VERIFYING ARITHMETIC CIRCUITS WITH (LINEAR) POLYNOMIALS

Daniela Kaufmann
TU Wien, Austria

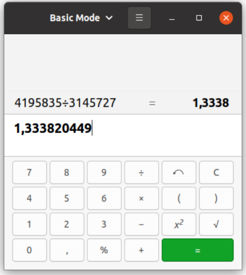
Invited Talk
SYNASC & AVM, Timisoara, Romania

September 24, 2025

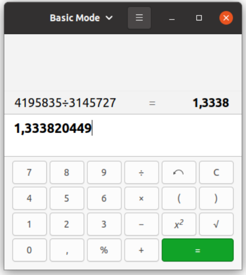
TU WIEN Informatics
FWF Austrian Science Fund

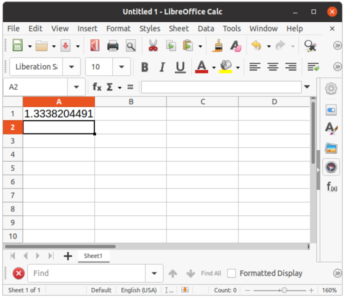
$$4\,195\,835 \div 3\,145\,727 = ?$$

$$4\,195\,835 \div 3\,145\,727 = ?$$

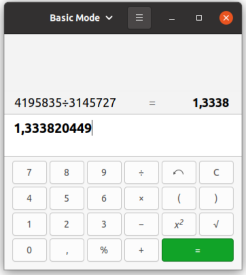


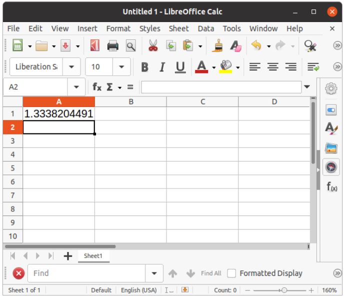
$$4\,195\,835 \div 3\,145\,727 = ?$$

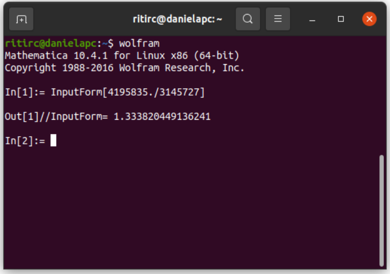


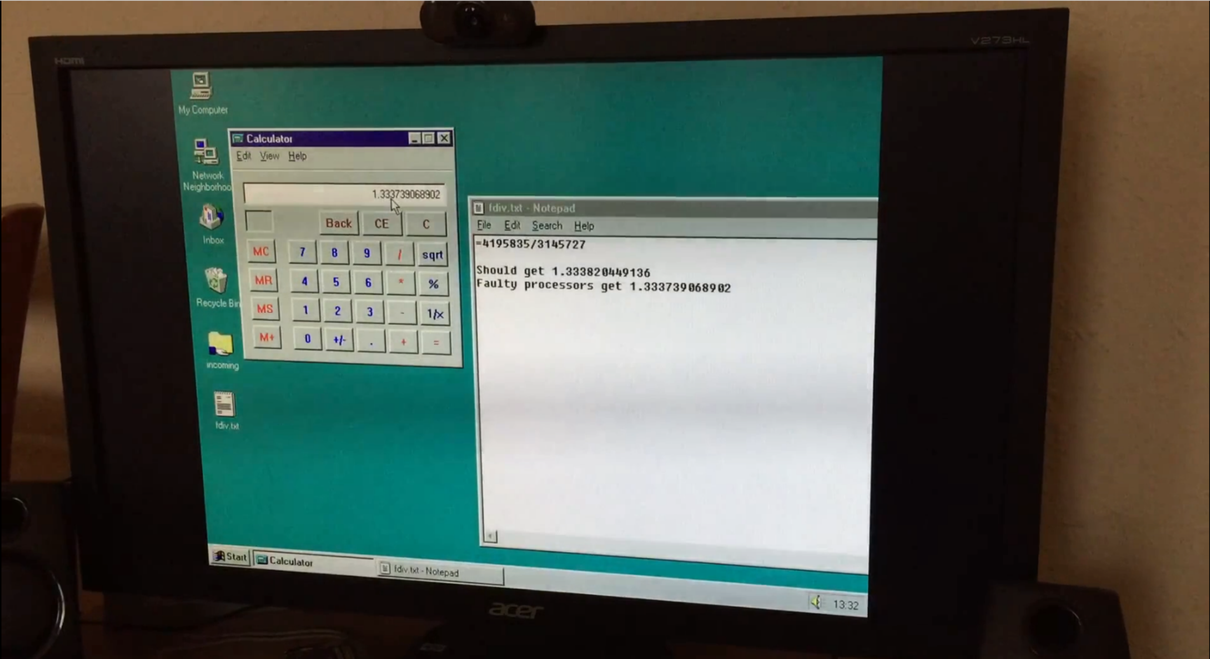


$$4\,195\,835 \div 3\,145\,727 = ?$$









Pentium FDIV bug and AMI GUI BIOS demo Quelle: https://youtu.be/hE7qMJVl15U

Intel Pentium FDIV-Bug 1994

Quelle: http://neology.com.au/portfolios/a80502-90-sx923/

- Affected floating point unit (FPU) in early Intel processors.
- Processor might return incorrect result for division.
- Cost in 1994: 500 million dollars.

Even more than 30 years later verification of arithmetic circuits is considered to be hard. Correctness proofs are not fully automated yet.

Challenge: Integer multipliers

Integer Multiplication

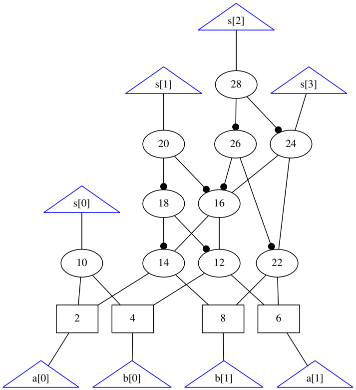
$$\begin{array}{ccccc} 1 & 1 & \cdot & 1 & 1 \\ \hline & & & 1 & 1 \\ & & 1 & 1 & \\ & {}_{1} & {}_{1} & {}_{0} & \\ \hline & 1 & 0 & 0 & 1 \end{array}$$

$3 \cdot 3 = 9$

Integer Multiplication

$$
\begin{array}{cccccc}
 & & 1 & 1 & \cdot & 1 \; 1 \\
\hline
 & & & & 1 & 1 \\
 & & {}_1 1 & {}_1 1 & {}_0 & \\
\hline
 & 1 & 0 & 0 & 1 &
\end{array}
$$

$3 \cdot 3 = 9$



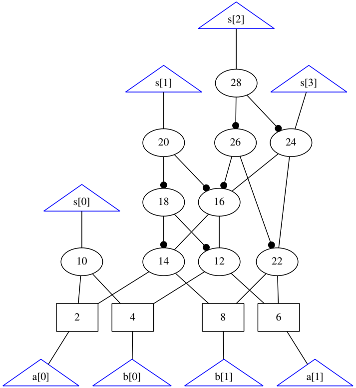
And-Inverter Graph

Multiplier Circuits

Given: Gate-level multiplier for fixed bit-width.

Question: For all possible $a_i, b_i \in \mathbb{B}$:

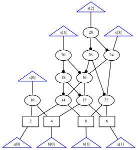
$$(2a_1 + a_0) * (2b_1 + b_0) = 8s_3 + 4s_2 + 2s_1 + s_0?$$



And-Inverter Graph

Formal Verification

System



Specification

For all $a_i, b_i \in \mathbb{B}$:

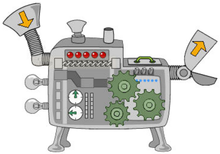
$$(2a_1 + a_0) * (2b_1 + b_0)$$

$$= 8s_3 + 4s_2 + 2s_1 + s_0 ?$$

Mathematical Model

$$
\begin{aligned}
B = \{ \\
& x - a_0 * b_0, \\
& y - a_1 * b_1, \\
& s_0 - x * y, \\
& \dots \\
\}
\end{aligned}
$$

Automated Decision Process



✓ | ✗

Formal Verification Techniques

Decision Diagrams

- First technique to detect Pentium bug

  [ChenBryant DAC'95]

- Requires knowledge of the layout

Formal Verification Techniques

Decision Diagrams

- First technique to detect Pentium bug

  [ChenBryant DAC'95]

- Requires knowledge of the layout

Theorem Proving

- Used in industry, e.g., ACL2

  [TemelSlobodovaHunt CAV'20]

- Automated on the RTL level

  [Temel TACAS'24]

Formal Verification Techniques

Decision Diagrams

- First technique to detect Pentium bug
  [ChenBryant DAC'95]
- Requires knowledge of the layout

Theorem Proving

- Used in industry, e.g., ACL2
  [TemelSlobodovaHunt CAV'20]
- Automated on the RTL level
  [Temel TACAS'24]

Satisfiability Checking (SAT)

- SAT 2016: Exponential run-time of solvers [Biere SATComp'16]
- SAT 2024: Equivalence checking of structural similar circuits
  [BiereFallerFazekasFleuryFroleyksPollitt SATComp'24]

Formal Verification Techniques

Decision Diagrams

- First technique to detect Pentium bug [ChenBryant DAC'95]
- Requires knowledge of the layout

Theorem Proving

- Used in industry, e.g., ACL2 [TemelSlobodovaHunt CAV'20]
- Automated on the RTL level [Temel TACAS'24]

Satisfiability Checking (SAT)

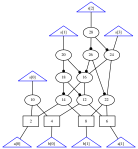
- SAT 2016: Exponential run-time of solvers [Biere SATComp'16]
- SAT 2024: Equivalence checking of structural similar circuits [BiereFallerFazekasFleuryFroleyksPollitt SATComp'24]

Algebraic Approach

- Seminal work: [LvKallaEnescu TCAD'13, CiesielskiYuBrownLiuRossi DAC'15] [SayedGroßeKühneSoekenDrechsler DATE'16]
- Polynomial encoding
- Works for non-trivial multiplier designs

Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$B = \{ \\ x - a_0 * b_0, \\ y - a_1 * b_1, \\ s_0 - x * y, \\ \dots \\ \}$$

Specification

$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$$

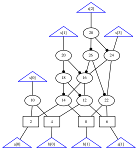
Implication

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$
\begin{aligned}
B = \{ & \\
& x - a_0 * b_0, \\
& y - a_1 * b_1, \\
& s_0 - x * y, \\
& \dots \\
\} &
\end{aligned}
$$

Specification

$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$$

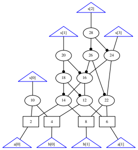
Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$
\begin{aligned}
B = \{ & \\
& x - a_0 * b_0, \\
& y - a_1 * b_1, \\
& s_0 - x * y, \\
& \ldots \\
\} &
\end{aligned}
$$

Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

Specification

$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$$

Multiplier Specification

Unsigned Integers:

$$0 = \sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \quad \in \mathbb{Q}[X]$$

Multiplier Specification

Unsigned Integers:

$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \quad \in \mathbb{Q}[X]$$

Multiplier Specification

Unsigned Integers:

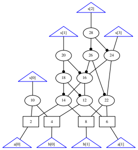
$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \quad \in \mathbb{Q}[X]$$

Signed Integers:

$$-2^{2n-1} s_{2n-1} + \sum_{i=0}^{2n-2} 2^i s_i - \left(-2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i\right)\left(-2^{n-1} b_{n-1} + \sum_{i=0}^{n-2} 2^i b_i\right) \quad \in \mathbb{Q}[X]$$

Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$
\begin{aligned}
B = \{ & \\
& x - a_0 * b_0, \\
& y - a_1 * b_1, \\
& s_0 - x * y, \\
& \dots \\
\} &
\end{aligned}
$$

Specification

$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$$

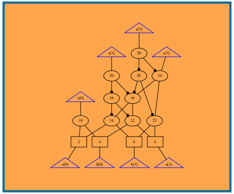
Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$
\begin{aligned}
B = \{ & \\
& x - a_0 * b_0, \\
& y - a_1 * b_1, \\
& s_0 - x * y, \\
& \ldots \\
\} &
\end{aligned}
$$

Specification

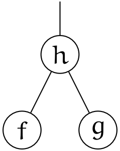
$$
\sum_{i=0}^{2n-1} 2^i s_i - \Big(\sum_{i=0}^{n-1} 2^i a_i\Big)\Big(\sum_{i=0}^{n-1} 2^i b_i\Big)
$$

Ideal Membership

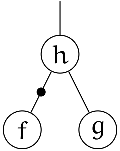
$= 0$ ✓

$\neq 0$ ✗

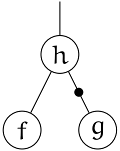
From AIGs to Polynomials



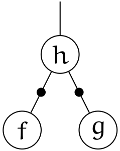
$h = f \wedge g$



$h = \neg f \wedge g$

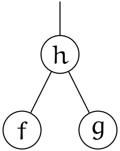


$h = f \wedge \neg g$



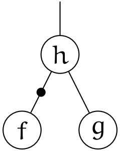
$h = \neg f \wedge \neg g$

From AIGs to Polynomials



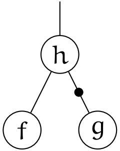
$h = f \wedge g$

f	g	h
0	0	0
0	1	0
1	0	0
1	1	1



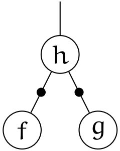
$h = \neg f \wedge g$

f	g	h
0	0	0
0	1	1
1	0	0
1	1	0



$h = f \wedge \neg g$

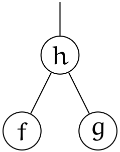
f	g	h
0	0	0
0	1	0
1	0	1
1	1	0



$h = \neg f \wedge \neg g$

f	g	h
0	0	1
0	1	0
1	0	0
1	1	0

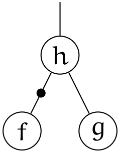
From AIGs to Polynomials



$$h = f \wedge g$$

f	g	h
0	0	0
0	1	0
1	0	0
1	1	1

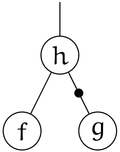
$$-h + fg$$



$$h = \neg f \wedge g$$

f	g	h
0	0	0
0	1	1
1	0	0
1	1	0

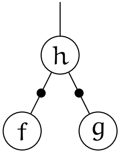
$$-h - fg + g$$



$$h = f \wedge \neg g$$

f	g	h
0	0	0
0	1	0
1	0	1
1	1	0

$$-h - fg + f$$



$$h = \neg f \wedge \neg g$$

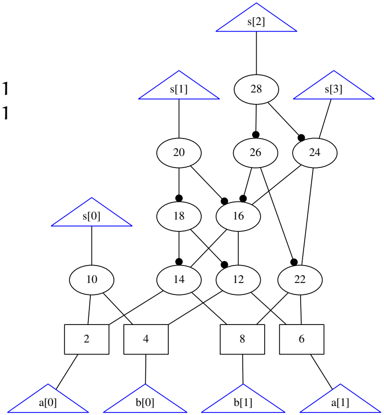
f	g	h
0	0	1
0	1	0
1	0	0
1	1	0

$$-h + fg - f - g + 1$$

From AIGs to Polynomials

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$
\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}
$$



From AIGs to Polynomials

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

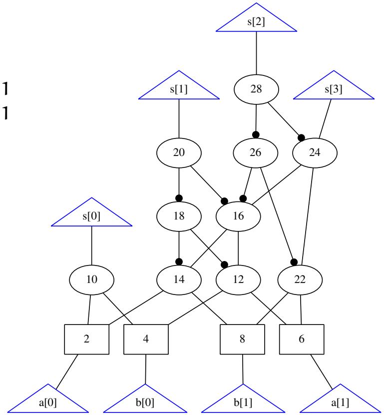
$$
\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}
$$

Boolean input constraints $B(C) \subseteq \mathbb{Q}[X]$.

$$
\begin{array}{ll}
a_1, a_0 \in \mathbb{B} & -a_1^2 + a_1,\ -a_0^2 + a_0, \\
b_1, b_0 \in \mathbb{B} & -b_1^2 + b_1,\ -b_0^2 + b_0
\end{array}
$$

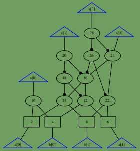
Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4b_1 a_1 - 2b_1 a_0 - 2b_0 a_1 - b_0 a_0$$



Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$
\begin{aligned}
B = \{ & \\
& x - a_0 * b_0, \\
& y - a_1 * b_1, \\
& s_0 - x * y, \\
& \ldots \\
\} &
\end{aligned}
$$

Specification

$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$$

Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

COMPUTER ALGEBRA

COMPUTER ALGEBRA

IDEALS AND GRÖBNER BASES

Ideal

Ideal. A subset $I \subset \mathbb{K}[X]$ is an ideal if it satisfies:

- $0 \in I$
- If $f, g \in I$, then $f + g \in I$.
- If $f \in I$ and $h \in \mathbb{K}[X]$ then $hf \in I$.

Ideal

Ideal. A subset $I \subset \mathbb{K}[X]$ is an ideal if it satisfies:

- $0 \in I$
- If $f, g \in I$, then $f + g \in I$.
- If $f \in I$ and $h \in \mathbb{K}[X]$ then $hf \in I$.

Ideal generated by a finite number of polynomials.
Let $f_1, \dots, f_s \in \mathbb{K}[X]$. Then we set

$$\langle f_1, \dots, f_s \rangle = \{h_1 f_1 + \cdots + h_s f_s \mid h_1, \dots, h_s \in \mathbb{K}[X]\}.$$

$\langle f_1, \dots, f_s \rangle$ is an ideal and is called the ideal generated by $f_1, \dots, f_s$.

We can think of $\langle f_1, \dots, f_s \rangle$ as consisting of all “polynomial consequences” of the equations $f_1 = f_2 = \ldots = f_s = 0$.

Applications of Ideals

The Ideal Membership Problem.

Given $f \in \mathbb{K}[X]$ and an ideal $I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{K}[X]$, determine if $f \in I$.

Applications of Ideals

The Ideal Membership Problem.

Given $f \in \mathbb{K}[X]$ and an ideal $I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{K}[X]$, determine if $f \in I$.

Divide f by $f_1, \ldots, f_s$?

Orderings on the Monomials

Univariate Polynomials - Sort by Degree.

$$\cdots > x^{m+1} > x^m > \cdots > x^2 > x > 1$$

Orderings on the Monomials

Univariate Polynomials - Sort by Degree.

$$\cdots > x^{m+1} > x^m > \cdots > x^2 > x > 1$$

Linear Polynomials - Sort by variables.

$$x_1 > x_2 > \cdots > x_n$$

Orderings on the Monomials

Univariate Polynomials - Sort by Degree.

$$\cdots > x^{m+1} > x^{m} > \cdots > x^{2} > x > 1$$

Linear Polynomials - Sort by variables.

$$x_1 > x_2 > \cdots > x_n$$

How to order non-linear multivariate polynomials in $\mathbb{K}[X]$?

Orderings on the Monomials

Let $\sigma_1 = x_1^{u_1} \cdots x_n^{u_n}, \qquad \sigma_2 = x_1^{v_1} \cdots x_n^{v_n}.$

Orderings on the Monomials

Let $\sigma_1 = x_1^{u_1} \cdots x_n^{u_n}, \qquad \sigma_2 = x_1^{v_1} \cdots x_n^{v_n}.$

Lexicographic Order $\prec_{\text{lex}}$.

$\sigma_1 \prec_{\text{lex}} \sigma_2$ iff there exists an index i with $u_j = v_j$ for all $j < i$, and $u_i < v_i$.

Example: $x_1^2 x_2^7 \prec_{\text{lex}} x_1^3 x_2 \prec_{\text{lex}} x_1^3 x_2^6$

Orderings on the Monomials

Let $\sigma_1 = x_1^{u_1} \cdots x_n^{u_n}$, $\qquad \sigma_2 = x_1^{v_1} \cdots x_n^{v_n}$.

Lexicographic Order $\prec_{\text{lex}}$.

$\sigma_1 \prec_{\text{lex}} \sigma_2$ iff there exists an index i with $u_j = v_j$ for all $j < i$, and $u_i < v_i$.

Example: $x_1^2 x_2^7 \prec_{\text{lex}} x_1^3 x_2 \prec_{\text{lex}} x_1^3 x_2^6$

Degree Lexicographic Order $\prec_{\text{dlex}}$.

$\sigma_1 \prec_{\text{dlex}} \sigma_2$ iff either $|\sigma_1| < |\sigma_2|$ or $|\sigma_1| = |\sigma_2|$ and $\sigma_1 \prec_{\text{lex}} \sigma_2$.

Example: $x_1^3 x_2 \prec_{\text{dlex}} x_1^2 x_2^7 \prec_{\text{dlex}} x_1^3 x_2^6$

Gröbner Bases

A Gröbner basis is a special kind of generating set for an ideal with the following properties:

1. Every $f \in \mathbb{K}[X]$ has a unique computable remainder $r \in \mathbb{K}[X]$ with respect to G.

Gröbner Bases

A Gröbner basis is a special kind of generating set for an ideal with the following properties:

1. Every $f \in \mathbb{K}[X]$ has a unique computable remainder $r \in \mathbb{K}[X]$ with respect to G.

2. $f - r \in \langle G \rangle$. Hence, $f \in \langle G \rangle$ iff the remainder of f with respect to G is zero.

Gröbner Bases

A Gröbner basis is a special kind of generating set for an ideal with the following properties:

1. Every $f \in \mathbb{K}[X]$ has a unique computable remainder $r \in \mathbb{K}[X]$ with respect to G.

2. $f - r \in \langle G \rangle$. Hence, $f \in \langle G \rangle$ iff the remainder of f with respect to G is zero.

3. Every ideal $I \subseteq \mathbb{K}[X]$ has a Gröbner basis G w.r.t. a fixed monomial order.

Gröbner Bases

A Gröbner basis is a special kind of generating set for an ideal with the following properties:

1. Every $f \in \mathbb{K}[X]$ has a unique computable remainder $r \in \mathbb{K}[X]$ with respect to G.

2. $f - r \in \langle G \rangle$. Hence, $f \in \langle G \rangle$ iff the remainder of f with respect to G is zero.

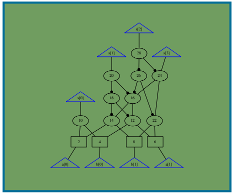
3. Every ideal $I \subseteq \mathbb{K}[X]$ has a Gröbner basis G w.r.t. a fixed monomial order.

4. Given an arbitrary basis, we can compute G (double-exponential).

BACK TO CIRCUITS

Basic Idea of Algebraic Approach

Multiplier



Polynomials

$$B = \{\; x - a_0 * b_0,\; y - a_1 * b_1,\; s_0 - x * y,\; \dots \;\}$$

Specification

$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$$

Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

Ideal Membership Problem

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$

Ideal Membership Problem

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.

Ideal Membership Problem

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.

Multiplier. A circuit C is called a multiplier if

$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i \right) \left(\sum_{i=0}^{n-1} 2^i b_i \right) \quad \in \quad J(C).$$

Verification Algorithm

Theorem (Kaufmann, Biere, Kauers FMCAD'17) For any circuit C, the set $G(C) \cup B(C)$ forms a Gröbner basis for $J(C)$ w.r.t. any lexicographic monomial order where "gate inputs $<$ gate output".

Verification Algorithm

Theorem (Kaufmann, Biere, Kauers FMCAD'17) For any circuit C, the set $G(C) \cup B(C)$ forms a Gröbner basis for $J(C)$ w.r.t. any lexicographic monomial order where "gate inputs $<$ gate output".

Reduce specification $\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$ by elements of $G(C) \cup B(C)$ until no further reduction is possible, then C is a multiplier iff remainder is zero.

Verification Algorithm

Theorem (Kaufmann, Biere, Kauers FMCAD'17) For any circuit C, the set $G(C) \cup B(C)$ forms a Gröbner basis for $J(C)$ w.r.t. any lexicographic monomial order where "gate inputs $<$ gate output".

Reduce specification $\sum_{i=0}^{2n-1} 2^i s_i - (\sum_{i=0}^{n-1} 2^i a_i)(\sum_{i=0}^{n-1} 2^i b_i)$ by elements of $G(C) \cup B(C)$

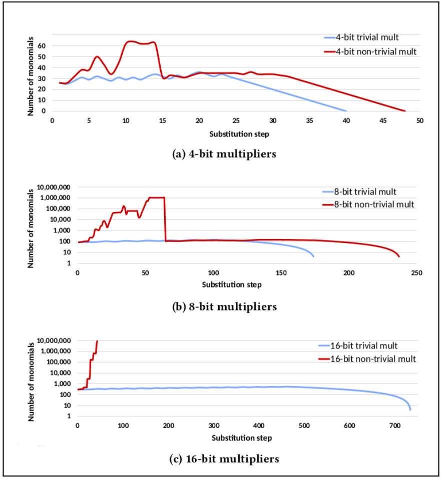
until no further reduction is possible, then C is a multiplier iff remainder is zero.

Computational Problems

- The number of monomials in the intermediate results blows-up.
- 8-bit multiplier cannot be verified within 20 minutes.

Verification Algorithm

[MahzoonGroßeDrechsler ICCAD'18]



Strategies

1. Encoding
   - Embedding different phases [KaufmannBeameBiereNordström DATE'22, KonradScholl FMCAD'24]
2. Preprocessing
   - Variable Elimination [MahzoonGroßeDrechsler DAC'19, RitircBiereKauers DATE'18]
3. Reduction
   - Incremental Algorithm [RitircBiereKauers FMCAD'17]
   - Dynamic Reduction Order [MahzoonGroßeSchollDrechsler DATE'20, KonradScholl FMCAD'24]
4. Tricky: OR Gates in final stage adder
   - Include SAT or BDDs [KaufmannBiereKauers FMCAD'19, DrechslerMahzoon ISEEIE'22]

Strategies

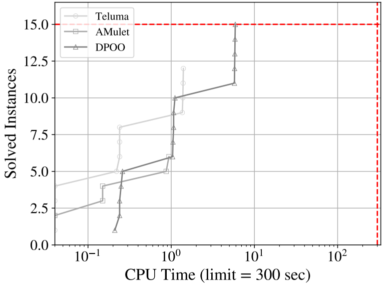
1. Encoding
   - Embedding different phases [KaufmannBeameBiereNordström DATE'22, KonradScholl FMCAD'24]
2. Preprocessing
   - Variable Elimination [MahzoonGroßeDrechsler DAC'19, RitircBiereKauers DATE'18]
3. Reduction
   - Incremental Algorithm [RitircBiereKauers FMCAD'17]
   - Dynamic Reduction Order [MahzoonGroßeSchollDrechsler DATE'20, KonradScholl FMCAD'24]
4. Tricky: OR Gates in final stage adder
   - Include SAT or BDDs [KaufmannBiereKauers FMCAD'19, DrechslerMahzoon ISEEIE'22]

All of these strategies rely on a **lexicographic term ordering** and aim to find an optimal reduction order.

Strategies

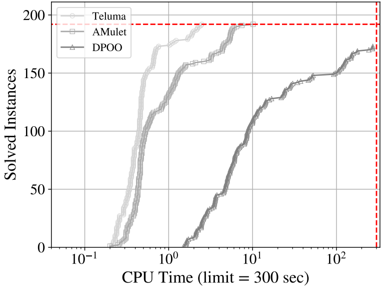
Optimized multipliers

- 5 architectures
- for 32, 64, 128 bit inputs



Unoptimized multipliers

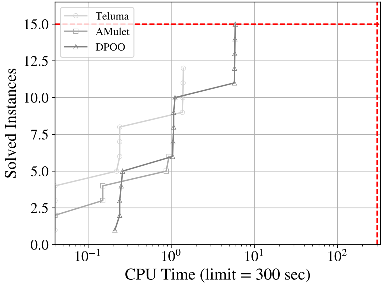
- 192 architectures
- 64 bit inputs



Strategies

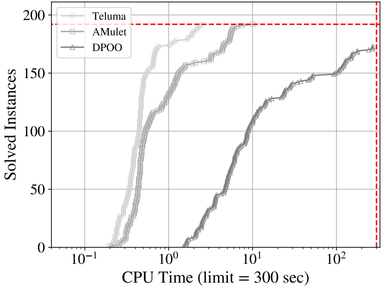
Optimized multipliers

- 5 architectures
- for 32, 64, 128 bit inputs



Unoptimized multipliers

- 192 architectures
- 64 bit inputs



👎 Fast approaches are overfitted

A New Idea

Change monomial order to degree-lexicographic order (Kaufmann, Berthomieu TACAS'25)

	$\prec_{\text{lex}}$	$\prec_{\text{dlex}}$
GB Computation	Easy	Hard
Spec Reduction	Hard	Easy

A New Idea

- Change monomial order to degree-lexicographic order (Kaufmann, Berthomieu TACAS'25)

	$\prec_{\text{lex}}$	$\prec_{\text{dlex}}$
GB Computation	Easy	Hard
Spec Reduction	Hard	Easy

Theorem (Kaufmann, Berthomieu TACAS'25) If the specification polynomial is linear, then the linear polynomials in a $\prec_{\text{dlex}}$-GB suffice to verify the circuit.

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

$G \leftarrow G(C) \cup B(C)$

$S_{\mathrm{lin}}, G \leftarrow$ Linearize-Spec(S, G)

$L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\mathrm{dlex}}$-GB(G)

$S_{\mathrm{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)

return $S_{\mathrm{lin}} \stackrel{?}{=} 0$

Verification via Linearization

Input: Circuit C, specification polynomial S

Output: Determine whether C fulfils S

$G \leftarrow G(C) \cup B(C)$

$S_{\mathrm{lin}}, G \leftarrow$ $\boxed{\text{Linearize-Spec}(S, G)}$

$L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\mathrm{dlex}}$-GB(G)

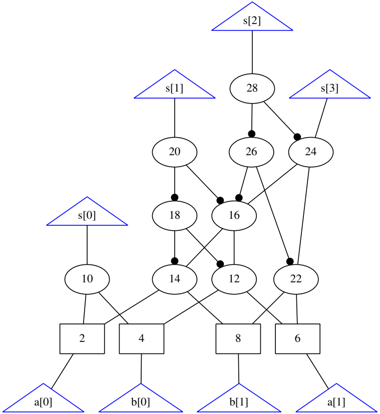
$S_{\mathrm{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)

return $S_{\mathrm{lin}} \stackrel{?}{=} 0$

$S_{\mathrm{lin}}, G_{\mathrm{ext}} \leftarrow$ Linearize-Spec(S, G)

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$
\begin{aligned}
&-s_3 + l_{24} && -l_{22} + a_1 b_1 \\
&-s_2 + l_{28} && -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
&-s_1 + l_{20} && -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
&-s_0 + l_{10} && -l_{16} + l_{14} l_{12} \\
&-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 && -l_{14} + a_0 b_1 \\
&-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 && -l_{12} + a_1 b_0 \\
&-l_{24} + l_{22} l_{16} && -l_{10} + a_0 b_0
\end{aligned}
$$



Specification $S \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4b_1 a_1 - 2b_1 a_0 - 2b_0 a_1 - b_0 a_0$$

$S_{\text{lin}}, G_{\text{ext}} \leftarrow$ **Linearize-Spec**(S, G)

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

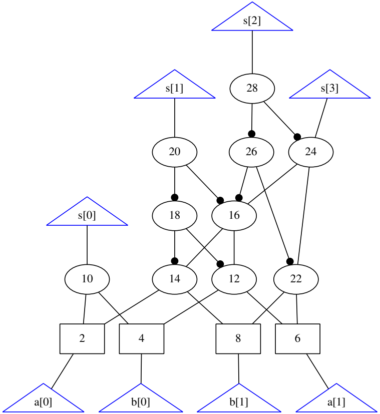
$$
\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}
$$

Extension polynomials $G_{\text{ext}} \subseteq \mathbb{Q}[X]$.

$$
\begin{array}{ll}
-t_{11} + a_1 b_1 & -t_{01} + a_0 b_1 \\
-t_{10} + a_1 b_0 & -t_{00} + a_0 b_0
\end{array}
$$

Linear Specification $S_{\text{lin}} \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4t_{11} - 2t_{10} - 2t_{01} - t_{00}$$



$S_{\text{lin}}, G_{\text{ext}} \leftarrow$ **Linearize-Spec**(S, G)

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

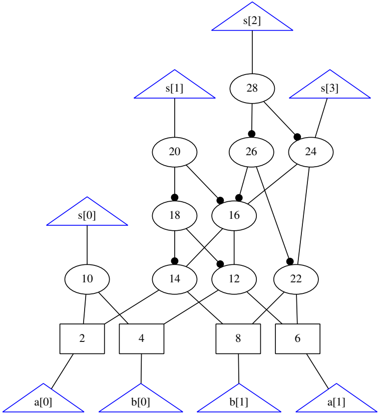
$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0b_1$$
$$-l_{12} + a_1b_0$$
$$-l_{10} + a_0b_0$$

Extension polynomials $G_{\text{ext}} \subseteq \mathbb{Q}[X]$.

$$-t_{11} + a_1b_1 \qquad -t_{01} + a_0b_1$$
$$-t_{10} + a_1b_0 \qquad -t_{00} + a_0b_0$$

Linear Specification $S_{\text{lin}} \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4t_{11} - 2t_{10} - 2t_{01} - t_{00}$$



$S_{\text{lin}}, G_{\text{ext}} \leftarrow$ **Linearize-Spec(S, G)**

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

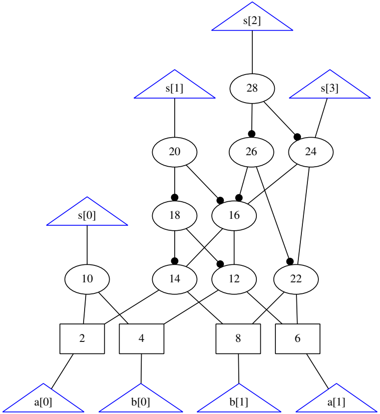
$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$

$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0 b_1$$
$$-l_{12} + a_1 b_0$$
$$-l_{10} + a_0 b_0$$

Extension polynomials $G_{\text{ext}} \subseteq \mathbb{Q}[X]$.

Linear Specification $S_{\text{lin}} \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$



Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

$G \leftarrow G(C) \cup B(C)$

$S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)

$L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\text{dlex}}$-GB(G)

$S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)

return $S_{\text{lin}} \stackrel{?}{=} 0$

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

$G \leftarrow G(C) \cup B(C)$

$S_{\mathrm{lin}}, G \leftarrow$ Linearize-Spec(S, G)

compute $\prec_{\mathrm{dlex}}$-GB of G

$L \leftarrow \boxed{\text{LINEAR-POLIES-OF-}\prec_{\mathrm{dlex}}\text{-GB}(G)}$

$S_{\mathrm{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)

return $S_{\mathrm{lin}} \stackrel{?}{=} 0$

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

$G \leftarrow G(C) \cup B(C)$

compute $\prec_{\text{dlex}}$-GB of G

$S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)

$L \leftarrow \boxed{\text{LINEAR-POLIES-OF-}\prec_{\text{dlex}}\text{-GB}(G)}$

$S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)

return $S_{\text{lin}} \stackrel{?}{=} 0$

Computing a $\prec_{\text{dlex}}$-GB for the whole circuit is infeasible.

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

1 $G \leftarrow G(C) \cup B(C)$
2 $S_{\mathrm{lin}}, G \leftarrow$ Linearize-Spec(S, G)
3 $L \leftarrow$ LINEAR-POLIES-OF-$\prec_{\mathrm{dlex}}$-GB(G)
4 $S_{\mathrm{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)
5 **return** $S_{\mathrm{lin}} \stackrel{?}{=} 0$

compute $\prec_{\mathrm{dlex}}$-GB of $G' \subseteq G$

Computing a $\prec_{\mathrm{dlex}}$-GB for the whole circuit is infeasible.

$\rightsquigarrow$ work locally: extract sub-circuits and only compute $\prec_{\mathrm{dlex}}$-GBs for them

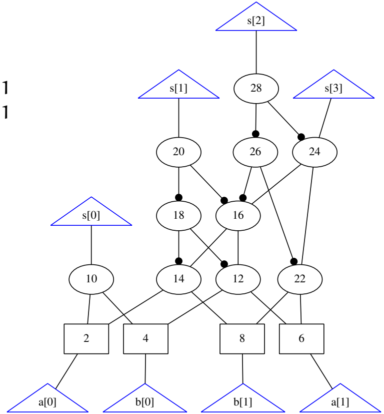
Local Linearization

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0 b_1$$
$$-l_{12} + a_1 b_0$$
$$-l_{10} + a_0 b_0$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$



Local Linearization

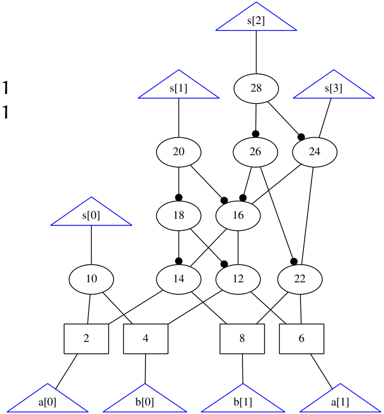
Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$$
$$-l_{16} + l_{14}l_{12}$$
$$-l_{14} + a_0 b_1$$
$$-l_{12} + a_1 b_0$$
$$-l_{10} + a_0 b_0$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$



Local Linearization

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

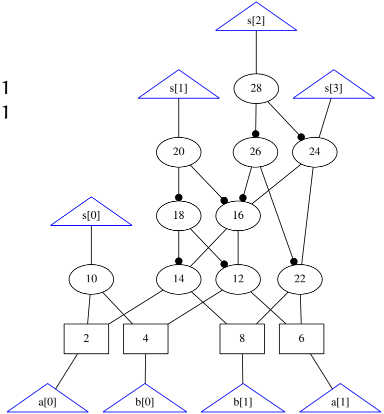
$$\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}$$



Local Linearization

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

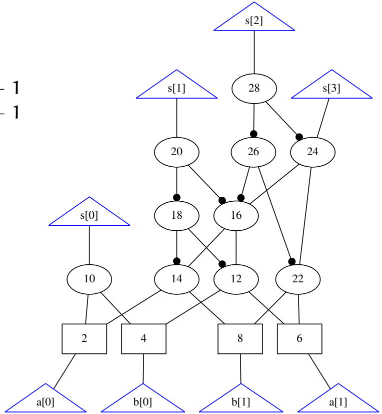
$$\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
\boxed{-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1} & -l_{14} + a_0 b_1 \\
-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

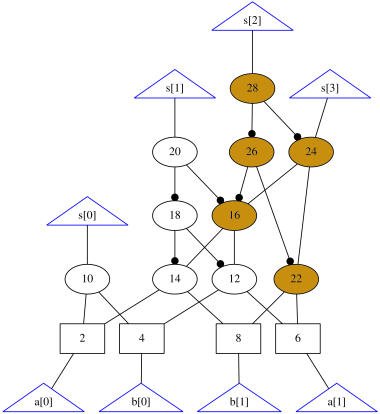
$$4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}$$



Local Linearization

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$
\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
\boxed{-l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1} & -l_{14} + a_0 b_1 \\
-l_{26} + l_{22} l_{16} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}
$$



Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}$$

Local Linearization

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

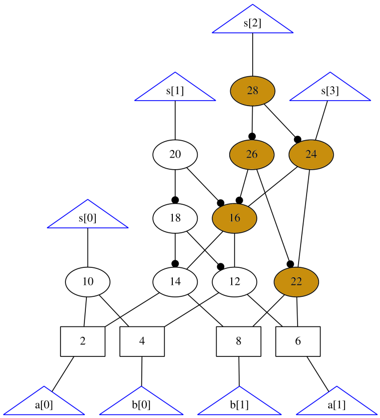
$$
\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26} + l_{24} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}
$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}$$



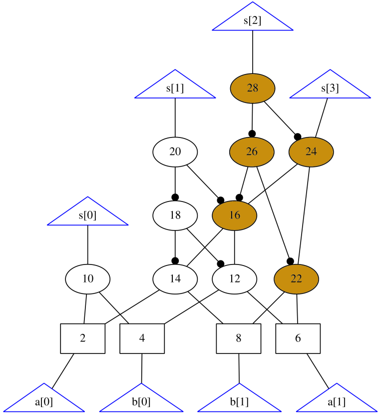
Local Linearization

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$
\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26} + l_{24} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}
$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} \\
-4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4
\end{aligned}
$$



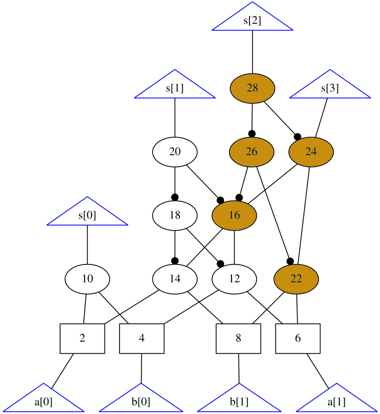
Local Linearization

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$
\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26} + l_{24} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}
$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} \\
-4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4 \\
2l_{20} + 4l_{16} - 2l_{14} - 2l_{12}
\end{aligned}
$$



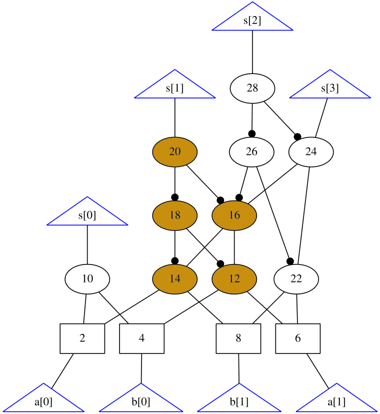
Local Linearization

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$
\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26} + l_{24} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}
$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} \\
-4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4 \\
2l_{20} + 4l_{16} - 2l_{14} - 2l_{12}
\end{aligned}
$$



Local Linearization

Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$\begin{array}{ll}
-s_3 + l_{24} & -l_{22} + a_1 b_1 \\
-s_2 + l_{28} & -l_{20} - l_{18} - l_{16} + 1 \\
-s_1 + l_{20} & -l_{18} + l_{16} - l_{14} - l_{12} + 1 \\
-s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\
-l_{28} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\
-l_{26} + l_{24} - l_{22} - l_{16} + 1 & -l_{12} + a_1 b_0 \\
-l_{24} + l_{22} l_{16} & -l_{10} + a_0 b_0
\end{array}$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

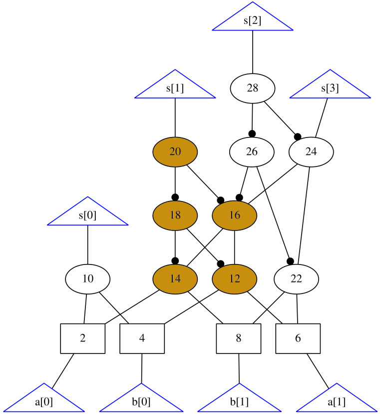
$$4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10}$$

$$4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12}$$

$$-4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4$$

$$2l_{20} + 4l_{16} - 2l_{14} - 2l_{12}$$

$$-2l_{18} + 2l_{16} - 2l_{14} - 2l_{12} + 2$$



Local Linearization

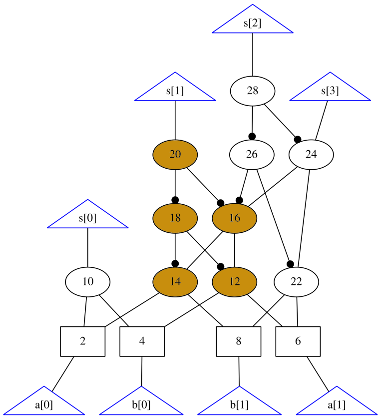
Gate polynomials $G(C) \subseteq \mathbb{Q}[X]$.

$$
\begin{aligned}
&-s_3 + l_{24} \\
&-s_2 + l_{28} \\
&-s_1 + l_{20} \\
&-s_0 + l_{10} \\
&-l_{28} - l_{26} - l_{24} + 1 \\
&-l_{26} + l_{24} - l_{22} - l_{16} + 1 \\
&-l_{24} + l_{22} l_{16}
\end{aligned}
$$

$$
\begin{aligned}
&-l_{22} + a_1 b_1 \\
&-l_{20} - l_{18} - l_{16} + 1 \\
&-l_{18} + l_{16} - l_{14} - l_{12} + 1 \\
&-l_{16} + l_{14} l_{12} \\
&-l_{14} + a_0 b_1 \\
&-l_{12} + a_1 b_0 \\
&-l_{10} + a_0 b_0
\end{aligned}
$$

Specification $\mathcal{S}_n \in \mathbb{Q}[X]$.

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4s_2 + 2s_1 + s_0 + 8l_{24} - 4l_{22} - 2l_{14} - 2l_{12} - l_{10} \\
4l_{28} + 8l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} \\
-4l_{26} + 4l_{24} - 4l_{22} + 2l_{20} - 2l_{14} - 2l_{12} + 4 \\
2l_{20} + 4l_{16} - 2l_{14} - 2l_{12} \\
-2l_{18} + 2l_{16} - 2l_{14} - 2l_{12} + 2 \\
0
\end{aligned}
$$

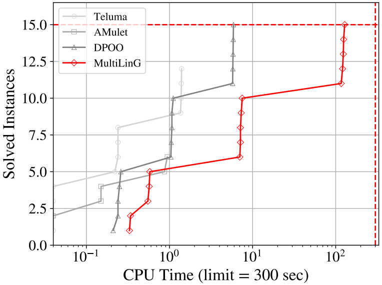


First Approach

(Kaufmann, Berthomieu TACAS'25)

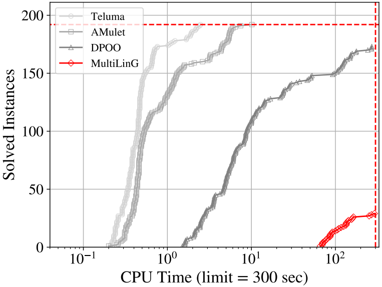
Optimized multipliers

- 5 architectures
- for 32, 64, 128 bit inputs



Unoptimized multipliers

- 192 architectures
- 64 bit inputs

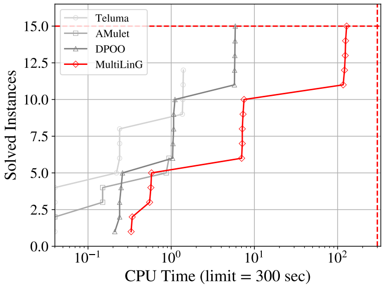


First Approach

(Kaufmann, Berthomieu TACAS'25)

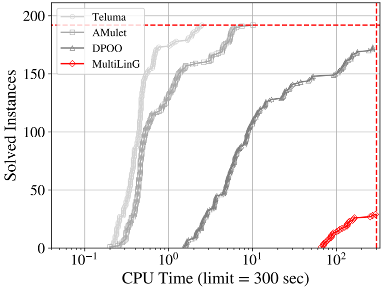
Optimized multipliers

- 5 architectures
- for 32, 64, 128 bit inputs



Unoptimized multipliers

- 192 architectures
- 64 bit inputs



- Approach seems to be more robust against optimization
- Black-box GB computation is too costly (can handle ≤ 8 vars)
- Completely ignores existing structure

Two Targeted Approaches

(Hofstadler, Kaufmann, CP'25)

FGLM-style Linearization

Guess & Prove Linearization

Two Targeted Approaches

(Hofstadler, Kaufmann, CP'25)

FGLM-style Linearization

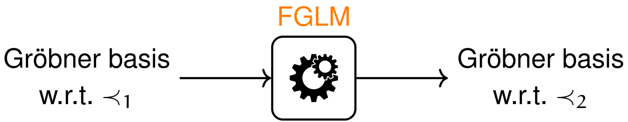
Guess & Prove Linearization

FGLM-style Linearization

- Exploit that we already have ($\prec_{\mathrm{lex}}$-)Gröbner basis.

FGLM-style Linearization

Exploit that we already have ($\prec_{\mathrm{lex}}$-)Gröbner basis.



One can adapt FGLM to compute all linear polynomials and then stop.

Exploit existing structure $\rightsquigarrow$ a lot faster than GB computation

FGLM-style Linearization

Exploit that we already have ($\prec_{\text{lex}}$-)Gröbner basis.

Gröbner basis w.r.t. $\prec_1$ $\xrightarrow{\text{FGLM}}$ Gröbner basis w.r.t. $\prec_2$

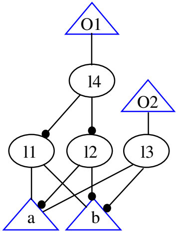
One can adapt FGLM to compute all linear polynomials and then stop.

- Exploit existing structure $\rightsquigarrow$ a lot faster than GB computation
- Runtime exponential in number of variables (can handle ≤ 15 vars)

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **generated by:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$



FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **generated by:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

1. Compute Normalforms:

$$l_1 = ab, \quad l_2 = ab - a - b + 1, \quad l_3 = -ab + a, \quad l_4 = -2ab + a + b$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **generated by:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

1. Compute Normalforms:

$$l_1 = ab, \quad l_2 = ab - a - b + 1, \quad l_3 = -ab + a, \quad l_4 = -2ab + a + b$$

2. Build Coefficient Matrix A:

$$\begin{array}{cc} \begin{array}{ccccccc} NF_G(1) & NF_G(a) & NF_G(b) & NF_G(l_1) & NF_G(l_2) & NF_G(l_3) & NF_G(l_4) \end{array} & \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & -1 & -2 \end{pmatrix} & \begin{array}{c} 1 \\ a \\ b \\ ab \end{array} \end{array}$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **generated by:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

3. Compute Kernel of A:

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{array} \\ \begin{pmatrix} 0 & -1 & -1 & 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

FGLM-style Linearization

Ideal $I \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **generated by:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

3. Compute Kernel of A:

$$\begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \\ \end{array}$$
$$\begin{pmatrix} 0 & -1 & -1 & 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & 0 & 0 \end{pmatrix}$$

4. Read off Linear Polynomials:

$$l_4 + 2l_1 - a - b, \qquad l_3 + l_1 - a, \qquad l_2 - l_1 + a + b - 1$$

Two Targeted Approaches

(Hofstadler, Kaufmann, CP'25)

FGLM-style Linearization

Guess & Prove Linearization

Two Targeted Approaches

(Hofstadler, Kaufmann, CP'25)

FGLM-style Linearization

Guess & Prove Linearization

Guess & Prove Linearization

- Exploit that we can compute (a lot of) points in the variety $V(G)$

Guess & Prove Linearization

Exploit that we can compute (a lot of) points in the variety $V(G)$

...and that our ideals are radical
...and that our ideals are zero-dimensional
...and that our ideals don't have additional roots in algebraic closure
...and that we can efficiently test ideal membership of linear polynomials

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{\, f \in \langle G \rangle \mid \deg(f) \leq 1 \,\}$

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{ f \in \langle G \rangle \mid \deg(f) \leq 1 \}$

Algorithm

```
Sample points P ⊆ V(G)
Compute all linear polynomials F that vanish on P
if F ⊆ ⟨G⟩
  return F
else add witness points to P and repeat
```

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{ f \in \langle G \rangle \mid \deg(f) \leq 1 \}$

Algorithm

1. Sample points $P \subseteq V(G)$ "guess"
2. Compute all linear polynomials F that vanish on P
3. **if** $F \subseteq \langle G \rangle$
4. **return** F
5. **else** add witness points to P and repeat

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{\, f \in \langle G \rangle \mid \deg(f) \leq 1 \,\}$

Algorithm

Sample points $P \subseteq V(G)$ "guess"
Compute all linear polynomials F that vanish on P
if $F \subseteq \langle G \rangle$ "prove"
  return F
else add witness points to P and repeat

Guess & Prove Linearization

Given: $G \subseteq \mathbb{K}[X]$ satisfying the assumptions from before

Compute: $\mathbb{K}$-VS basis of $\{ f \in \langle G \rangle \mid \deg(f) \leq 1 \}$

Algorithm

Sample points $P \subseteq V(G)$ "guess"
Compute all linear polynomials F that vanish on P
if $F \subseteq \langle G \rangle$ "prove"
  return F "repair"
else add witness points to P and repeat

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **consist of:**

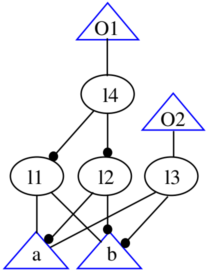
$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **consist of:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Sample: $(0, 0, 0, 1, 0, 0),\ (1, 0, 0, 0, 1, 1),\ (0, 1, 0, 0, 0, 1) \in V(G)$



Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **consist of:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Sample: $(0, 0, 0, 1, 0, 0),\ (1, 0, 0, 0, 1, 1),\ (0, 1, 0, 0, 0, 1) \in V(G)$

Matrix A**:**

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **consist of:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Kernel A**:**

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3, l_4]$ **consist of:**

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Kernel A**:**

$$\begin{array}{c} \begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Kernel yields Guesses:

$l_4 - b - a \qquad l_3 - a \qquad l_2 + a + b - 1 \qquad l_1$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $\quad l_4 - b - a \qquad l_3 - a \qquad l_2 + a + b - 1 \qquad l_1$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $\quad l_4 - b - a \quad\quad l_3 - a \quad\quad l_2 + a + b - 1 \quad\quad l_1$

Prove:

👎 Algebraic proof (reduction by G) does not scale!

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $l_4 - b - a \qquad l_3 - a \qquad l_2 + a + b - 1 \qquad l_1$

Prove:

- Algebraic proof (reduction by G) does not scale!
- Use a SAT solver!

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $\quad l_4 - b - a \qquad l_3 - a \qquad l_2 + a + b - 1 \qquad l_1$

Prove:

- Algebraic proof (reduction by G) does not scale!
- Use a SAT solver!

Encode $\displaystyle\bigwedge_{g \in G} (g = 0) \;\wedge\; (\text{guess} \neq 0)$ as propositional formula and give to a SAT solver.

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $\quad l_4 - b - a \quad\quad l_3 - a \quad\quad l_2 + a + b - 1 \quad\quad l_1$

Prove:

- Algebraic proof (reduction by G) does not scale!
- Use a SAT solver!

Encode $\quad \bigwedge_{g \in G} (g = 0) \; \wedge \; (\text{guess} \neq 0) \quad$ as propositional formula and give to a SAT solver.

If the result is

UNSAT, then guess $\in \langle G \rangle$

SAT, then we get $p \in V(G)$: guess$(p) \neq 0$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab, \quad l_2 - (1-a)(1-b), \quad l_3 - a(1-b), \quad l_4 - (1-l_1)(1-l_2), \quad a^2 - a, \quad b^2 - b$$

Guess: $\quad l_4 - b - a \quad\quad l_3 - a \quad\quad l_2 + a + b - 1 \quad\quad l_1$

Prove:

- Algebraic proof (reduction by G) does not scale!
- Use a SAT solver!

Encode $\displaystyle\bigwedge_{g \in G} (g = 0) \ \wedge\ (\text{guess} \neq 0)$ as propositional formula and give to a SAT solver.

If the result is

UNSAT, then guess $\in \langle G \rangle$

SAT, then we get $p \in V(G)$: guess$(p) \neq 0$

In this case, SAT solver returns SAT and $p = (1, 1, 1, 0, 0, 0)$.

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab \qquad l_2 - (1-a)(1-b) \qquad l_3 - a(1-b) \qquad a^2 - a \qquad b^2 - b$$

Repair Samples $\quad (0,0,0,1,0,0),\ (1,0,0,0,1,1),\ (0,1,0,0,0,1),\ (1,1,1,0,0,0) \in V(G)$

Matrix A:

$$\begin{array}{c}
\begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \end{array} \\
\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}
\end{array}$$

Guess & Prove Linearization

Let $G \subseteq \mathbb{Q}[a, b, l_1, l_2, l_3]$ consist of

$$l_1 - ab \qquad l_2 - (1-a)(1-b) \qquad l_3 - a(1-b) \qquad a^2 - a \qquad b^2 - b$$

Repair Samples $(0,0,0,1,0,0),\ (1,0,0,0,1,1),\ (0,1,0,0,0,1),\ (1,1,1,0,0,0) \in V(G)$

Kernel A**:**

$$\begin{array}{ccccccc} 1 & a & b & l_1 & l_2 & l_3 & l_4 \\ \end{array}$$
$$\begin{pmatrix} 0 & -1 & -1 & 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & 0 & 0 \end{pmatrix}$$

Kernel yields:

$$l_4 + 2l_1 - a - b, \qquad l_3 + l_1 - a, \qquad l_2 - l_1 + a + b - 1$$

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$?

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder...

...why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder…

…why not interpolate on all of $V(G)$? $\quad |V(G)| \approx 2^{80} \approx$ stars in universe

…how many samples do we need?

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder…

…why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

…how many samples do we need? $\sim 2^{14} \approx \frac{|V(G)|}{10\,000\,000\,000\,000\,000\,000}$

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder…

…why not interpolate on all of $V(G)$? $\quad |V(G)| \approx 2^{80} \approx$ stars in universe

…how many samples do we need? $\quad \sim 2^{14} \approx \frac{|V(G)|}{10\,000\,000\,000\,000\,000\,000}$

…how many iterations of the repair loop do we need?

Guess & Prove Linearization

Works really well also for larger circuits (can handle ≥ 200 variables)!

One might wonder…

…why not interpolate on all of $V(G)$? $|V(G)| \approx 2^{80} \approx$ stars in universe

…how many samples do we need? $\sim 2^{14} \approx \frac{|V(G)|}{10\,000\,000\,000\,000\,000\,000}$

…how many iterations of the repair loop do we need? ≤ 5

Verification via Linearization

Input: Circuit C, specification polynomial S
Output: Determine whether C fulfils S

$G \leftarrow G(C) \cup B(C)$

$S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)

$L \leftarrow \boxed{\text{LINEAR-POLIES-OF-}\prec_{\text{dlex}}\text{-GB}(G)}$

compute $\prec_{\text{dlex}}$-GB of $G' \subseteq G$

$S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)

return $S_{\text{lin}} \stackrel{?}{=} 0$

Verification via Linearization

Input: Circuit C, specification polynomial S

Output: Determine whether C fulfils S

$G \leftarrow G(C) \cup B(C)$

$S_{\text{lin}}, G \leftarrow$ Linearize-Spec(S, G)

$L \leftarrow \boxed{\text{LINEAR-POLIES-OF-}{\prec_{\text{dlex}}}\text{-GB}(G)}$

$S_{\text{lin}} \leftarrow$ Linear-Reduce(S_{lin}, L)

return $S_{\text{lin}} \stackrel{?}{=} 0$

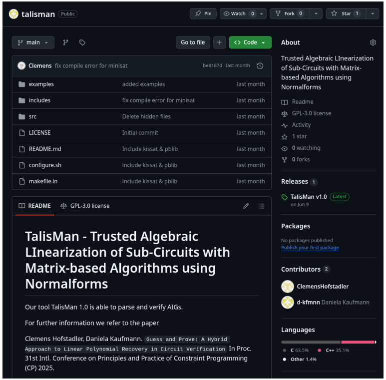
pick $G' \subseteq G$

if $|G'|$ is small use FGLM

else use Guess & Prove

TalisMan

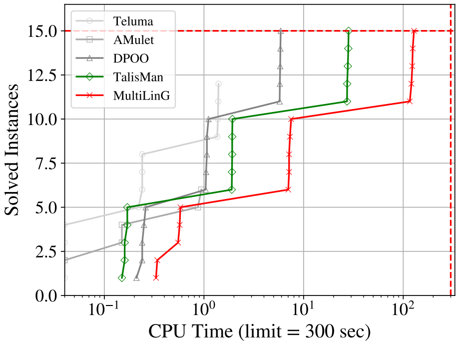
- https://github.com/d-kfmnn/talisman/
- Builds on MultiLing, written in C++
- Uses Flint for matrix computations
- Guess and Prove Algorithm:
  - Uses PBLib to translate guessed polynomial to CNF
  - Uses Kissat as a SAT solver
  - Uses caching of isomorphic subcircuits



Evaluation – Multiplier Circuits

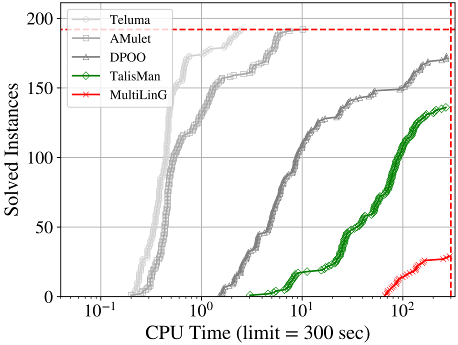
Optimized multipliers

- 5 architectures
- for 32, 64, 128 bit inputs



Unoptimized multipliers

- 192 architectures
- 64 bit inputs



Conclusion

Algebraic circuit verification (GB ✅, Reduction ⚠️)

Conclusion

Algebraic circuit verification (GB ✅, Reduction ⚠️)

💡 Flip things around: $\prec_{\text{lex}} \rightsquigarrow \prec_{\text{dlex}}$ (GB ⚠️, Reduction ✅)

Conclusion

Algebraic circuit verification (GB ✅, Reduction ⚠️)

💡 Flip things around: $\prec_{\text{lex}} \rightsquigarrow \prec_{\text{dlex}}$ (GB ⚠️, Reduction ✅)

Three approaches

Compute GB

- no requirements
- full GB computation
- double exponential runtime
- small circuits (≤ 10 variables)

FGLM

- requires a Gröbner basis
- first steps of FGLM
- exponential runtime
- moderate circuits (≤ 15 variables)

Guess & Prove

- requires variety
- interpolation (guess)
- uses SAT solver (prove)
- larger circuits (≥ 200 variables)

References I

[Biere SATComp'16] A. Biere. Collection of Combinational Arithmetic Miters Submitted to the SAT Competition 2016. In SAT Competition 2016, pages 65–66, Dep. of Computer Science Report Series B, University of Helsinki, 2016.

[BiereFallerFazekasFleuryFroleyksPollitt SATComp'24] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks and F. Pollitt CaDiCaL, Gimsatul, IsaSAT and Kissat Entering the SAT Competition 2024. In SAT Competition 2024, pages 8–10, Dep. of Computer Science Report Series B, University of Helsinki, 2024.

[Buchberger'65] B. Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. PhD Thesis, University of Innsbruck, 1965.

[CiesielskiYuBrownLiuRossi DAC'15] M. Ciesielski, C. Yu, W. Brown, D. Liu, and A. Rossi. Verification of Gate-level Arithmetic Circuits by Function Extraction. In Proc. of DAC'15, pages 52:1–52:6, ACM, 2015.

[ChenBryant DAC'95] Y. Chen and R. Bryant. Verification of Arithmetic Circuits with Binary Moment Diagrams. In Proc. of DAC'95, pages 535–541, ACM, 1995.

[DrechslerMahzoon ISEEIE'22] R. Drechsler and A. Mahzoon. Design Modification for Polynomial Formal Verification. In Proc. of ISEEIE'22, pages 187–194, IEEE, 2022.

References II

[KaufmannBeameBiereNordström DATE'22] D. Kaufmann, P.Beame, A. Biere and J. Nordström. Adding Dual Variables to Algebraic Reasoning for Gate-Level Multiplier Verification. In Proc. of DATE'22, pages 1431–1436, IEEE, 2022.

[KaufmannBiereKauers FMCAD'19] D. Kaufmann, A. Biere and M. Kauers. Verifying Large Multipliers by Combining SAT and Computer Algebra. In Proc. of FMCAD'19, pages 28–36, IEEE, 2019.

[KaufmannBiereKauers FMSD'20] D. Kaufmann, A. Biere and M. Kauers. Incremental Column-Wise Verification of Arithmetic Circuits Using Computer Algebra. In FMSD, vol 56, pages 22–54, 2020.

[KaufmannFleuryBiere FMCAD'20] D. Kaufmann, M. Fleury and A. Biere. Pacheck and Pastèque Checking Practical Algebraic Calculus Proofs. In Proc. of FMCAD'20, pages 264–269, TU Vienna Academic Press, 2020.

[KaufmannFleuryBiereKauers FMSD'21] D. Kaufmann, M. Fleury, A. Biere and M. Kauers. Practical Algebraic Calculus and Nullstellensatz with the Checkers Pacheck and Pastèque and Nuss-Checker. In FMSD, online first, 2021.

[KonradScholl FMCAD'24] A. Konrad and C. Scholl. Practical Algebraic Calculus and Nullstellensatz with the Checkers Pacheck and Pastèque and Nuss-Checker. In FMSD, online first, 2021.

References III

[LvKallaEnescu TCAD'13] J. Lv, P. Kalla, F. Enescu. Efficient Gröbner Basis Reductions for Formal Verification of Galois Field Arithmetic Circuits. In IEEE TCAD, vol. 32, pages 1409–1420, 2013.

[MahzoonGroßeDrechsler DAC'19] A. Mahzoon, D. Große and R. Drechsler. RevSCA: Using Reverse Engineering to Bring Light into Backwards Rewriting for Big and Dirty Multipliers. In Proc. of DAC'19, pages 185:1–185:6, ACM, 2019.

[MahzoonGroßeDrechsler ICCAD'18] A. Mahzoon, D. Große and R. Drechsler. PolyCleaner: Clean your Polynomials before Backward Rewriting to verify Million-gate Multipliers. In Proc. of ICCAD'18, pages 129:1–129:8, ACM, 2018.

[MahzoonGroßeSchollDrechsler DATE'20] A. Mahzoon, D. Große, Christoph Scholl and R. Drechsler. Towards Formal Verification of Optimized and Industrial Multipliers. In Proc. of DATE'20, pages 544–549, DATE, 2020.

[RitircBiereKauers DATE'18] D. Ritirc, A. Biere and M. Kauers. Improving and Extending the Algebraic Approach for Verifying Gate-Level Multipliers. In Proc. of DATE'18, pages 1556–1561, IEEE, 2018.

[RitircBiereKauers FMCAD'17] D. Ritirc, A. Biere and M. Kauers. Column-Wise Verification of Multipliers Using Computer Algebra. In Proc. of FMCAD'17, pages 23–30, IEEE, 2017.

References IV

[SayedGroßeKühneSoekenDrechsler DATE'16] A. Sayed-Ahmed, D. Große, U. Kühne, M. Soeken, and R. Drechsler. Formal verification of integer multipliers by combining Gröbner basis with logic reduction. In Proc. of DATE'16, pages 1048–1053, IEEE, 2016.

[SchollKonradMahzoonGroßeDrechsler DATE'21] C. Scholl, A. Konrad, A. Mahzoon, D. Große and R. Drechsler. Verifying Dividers Using Symbolic Computer Algebra and Don't Care Optimization. In Proc. of DATE'21, pages 1110–1115, IEEE, 2021.

[Temel TACAS'24] M. Temel eSCMul: Verified Implementation of S-C-Rewriting for Multiplier Verification. In Proc. of TACAS'24, pages 485–507, Springer, 2024.

[TemelSlobodovaHunt CAV'20] M. Temel, A. Slobodova and W. Hunt. Automated and Scalable Verification of Integer Multipliers. In Proc. of CAV'20, pages 485–507, Springer, 2020.