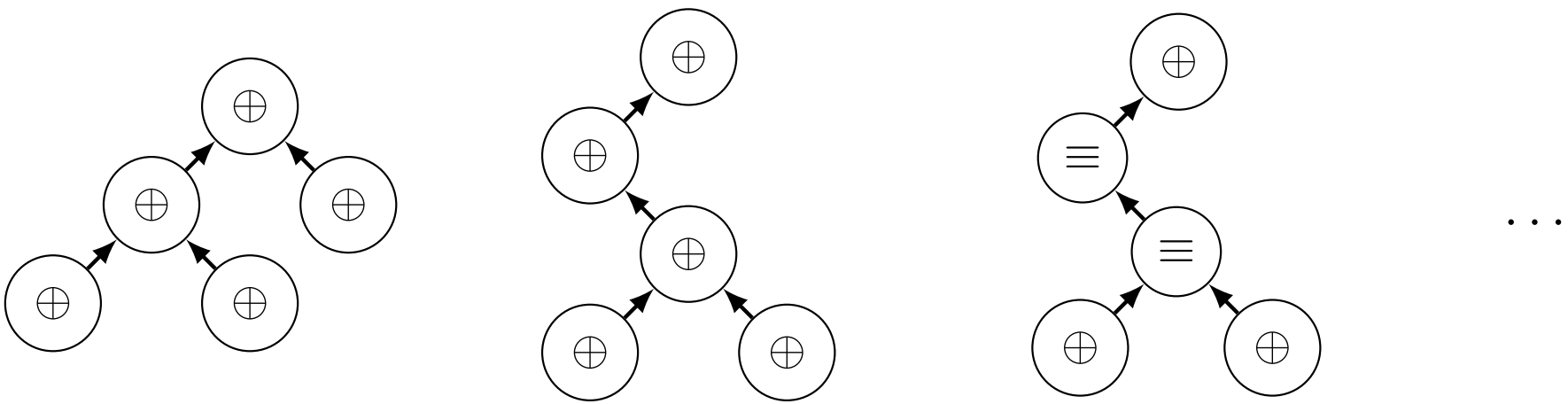
- When $\neg$ *count* towards size, structure of XOR_n is known
  - Optimal XOR_n ckts are binary trees of XOR_2 widgets [Kombarov 2011]
  - Max fanout is 2 and the # of optimal circuits is $2^{O(n)}$ as required
- We extend this characterization when $\neg$ gates are "free"



XOR_6 **circuits are binary trees of 5** $\mathrm{XOR}_2(\oplus)$ **or** $\neg\mathrm{XOR}_2(\equiv)$ **widgets**

Characterizing XOR

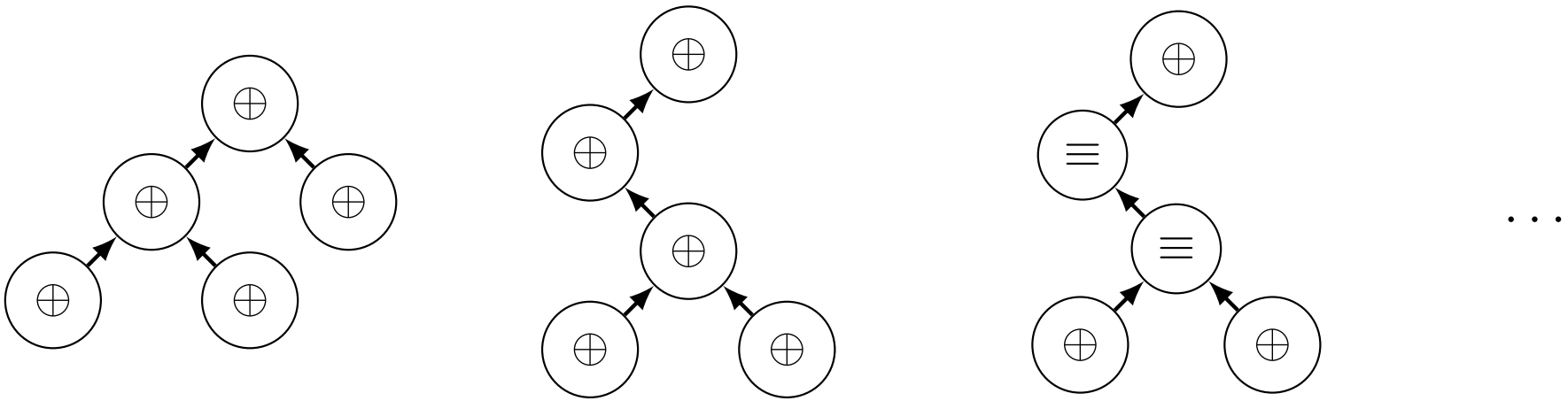
- When $\neg$ *count* towards size, structure of XOR_n is known
  - Optimal XOR_n ckts are binary trees of XOR_2 widgets [Kombarov 2011]
  - Max fanout is 2 and the # of optimal circuits is $2^{O(n)}$ as required
- We extend this characterization when $\neg$ gates are "free"
- Lengthy case analysis using gate elimination to build a $(\neg)\mathrm{XOR}_2$ widget which reads two variables: x_i and x_j



XOR_6 **circuits are binary trees of 5** $\mathrm{XOR}_2(\oplus)$ **or** $\neg\mathrm{XOR}_2(\equiv)$ **widgets**

Characterizing XOR

- When $\neg$ *count* towards size, structure of XOR_n is known
  - Optimal XOR_n ckts are binary trees of XOR_2 widgets [Kombarov 2011]
  - Max fanout is 2 and the # of optimal circuits is $2^{O(n)}$ as required
- We extend this characterization when $\neg$ gates are "free"
- Lengthy case analysis using gate elimination to build a $(\neg)\mathrm{XOR}_2$ widget which reads two variables: x_i and x_j
- This structure + our FPT algorithm gives us that XOR-SEP is in P



XOR_6 **circuits are binary trees of 5** $\mathrm{XOR}_2(\oplus)$ **or** $\neg\mathrm{XOR}_2(\equiv)$ **widgets**

Table of Contents

Towards ETH-hardness of MCSP

- Goal: Find a function $\mathcal{F}$ which bypasses our algorithm

Towards ETH-hardness of MCSP

- Goal: Find a function $\mathcal{F}$ which bypasses our algorithm
- All known explicit bounds are linear, more likely to bypass fanout

Towards ETH-hardness of MCSP

- Goal: Find a function $\mathcal{F}$ which bypasses our algorithm
- All known explicit bounds are linear, more likely to bypass fanout

Function(s)	Lower Bound	Upper Bound	$\Omega(1)$ Fanout?
XOR	$\mathbf{3(n-1)}$	$\mathbf{3(n-1)}$	NO
Sum Mod 4	$4n - O(1)$	$5n - O(1)$	NO?
Sum Mod 2^k	$4n - O(1)$	$7n - o(n)$	NO?
Multiplexer	$2(n-1)$	$2n + O(\sqrt{n})$	YES?
Well-Mixed	$5n - o(n)$	poly	MAYBE?
Weighted Sum of Parities	$5n - o(n)$	$5n + o(n)$	YES?

Towards ETH-hardness of MCSP

- Goal: Find a function $\mathcal{F}$ which bypasses our algorithm
- All known explicit bounds are linear, more likely to bypass fanout

Function(s)	Lower Bound	Upper Bound	$\Omega(1)$ Fanout?
XOR	$\mathbf{3(n-1)}$	$\mathbf{3(n-1)}$	NO
Sum Mod 4	$4n - O(1)$	$5n - O(1)$	NO?
Sum Mod 2^k	$4n - O(1)$	$7n - o(n)$	NO?
Multiplexer	$2(n-1)$	$2n + O(\sqrt{n})$	YES?
Well-Mixed	$5n - o(n)$	poly	MAYBE?
Weighted Sum of Parities	$5n - o(n)$	$5n + o(n)$	YES?

- More generally, can we learn more about the *structure* of optimal circuits, not just bounds?

Thank You!

https://cs-people.bu.edu/ndang

Ngu (Nathan) Dang

Boston University

Email: ndang@bu.edu

Building an XOR_2 Widget

- Let C be an optimal circuit for XOR_n

Building an XOR_2 Widget

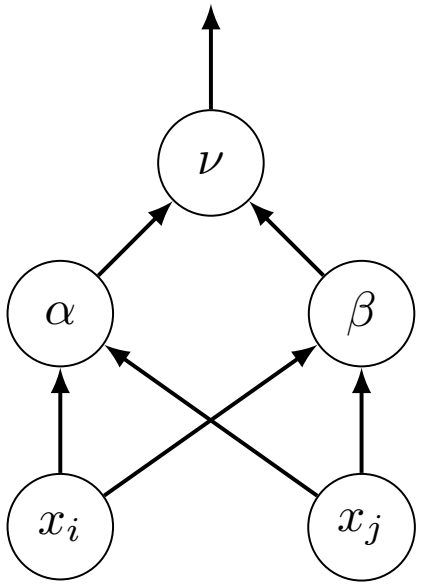
- Let C be an optimal circuit for XOR_n
- Let α be a gate reading two variables x_i, x_j

Building an XOR_2 Widget

- Let C be an optimal circuit for XOR_n
- Let α be a gate reading two variables x_i, x_j
- Goal: Build up $(\neg)\mathrm{XOR}_2$ widget around α

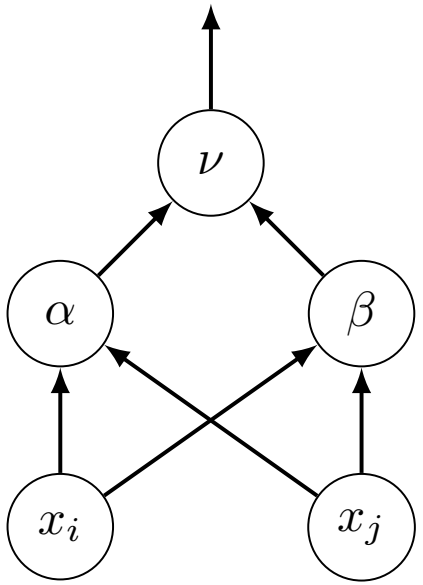
Building an XOR_2 Widget

- Let C be an optimal circuit for XOR_n
- Let α be a gate reading two variables x_i, x_j
- Goal: Build up $(\neg)\mathrm{XOR}_2$ widget around α



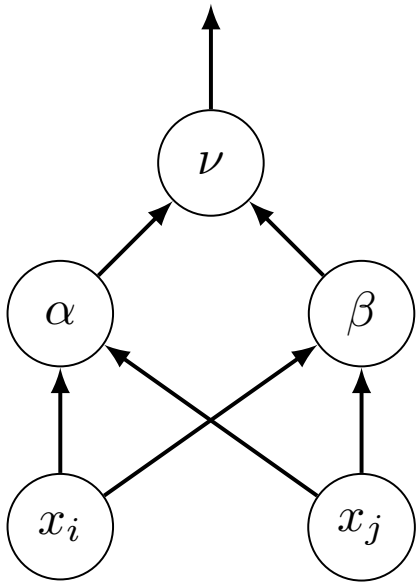
Building an XOR_2 Widget

- Let C be an optimal circuit for XOR_n
- Let α be a gate reading two variables x_i, x_j
- Goal: Build up $(\neg)\mathrm{XOR}_2$ widget around α
- Repeated arguments via contradiction



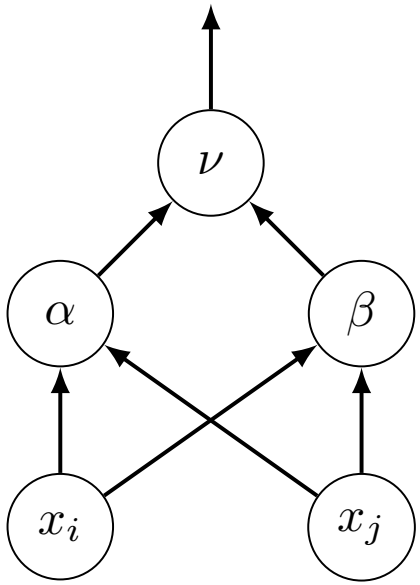
Building an XOR_2 Widget

- Let C be an optimal circuit for XOR_n
- Let α be a gate reading two variables x_i, x_j
- Goal: Build up $(\neg)\mathrm{XOR}_2$ widget around α
- Repeated arguments via contradiction
  - Assume x_i and x_j do not feed a XOR_2 widget



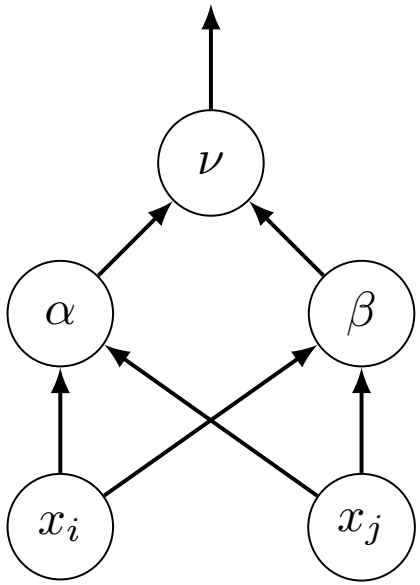
Building an XOR_2 Widget

- Let C be an optimal circuit for XOR_n
- Let α be a gate reading two variables x_i, x_j
- Goal: Build up $(\neg)\mathrm{XOR}_2$ widget around α
- Repeated arguments via contradiction
  - Assume x_i and x_j do not feed a XOR_2 widget
  - Substitute $x_i \leftarrow b$ and simplify



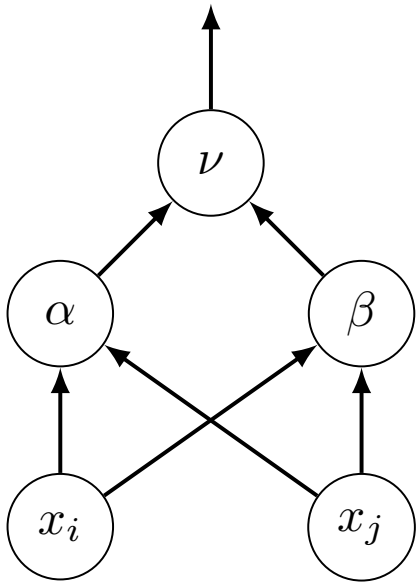
Building an XOR_2 Widget

- Let C be an optimal circuit for XOR_n
- Let α be a gate reading two variables x_i, x_j
- Goal: Build up $(\neg)\mathrm{XOR}_2$ widget around α
- Repeated arguments via contradiction
  - Assume x_i and x_j do not feed a XOR_2 widget
  - Substitute $x_i \leftarrow b$ and simplify
  - Violate some fact about $\mathrm{XOR}_n\restriction_{x_i \leftarrow b}$



Building an XOR_2 Widget

- Let C be an optimal circuit for XOR_n
- Let α be a gate reading two variables x_i, x_j
- Goal: Build up $(\neg)\mathrm{XOR}_2$ widget around α
- Repeated arguments via contradiction
  - Assume x_i and x_j do not feed a XOR_2 widget
  - Substitute $x_i \leftarrow b$ and simplify
  - Violate some fact about $\mathrm{XOR}_n\restriction_{x_i \leftarrow b}$
    - Like x_j is no longer read by the circuit



Building an XOR_2 Widget

- Let C be an optimal circuit for XOR_n
- Let α be a gate reading two variables x_i, x_j
- Goal: Build up $(\neg)\mathrm{XOR}_2$ widget around α
- Repeated arguments via contradiction
  - Assume x_i and x_j do not feed a XOR_2 widget
  - Substitute $x_i \leftarrow b$ and simplify
  - Violate some fact about $\mathrm{XOR}_n\restriction_{x_i \leftarrow b}$
    - Like x_j is no longer read by the circuit
    - Or we eliminate too many gates