Symmetric Is Better: Can We Exploit Regularities in Logic Synthesis?

Valentina Ciriani

This work was supported in part by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them.
Logic Synthesis

**Input:** Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$

**Output:** circuit

Algebraic representation

$$F = (\neg A \lor B) \land C$$

**VHDL**

```vhdl
... ARCHITECTURE synthesis OF andor IS
BEGIN
  p <= a AND b;
  o <= a OR p
END synthesis;
```

**On-set Truth Table**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
</tr>
<tr>
<td>1011</td>
<td>1</td>
</tr>
<tr>
<td>1111</td>
<td>1</td>
</tr>
</tbody>
</table>

The circuit computes the given Boolean function $f$
Logic Synthesis depends on:
1) The input representation
2) The output circuit
3) The minimization metric(s):
   1) Area
   2) Delay
   3) Power consumption
   4) Testability
   5) Security objectives
   6) ...
Two simple cases of the Logic Synthesis problem:

**Problem: Truth Table to MIN SOP**
*Instance:* truth table of the Boolean function \( f \) and an integer \( k \)
*Question:* is there a DNF (or SOP) formula, representation of \( f \), containing at most \( k \) terms?
*Complexity:* NP-complete

**Problem: SOP to MIN SOP**
*Instance:* algebraic representation in DNF (or SOP) \( p \) and an integer \( k \)
*Question:* is there a DNF formula \( p' \), equivalent to \( p \), containing at most \( k \) terms?
*Complexity:* \( \Sigma_2^P \)-complete (i.e., NP\(^{NP}\)-complete) in the polynomial hierarchy (polynomial-time nondeterministic Turing Machine with NP-oracle)

Function regularities: motivations

- Real life functions have often a regular structure
- Synthesis algorithms do not take into account function regularities
- Idea: exploiting function regularities for ease their synthesis
Boolean function regularities

**Input:** Boolean function \( f: \{0,1\}^n \rightarrow \{0,1\} \)

- **Algebraic representation:**
  \[ F = (\neg A \lor B) \lor C \]

- **VHDL**
  ```vhdl
  ARCHITECTURE synthesis OF andor IS
  BEGIN
  p <= a AND b;
  o <= a OR p
  END synthesis;
  ```

- **On-set Truth Table**
<table>
<thead>
<tr>
<th></th>
<th>0010</th>
<th>0100</th>
<th>0101</th>
<th>0001</th>
<th>1011</th>
<th>0111</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Output:** circuit

- **CMOS circuit**
- **Reversible circuit (quantum)**
- **Switching lattice (nano-crossbar array)**
- **XOR-AND Inverter graph (XAG)**

- **Fast pre-processing phase:** *regularity test*
- **Fast post-processing phase:** *reconstruction*
Input: Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}$

- Algebraic representation: $F = (\neg A \lor B) \land C$
- VHDL:
  ```vhdl
  ... ARCHITECTURE synthesis OF andor IS BEGIN p <= a AND b; o <= a OR p END synthesis;
  ...
  ```
- On-set Truth Table:
  
<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010</td>
<td>0</td>
</tr>
<tr>
<td>0100</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
</tr>
<tr>
<td>1011</td>
<td>1</td>
</tr>
<tr>
<td>1111</td>
<td>1</td>
</tr>
</tbody>
</table>

Output: circuit

- $f^R$ is smaller than $f$
- $f^R$ is smaller than $f$
- Synthesis
- Reconstruction
- $C$ and $C'$ are circuits for $f$
The **regularity test** and the **reconstruction** must be fast:

- polynomial complexity

**Regularity test:**
- The new function $f^R$ should be easier to be synthetized w.r.t. $f$
  - E.g. $f^R$ depends on less variables or covers less minterms

**Reconstruction:**
- With a limited number of new gates:
  - $C' = C^R +$ some additional gates
  - $C = C'$ or $C \neq C'$
  - $C$ and $C'$ must be circuits for $f$
Synthesis of regular functions: advantages

Exact Synthesis (exponential):
- Smaller input function eases the synthesis:
  - Less computational time

Heuristic Synthesis (polynomial):
- Smaller input function eases the synthesis:
  - Less computational time
  - Smaller circuits ($C'$ smaller than $C$)
Symmetry
A totally symmetric Boolean function
• is a Boolean function whose value does not depend on the order of its input variables
• The value depends only on the number of ones in the input

A heuristic procedure for logic synthesis that takes advantage of symmetry is described in

Function regularities

Autosymmetry
Function regularities: autosymmetry

• We express the regularity of $f : \{0,1\}^n \to \{0,1\}$ by an *autosymmetry degree* $k$ ($0 \leq k \leq n$)

• $k = 0$: no regularity

• $k \geq 1$:
  • the function $f$ is *autosymmetric*
  • a new function $f_k$, called the *restriction* of $f$, is identified

• Note that:
  • an autosymmetric function $f$ depends on *all the $n$ input variables*
  • *but a smaller function* $f_k$ (with $n-k$ variables) is synthetized
### Autosimmetry: example

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$f_2$

\[
\begin{align*}
Y_1 &= X_2 \\
Y_2 &= X_1 \oplus X_3 \oplus X_4
\end{align*}
\]
Minimization strategy

- \( f \)
- \( f_k \)
- Equations
- Synthesis
- circuit for \( f_k \)
- circuit for \( f \)

Equations:

\[
\left( |f|, n \right)
\]

Polynomial:

\[
( |f|/2^k, n-k )
\]

Exponential/Polynomial:

Linear:

Minimization strategy
Logic network for $f$

Logic Network for $f_k$

$X_1, X_2, \ldots, X_n$ → $Y_1, Y_2, \ldots, Y_{n-k}$ → $f$
Autosymmetric function: complete example

\[ f(x_1, x_2, x_3, x_4) = y_1 \oplus y_2 \]

Pre-processing

- Regularity test

Post-processing

- Standard Synthesis of \( f \)
- Reconstruction

Reduction equations:

\[ y_1 = x_2 \]
\[ y_2 = x_1 \oplus x_3 \oplus x_4 \]

Restriction \( f_k \)
How may autosymmetric functions?

• The number of Boolean functions of $n$ variables is $2^{2^n}$

• The number of autosymmetric Boolean functions is $(2^n - 1)2^{2n-1}$

(The number of symmetric functions is $2^{n+1}$)

• Indeed, about 24% of the functions in classical benchmark suites have at least one autosymmetric output

• This is due to the regular nature of real-life functions
Standard synthesis of $f_k$

<table>
<thead>
<tr>
<th>Network for $f_k$</th>
<th>Network for $f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND-OR (SOP)</td>
<td>EXOR-AND-OR (ORAX)</td>
</tr>
<tr>
<td>EXOR-AND-OR (SPP)</td>
<td>EXOR-AND-OR (SPP)</td>
</tr>
<tr>
<td>AND-OR-AND</td>
<td>EXOR-AND-OR-AND</td>
</tr>
<tr>
<td>AND-OR-EXOR</td>
<td>EXOR-AND-OR-EXOR</td>
</tr>
</tbody>
</table>

- In general the network for $f$ has one more level
- SPP synthesis: the two level of EXORs are merged
- SPP synthesis: a minimal SPP $f_k$ gives a minimal SPP for $f$
Autosymmetric functions

• \(f\) is closed under \(\alpha \in \{1,0\}^n\) if

\[
\forall \omega \in \{1,0\}^n \quad f(\alpha \oplus \omega) = f(\omega)
\]

• Any \(f\) is closed under \(\alpha = 00....0\)

\[
\begin{array}{cccc}
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

is closed under

\[
\begin{array}{cccc}
0 & 0 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 \\
\end{array}
\]
The set $L_f = \{ \alpha \mid f \text{ is closed under } \alpha \}$ is a vector subspace of $\langle \{0,1\}^n, \oplus \rangle$.

- $f$ is $k$-autosymmetric if $\dim(L_f) = k$.
- $f$ is autosymmetric if $\dim(L_f) > 0$.

The function $f$ is autosymmetric if $f$ is closed under $L_f$. Here is an example:

\[
\begin{bmatrix}
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]

is closed under

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1
\end{bmatrix}
\]

$f$ is 1-autosymmetric.
Properties of autosymmetric functions

• A $k$-autosymmetric function $f$ is a disjoint union of $|f|/2^k$ affine spaces over $L_f$

\[
\begin{bmatrix}
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]

\[
L_f = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1
\end{bmatrix}
\]

Independent variable (canonical)
Construction of $f_k$

- $f_k$ consists of the $|f|/2^k$ points of $f$ contained in the subspace $\{0,1\}^{n-k}$ where all the canonical variables have value 0

$$
f_k = \bigcup \begin{array}{c}
1 0 1 \\
\oplus \\
1 0 1
\end{array} \bigcup \begin{array}{c}
0 0 0 0 \\
\oplus \\
0 0 0 0
\end{array} \bigcup \begin{array}{c}
1 1 0 0 \\
\oplus \\
0 0 1 1
\end{array}
$$

$$
f_1 = \begin{array}{c}
1 0 1 \\
1 1 0
\end{array}
$$
Construction of the equations

• The characteristic function representing the points of \( L_f \) is:

\[
\overline{X_1} \overline{X_2} (X_3 \oplus \overline{X_4})
\]

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1
\end{bmatrix}
\]

The \( Y_i \) are the EXOR factors of this function without complementations:

\[
\begin{align*}
Y_1 &= X_1 \\
Y_2 &= X_2 \\
Y_3 &= X_3 \oplus X_4
\end{align*}
\]
Theorem: Let $f$ be a Boolean function, then $L_f$:

$$L_f = \bigcap (u \oplus f) \text{ for each } u \in f$$

Algorithm:

1. For all $u \in f$, build $u \oplus f$;
2. Build $L_f = \bigcap (u \oplus f)$;
3. Compute $k = \log |L_f|$;

Complexity: $O(|f|^2 \cdot n)$
Autosymmetry test: example

\[ f = \begin{array}{cccc}
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array} \]

\[
f \oplus \begin{array}{c}
1001
\end{array} = \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
\end{array} 
\]

\[
f \oplus \begin{array}{c}
1010
\end{array} = \begin{array}{cccc}
0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
\end{array} 
\]

\[
f \oplus \begin{array}{c}
1100
\end{array} = \begin{array}{cccc}
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
\end{array} 
\]

\[
f \oplus \begin{array}{c}
1111
\end{array} = \begin{array}{cccc}
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 \\
\end{array} 
\]

This can be efficiently done exploiting ORBDDs

\[ \bigcap L_f = \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{array} \]
Function regularities

D-reducibility
D-reducibility: the idea

Projection into a space $A$ smaller than the entire Boolean space $B^n$
Example: algebraic expressions

- Minimum SOP expression for $f$: 
  \[ \overline{x}_1 x_3 \overline{x}_4 + \overline{x}_1 x_2 \overline{x}_4 + x_1 \overline{x}_2 x_3 x_4 + x_1 x_2 \overline{x}_3 x_4 \]

- Minimum SOP expression for $f_A$: 
  \[ \overline{x}_2 x_3 + \overline{x}_1 x_2 + x_2 \overline{x}_3 \]

- New form for $f$: 
  \[ (x_1 \oplus \overline{x}_4) (\overline{x}_2 x_3 + \overline{x}_1 x_2 + x_2 \overline{x}_3) \]
Minimization strategy

Polynomial

$\begin{align*}
 f \\
 f_A
\end{align*}$

Exponential/Polynomial

Synthesis

Linear

$\begin{align*}
 \text{circuit for } f_A \\
 \text{circuit for } f
\end{align*}$
Logic Network for $f$

$$X_1 \quad X_2 \quad \ldots \quad X_n$$

$$\text{Logic Network for } f_A$$

$$\text{AND gate}$$

$$f$$
### Example

**Truth Table for $f_A$**

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>$f_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**SOP for $f_A$**

\[
SOP(f_A) = \bar{X}_2X_3 + \bar{X}_1X_2 + X_2\bar{X}_3
\]

**Space A:**

\[
(x_1 \oplus \bar{x}_4)
\]

**Diagram:**

- **SOP for $f_A$**
- **EXOR gate**
- **AND gate**
- **Result** $f$
How can we find the smallest space containing a function $f$?

Observation:
- $\{0,1\}^n$ is a space containing $f$

The smallest vector space containing $f$
- is called: the support of $f$
- can be computed: with Gauss elimination

And the smaller affine space containing $f$?
The affine space $A$ over the vector space $V \subseteq \{0,1\}^n$ (with operator $\oplus$) is:

$$A = \{ p \oplus v \mid v \in V \} = p \oplus V$$

Affine space

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Vector space

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Translation point

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \oplus V$$

Translation
Point

Vector
Space
Affine space

Algebraic representation

$$x_1 \cdot (x_2 \oplus x_3 \oplus \overline{x_4})$$

Red: canonical variables
Black: non canonical variables

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$$= 1 \quad 0 \quad 0 \quad 0$$
Vector space

Algebraic representation:

\[ \overline{x}_1 \cdot (x_2 \oplus x_3 \oplus \overline{x}_4) \]

Red: canonical variables
Black: non canonical variables

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

0 0 0 0
0 0 1 1
0 1 0 1
0 1 1 0
Smallest affine space

- We use affine spaces for the projection
- Why?
- Properties of the smallest affine space $A$ containing $f$:
  - its representation is compact (AND of XORs expression)
  - $A$ can be computed in polynomial time
  - $A$ can be smaller than the smallest vector space containing $f$
Example

<table>
<thead>
<tr>
<th></th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

V = \overline{x_1}  

A = \overline{x_1}(x_2 \oplus x_3 \oplus x_4)

\begin{align*}
  f &= \overline{x_1}(\overline{x_2}x_3x_4 + \overline{x_2}x_3x_4 + x_2\overline{x_3}x_4) \\
  f &= \overline{x_1}(x_2 \oplus x_3 \oplus x_4) (x_2\overline{x_3} + x_2\overline{x_4})
\end{align*}

- 10 literals
- 8 literals
D-reducible functions

• A Boolean function \( f \) with \( n \) variables is **D-reducible** if
  • \( f \) is contained in an affine space \( A \) of dimension strictly smaller than \( n \)

• Let \( f \) be a D-reducible function and \( A \) its associated affine space. Then
  • \( f = A \cdot f_A \)
  • where \( f_A \) is the projection of \( f \) on \( A \)

• **Observations:**
  • \( f_A \) depends on \( \dim A (< n) \) variables
  • \( A \) is represented by an AND of EXOR
Synthesis of D-reducible functions

• $f = A \cdot f_A$

• D-reducibility test (construction of $A$)
  • Let $f' = v \oplus f$ with $v$ any vector of $f$
  • Gauss elimination on $f'$
    • computes a vector space $V$
  • $A = v \oplus V$
  • if $\text{dim } A < n$ than $f$ is D-reducible
    • of degree $k = n - \text{dim } A$
\[ f = \{0010, 0100, 0110, 1101, 1011\}, \quad v = 0010 \]
\[ f' = 0010 \oplus f = \{0000, 0110, 0100, 1111, 1001\} \]
\[ V = \{0000, 0010, 0100, 0110, 1001, 1101, 1011, 1111\} \]
\[ A = 0010 \oplus V = \{0000, 0010, 0100, 0110, 1001, 1101, 1011, 1111\} \]
Example

\[ A = \{0000, 0010, 0100, 0110, 1001, 1101, 1011, 1111\} \]

\[ f = \{0010, 0100, 0110, 1101, 1011\} \]

\[ f_A = \{001, 010, 011, 110, 101\} \]

\[ \text{minSOP}(f_A) = \overline{X_2}X_3 + \overline{X_1}X_2 + X_2\overline{X_3} \]
Example

\[ f = \text{CEX}(A) \cdot \text{minSOP}(f_A) = \]

\[(x_1 \oplus \overline{x}_4) (\overline{x}_2 x_3 + \overline{x}_1 x_2 + x_2 \overline{x}_3)\]

\[\text{minSOP}(f) = \]

\[\overline{x}_1 x_3 \overline{x}_4 + \overline{x}_1 x_2 \overline{x}_4 + x_1 \overline{x}_2 x_3 x_4 + x_1 x_2 \overline{x}_3 x_4\]

Properties

- new form is more compact
- but there are 3 levels of logic
D-reducibility and Autosymmetry

• D-reducible functions are not, in general, autosymmetric

BUT

• D-reducible functions are connected to autosymmetric functions through their Fourier transform:
  • Let $f$ be a D-reducible function with degree $k$
    • then its Fourier transform in absolute value is $k$-autosymmetric
  • Let $f$ be a $k$-autosymmetric function
    • then the characteristic function of its Fourier transform is D-reducible with degree $k$
In each new technology the key problem is to define an ad hoc \textit{reconstruction method}
Secure multiparty computation

Problem: Given \( N \) participants each having private data: \( d_1, d_2, \ldots, d_N \)
Compute the value of a public function: 
\[ f(d_1, d_2, \ldots, d_N) \]
while keeping their own inputs secret

The problem can be described as a synthesis problem where the output is a XAG and the minimization metric is the number of non-EXOR nodes
1. New regularities:
   • generalization of autosimmetry
   • XOR based regularities

2. Application to
   • emerging technologies
   • new Boolean problems

3. New regularity tests for
   • AIGs
   • Verilog inputs
Thanks!

Breakfast circuit!