TAMING THE POLYNOMIAL EXPLOSION: A NEW APPROACH TO ALGEBRAIC CIRCUIT VERIFICATION

Daniela Kaufmann

TU Wien, Austria

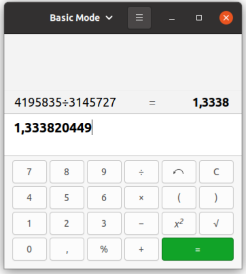
Joint work with Jérémy Berthomieu, Sorbonne Université, CNRS, Paris, France

Invited Talk
University of Freiburg, Germany

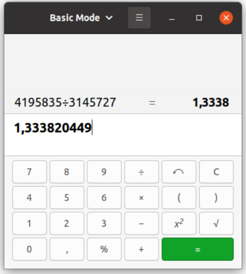
December 3, 2024

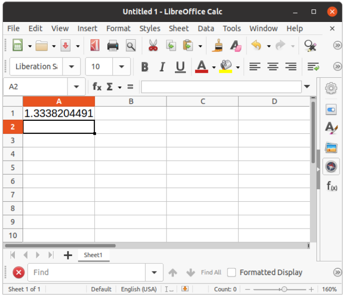
$$4\,195\,835 \;\div\; 3\,145\,727 \;=\; ?$$

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

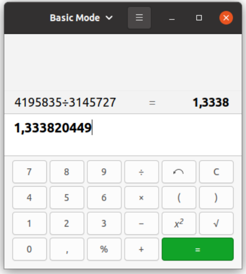


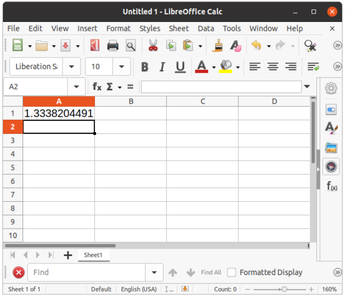
$$4\,195\,835 \;\div\; 3\,145\,727 \;=\; ?$$

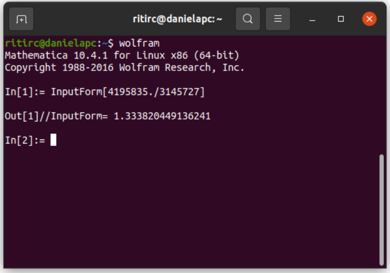


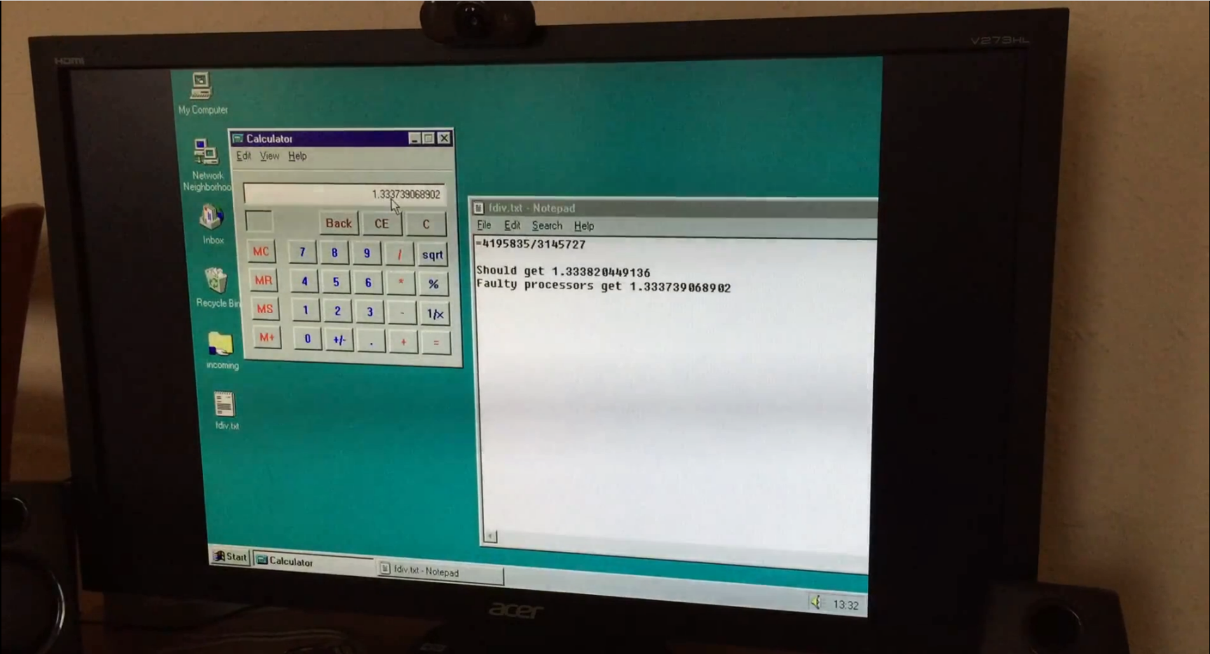


$$4\,195\,835 \div 3\,145\,727 = ?$$









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

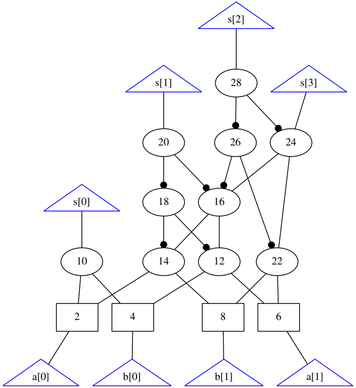
$$
\begin{array}{cccccc}
 & 1 & 1 & \cdot & 1 & 1 \\
\hline
 & & & & 1 & 1 \\
 & & & 1 & 1 & \\
 & & {}_1 & {}_1 & {}_0 & \\
\hline
 & & 1 & 0 & 0 & 1
\end{array}
$$

$3 \cdot 3 = 9$

Integer Multiplication

$$
\begin{array}{rrrr}
 & 1\,1 & \cdot & 1\,1 \\
\hline
 & & 1 & 1 \\
 & 1 & 1 & \\
{}_1 & {}_1 & {}_0 & \\
\hline
1 & 0 & 0 & 1
\end{array}
$$

$3 \cdot 3 = 9$



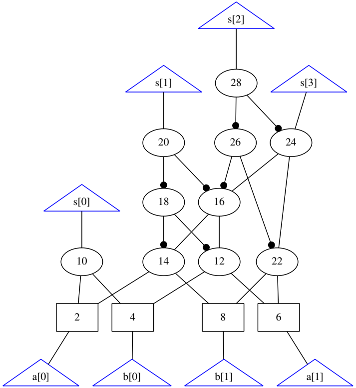
And-Inverter Graph

Multiplier Circuits

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

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

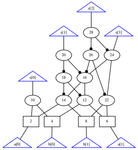
$$(2a_1 + a_0) * (2b_1 + b_0) = 8s_3 + 4s_2 + 2s_1 + s_0?$$



And-Inverter Graph

Formal Verification

System



Specification

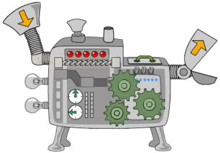
For all $a_i, b_i \in \mathbb{B}$:

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

Mathematical Model

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

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

Satisfiability Checking (SAT)

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

  [BiereFallerFazekasFleuryFroleyksPollitt SATComp'24]

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

Satisfiability Checking (SAT)

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

Theorem Proving

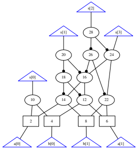
- Used in industry, e.g., ACL2 [TemelSlobodovaHunt CAV'20]
- Automated on the RTL level [Temel TACAS'24]

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

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

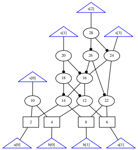
Implication

$= 0$ ✓

$\neq 0$ ✗

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

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

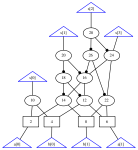
Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Multiplier



Polynomials

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

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

$= 0$ ✓

$\neq 0$ ✗

Multiplier Specification

Unsigned Integers:

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

Multiplier Specification

Unsigned Integers:

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

Multiplier Specification

Unsigned Integers:

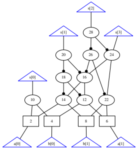
$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right) \quad \in \mathbb{Z}[X]$$

Signed Integers:

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

Basic Idea of Algebraic Approach

Multiplier



Polynomials

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

Specification

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

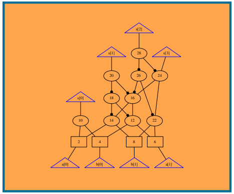
Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

Basic Idea of Algebraic Approach

Multiplier



Polynomials

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

Specification

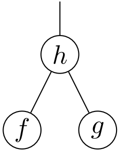
$$\sum_{i=0}^{2n-1} 2^i s_i - \left(\sum_{i=0}^{n-1} 2^i a_i\right)\left(\sum_{i=0}^{n-1} 2^i b_i\right)$$

Ideal Membership

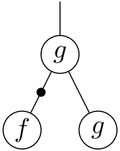
$= 0$ ✓

$\neq 0$ ✗

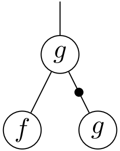
From AIGs to Polynomials



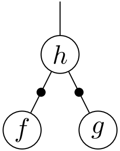
$$h = f \wedge g$$



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

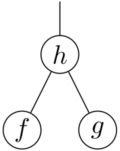


$$h = f \wedge \neg g$$



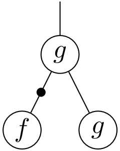
$$h = \neg f \wedge \neg g$$

From AIGs to Polynomials



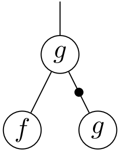
$h = f \land g$

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



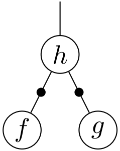
$h = \neg f \land g$

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



$h = f \land \neg g$

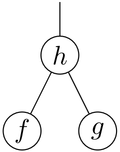
f	g	h
0	0	0
0	1	0
1	0	1
1	1	0



$h = \neg f \land \neg g$

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

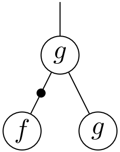
From AIGs to Polynomials



$h = f \wedge g$

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

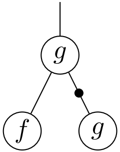
$-h + fg$



$h = \neg f \wedge g$

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

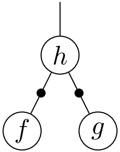
$-h - fg + g$



$h = f \wedge \neg g$

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

$-h - fg + f$



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

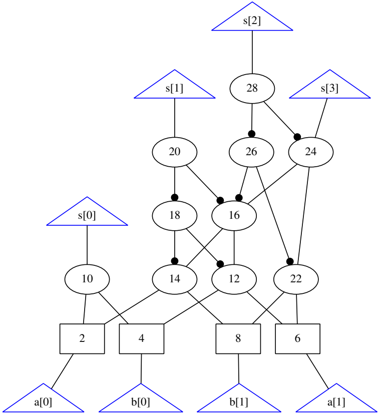
f	g	h
0	0	1
0	1	0
1	0	0
1	1	0

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

From AIGs to Polynomials

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

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



From AIGs to Polynomials

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

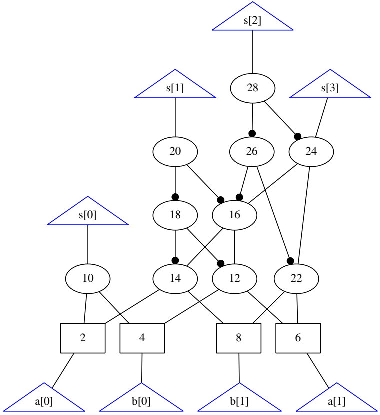
$$\begin{array}{ll} -s_3 + l_{24} & -l_{22} + a_1 b_1 \\ -s_2 + l_{28} & -l_{20} + l_{18} l_{16} - l_{18} - l_{16} + 1 \\ -s_1 + l_{20} & -l_{18} + l_{14} l_{12} - l_{14} - l_{12} + 1 \\ -s_0 + l_{10} & -l_{16} + l_{14} l_{12} \\ -l_{28} + l_{26} l_{24} - l_{26} - l_{24} + 1 & -l_{14} + a_0 b_1 \\ -l_{26}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

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

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

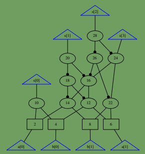
Specification $\mathcal{S}_n \in \mathbb{Z}[X]$.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4b_1a_1 - 2b_1a_0 - 2b_0a_1 - b_0a_0$$



Basic Idea of Algebraic Approach

Multiplier



Polynomials

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

Specification

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

Ideal Membership

$= 0$ ✓

$\neq 0$ ✗

COMPUTER ALGEBRA

COMPUTER ALGEBRA

IDEALS AND GRÖBNER BASES

Ideal

Ideal. A subset $I \subset R[X]$ is an ideal if it satisfies:

- $0 \in I$
- If $f, g \in I$, then $f + g \in I$.
- If $f \in I$ and $h \in R[X]$ then $hf \in I$.

Ideal

Ideal. A subset $I \subset R[X]$ is an ideal if it satisfies:

- $0 \in I$
- If $f, g \in I$, then $f + g \in I$.
- If $f \in I$ and $h \in R[X]$ then $hf \in I$.

Ideal generated by a finite number of polynomials.
Let $f_1, \ldots, f_s \in R[X]$. Then we set

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

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

Hilbert Basis Theorem. Every ideal has a finite basis.

Ideal

The ideal $\langle f_1, \dots, f_s \rangle$ has a nice interpretation in terms of polynomial equations.

Given $f_1, \dots, f_s \in R[X]$, we get the system of equations

$$f_1 = 0,$$
$$\vdots$$
$$f_s = 0.$$

Let $h_1, \dots, h_s \in R[X]$. We can derive $h_1 f_1 = 0$, $h_2 f_2 = 0$, $h_1 f_1 + h_2 f_2 = 0$ etc.

Hence we obtain $h_1 f_1 + \cdots + h_s f_s = 0$ as a consequence of our initial system.

Thus, we can think of $\langle f_1, \dots, f_s \rangle$ as consisting of all "polynomial consequences" of the equations $f_1 = f_2 = \dots = f_s = 0$.

Applications of Ideals

The Ideal Membership Problem.

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

Applications of Ideals

The Ideal Membership Problem.

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

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

Orderings on the Monomials

Univariate Polynomials - Sort by Degree.

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

Example: $7x^5 + 5x^4 - 2x^3 + x - 6$

Orderings on the Monomials

Univariate Polynomials - Sort by Degree.

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

Example: $7x^5 + 5x^4 - 2x^3 + x - 6$

Linear Polynomials - Sort by variables.

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

Example: $8x + 3y - 3z$ for $x > y > z$

Orderings on the Monomials

Univariate Polynomials - Sort by Degree.

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

Example: $7x^5 + 5x^4 - 2x^3 + x - 6$

Linear Polynomials - Sort by variables.

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

Example: $8x + 3y - 3z$ for $x > y > z$

How to order non-linear multivariate polynomials in $R[X]$?

Orderings on the Monomials

Requirements:

Orderings on the Monomials

Requirements:

- Total Ordering: $\forall \sigma_1, \sigma_2 :$ $\quad \sigma_1 < \sigma_2$ or $\sigma_1 = \sigma_2$ or $\sigma_2 < \sigma_1$

Orderings on the Monomials

Requirements:

- Total Ordering: $\forall \sigma_1, \sigma_2 :$ $\quad \sigma_1 < \sigma_2$ or $\sigma_1 = \sigma_2$ or $\sigma_2 < \sigma_1$
- Multiplication: For any monomials σ_1, σ_2, τ, we require that $\sigma_1 < \sigma_2 \implies \sigma_1\tau < \sigma_2\tau$.

Orderings on the Monomials

Requirements:

- Total Ordering: $\forall \sigma_1, \sigma_2 : \qquad \sigma_1 < \sigma_2 \quad \text{or} \quad \sigma_1 = \sigma_2 \quad \text{or} \quad \sigma_2 < \sigma_1$
- Multiplication: For any monomials σ_1, σ_2, τ, we require that $\sigma_1 < \sigma_2 \implies \sigma_1\tau < \sigma_2\tau$.
- Well-Ordering: Every nonempty subset of monomials has a smallest element.

Orderings on the Monomials

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

Orderings on the Monomials

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

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

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

Orderings on the Monomials

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

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 $u_i < v_i$.

Degree Reverse Lexicographic Order $\prec_{\text{drl}}$.

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

Orderings on the Monomials

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

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 $u_i < v_i$.

Degree Reverse Lexicographic Order $\prec_{\text{drl}}$.

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

Example: Let $f = 5xy^3 + 4x^2y - 3xy + 2x^2 \in \mathbb{Q}[x, y]$
Ordered according to $\prec_{\text{lex}}$ for $x > y$: $f = 4x^2y + 2x^2 + 5xy^3 - 3xy$
Ordered according to $\prec_{\text{drl}}$ for $x > y$: $f = 5xy^3 + 4x^2y - 3xy + 2x^2$

Leading Elements

Let f in $R[X]$ be ordered w.r.t to an ordering $<$ such that

$$f = a_1\tau_1 + a_2\tau_2 + \ldots + a_m\tau_m.$$

Then we call

- $\mathrm{lt}(f) = a_1\tau_1$ is the **leading term** of f.
- $\mathrm{lm}(f) = \tau_1$ is the **leading monomial** of f.
- $\mathrm{lc}(f) = a_1$ is the **leading coefficient** of f.
- $f - \mathrm{lt}(f) = a_2\tau_2 + \ldots + a_m\tau_m$ is the **tail** of f.

Ideal Membership Problem

Let $I = \langle x^2 - \frac{3}{4}y, 2x^2 - 3 \rangle \subset \mathbb{Q}[x, y]$.
Is the polynomial $f = 10x^3y - x^2y + x^2 - 15xy + \frac{3}{4}y^2 - \frac{3}{4}y \in I$?

Ideal Membership Problem

Let $I = \langle x^2 - \frac{3}{4}y, 2x^2 - 3\rangle \subset \mathbb{Q}[x, y]$.
Is the polynomial $f = 10x^3y - x^2y + x^2 - 15xy + \frac{3}{4}y^2 - \frac{3}{4}y \in I$?

Spoiler: Yes, because

$$(1 - y)(x^2 - \frac{3}{4}y) + (5xy)(2x^2 - 3) = 10x^3y - x^2y + x^2 - 15xy + \frac{3}{4}y^2 - \frac{3}{4}y$$

Ideal Membership Problem

Let $I = \langle x^2 - \frac{3}{4}y, 2x^2 - 3 \rangle \subset \mathbb{Q}[x, y]$.
Is the polynomial $f = 10x^3y - x^2y + x^2 - 15xy + \frac{3}{4}y^2 - \frac{3}{4}y \in I$?

$$10x^3y - x^2y + x^2 - 15xy + \frac{3}{4}y^2 - \frac{3}{4}y \xrightarrow{x^2 - \frac{3}{4}y} \frac{15}{2}xy^2 - 15xy$$

$$10x^3y - x^2y + x^2 - 15xy + \frac{3}{4}y^2 - \frac{3}{4}y \xrightarrow{2x^2 - 3} \frac{3}{4}y^2 - \frac{9}{4}y + \frac{3}{2}$$

Operation $\xrightarrow{p}$ is multivariate variant of polynomial division.

Gröbner Bases - The Idea

Given a set of polynomials F in $R[X]$.

- Transform F into another set $G \subset R[X]$ **with certain nice properties** such that $\langle F \rangle = \langle G \rangle$.

Gröbner Bases - The Idea

Given a set of polynomials F in $R[X]$.

- Transform F into another set $G \subset R[X]$ **with certain nice properties** such that $\langle F \rangle = \langle G \rangle$.
- A whole range of problems defined for an arbitrary set of polynomials F becomes algorithmically solvable using G.

Gröbner Bases - The Idea

Given a set of polynomials F in $R[X]$.

- Transform F into another set $G \subset R[X]$ **with certain nice properties** such that $\langle F \rangle = \langle G \rangle$.
- A whole range of problems defined for an arbitrary set of polynomials F becomes algorithmically solvable using G.
- G is called a **Gröbner basis** [Buchberger'65].

Properties of Gröbner Bases

Lemma 1. Every ideal $I \subseteq R[X]$ has a Gröbner basis w.r.t. a fixed term order.

Lemma 2. If $G \subset R[X]$ is a Gröbner basis, then every $f \in R[X]$ has a unique remainder $r \in R[X]$ with respect to G such that no term in r is divisible by any of $\mathrm{lt}(g_i)$.

Furthermore, $f - r \in \langle G \rangle$.

In particular, r is the remainder on division of f by G no matter how the elements of G are listed when using the division algorithm.

Lemma 3. Let $G \subseteq R[X]$ be a Gröbner basis, and let $f \in R[X]$. Then $f \in \langle G \rangle$ iff the remainder of f with respect to G is zero.

Computing a Gröbner Basis

Algorithm: Buchberger's Algorithm

Input : $F = \{f_1, \ldots, f_s\}$, monomial ordering $<$
Output: Gröbner basis $G = \{g_1, \ldots, g_t\}$ w.r.t. $<$, such that $\langle f_1, \ldots, f_s \rangle = \langle g_1, \ldots, g_t \rangle$

$G = F$;
$C = \{\{g_1, g_2\} \mid g_1, g_2 \in G, g_1 \neq g_2\}$;
while not all pairs $\{g_1, g_2\} \in C$ are marked **do**
    choose unmarked pair $\{g_1, g_2\}$;
    mark $\{g_1, g_2\}$;
    $h =$ normalform of $\mathrm{spol}(g_1, g_2)$ w.r.t. G $(\mathrm{spol}(g_1, g_2) \xrightarrow{G} h)$;
    if $h \neq 0$ **then**
        $C = C \cup \{\{g, h\} \mid g \in G\}$;
        $G = G \cup \{h\}$;
return G

Product Criterion. If $p, q \in k[x_1, \ldots, x_n] \setminus \{0\}$ are such that the leading monomials are coprime, i.e., $\mathrm{lcm}(\mathrm{lm}(p), \mathrm{lm}(q)) = \mathrm{lm}(p)\,\mathrm{lm}(q)$, then $\mathrm{spol}(p, q)$ reduces to zero mod $\{p, q\}$.

Ideal Membership Problem II

Let $I = \langle x^2 - \frac{3}{4}y, 2x^2 - 3\rangle \subset \mathbb{Q}[x,y]$.
Is the polynomial $f = 10x^3y - x^2y + x^2 - 15xy + \frac{3}{4}y^2 - \frac{3}{4}y \in I$?

1. Calculate a Gröbner basis G of I:
Let $f_1 = x^2 - \frac{3}{4}y$, $f_2 = 2x^2 - 3$. We order terms lexicographic with $x > y$.

$$\mathrm{spol}(f_1, f_2) = 2f_1 - f_2 = -\frac{6}{4}y + 3 \to \mathbf{y - 2} =: \mathbf{f_3}$$

$$\mathrm{spol}(f_1, f_3) = yf_1 - x^2 f_3 = 2x^2 - \frac{3}{4}y^2 \xrightarrow{f_1} \frac{3}{4}y^2 - \frac{6}{4}y \xrightarrow{f_3} 0$$

$$\mathrm{spol}(f_2, f_3) = yf_2 - 2x^2 f_3 = 4x^2 - 3y \xrightarrow{f_1} 0$$

For $\mathrm{spol}(f_1, f_3)$ and $\mathrm{spol}(f_2, f_3)$ we could also make use of the product criterion.

Gröbner(f_1, f_2) = $G = \{x^2 - \frac{3}{4}y, 2x^2 - 3, y - 2\}$

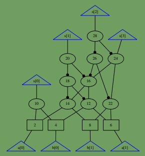
2. Calculate the remainder r of dividing f by G :

$$10x^3y - x^2y + x^2 - 15xy + \frac{3}{4}y^2 - \frac{3}{4}y \xrightarrow{2x^2-3} \frac{3}{4}y^2 - \frac{9}{4}y + \frac{3}{2} \xrightarrow{y-2} 0$$

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

[RitircBiereKauers FMCAD'17]

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

Ideal Membership Problem

[RitircBiereKauers FMCAD'17]

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.
- Lexicographic term order: Output variable of a gate is greater than input variables.

Ideal Membership Problem

[RitircBiereKauers FMCAD'17]

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.
- Lexicographic term order: Output variable of a gate is greater than input variables.

Theorem

$G(C) \cup B(C)$ *is a Gröbner basis for* $J(C)$.

Proof idea: Application of Buchberger's Product criterion.

Ideal Membership Problem

[RitircBiereKauers FMCAD'17]

- Polynomial Encoding:
  - Gate polynomials $G(C)$
  - Boolean input constraints $B(C)$
- Let $J(C) = \langle G(C) \cup B(C) \rangle$.
- Lexicographic term order: Output variable of a gate is greater than input variables.

Theorem

$G(C) \cup B(C)$ is a Gröbner basis for $J(C)$.

Proof idea: Application of Buchberger's Product criterion.

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

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

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

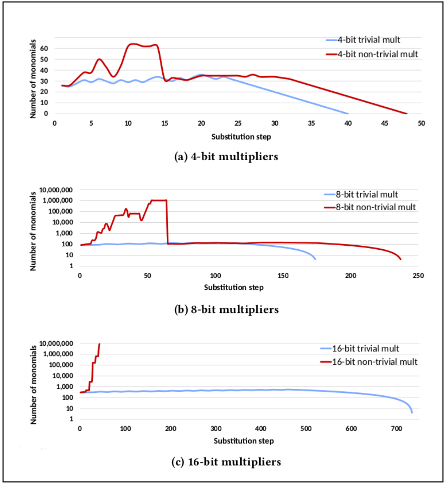
until no further reduction is possible, then C is a multiplier iff remainder is zero.

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**.

Change of Order[1]

	$\prec_{\text{lex}}$	$\prec_{\text{drl}}$
GB Computation	✓ Easy	⚠ Hard
Spec Reduction	⚠ Hard	✓ Easy

[1] D. Kaufmann and J. Berthomieu. Extracting Linear Relations from Gröbner Bases for Formal Verification of And-Inverter Graphs. Submitted. Preprint at https://arxiv.org/abs/2411.16348

Change of Order[1]

	$\prec_{\text{lex}}$	$\prec_{\text{drl}}$
GB Computation	✔ Easy	⚠ Hard
Spec Reduction	⚠ Hard	✔ Easy

If the specification polynomial is linear,

a Gröbner basis with respect to a
degree reverse lexicographic term ordering

contains linear polynomials that suffice
to derive correctness of the circuit.

[1] D. Kaufmann and J. Berthomieu. Extracting Linear Relations from Gröbner Bases for Formal Verification of And-Inverter Graphs. Submitted. Preprint at https://arxiv.org/abs/2411.16348

Theorem

Theorem

Let $p \in \mathbb{K}[X]$ *with* $\deg(p) = 1$, $I \subseteq \mathbb{K}[X]$ *be an ideal. Let* G *be a Gröbner basis of* I *with respect to* $\prec_{\mathrm{drl}}$ *and let* $G_1 = \{g \in G \mid \deg(g) \leq 1\}$*. We have* $p \in I$ *if and only if* $p \to_{G_1} 0$*. In particular,* $p = \alpha_1 g_1 + \cdots + \alpha_m g_m$ *with* $g_i \in G_1$, $\alpha_i \in \mathbb{K}$*.*

Linear Gröbner Basis Reduction Algorithm

Algorithm: Linear Gröbner basis reduction

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\text{init}} \leftarrow$ Gate-Polynomials$(C) \cup$ Boolean-Input-Polynomials(C);

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize($\mathcal{S}$);

$G_{\text{drl}} \leftarrow$ Compute-$\prec_{\text{drl}}$-Gröbner-Basis($G_{\text{init}} \cup G_{\text{ext}}$)

$G_1 \leftarrow \{g \mid g \in G_{\text{drl}} \wedge \deg(g) \leq 1\}$;

while $\text{lm}(\mathcal{S}_{\text{lin}}) \in \{\text{lm}(g) | g \in G_1\}$ **do**

    $p_{\text{lin}} \leftarrow g \in G_1$ such that $\text{lm}(g) = \text{lm}(\mathcal{S}_{\text{lin}})$;

    if $\nexists p_{\text{lin}}$ **then return** $\perp$;

    $\mathcal{S}_{\text{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\text{lin}}, p_{\text{lin}}$);

return $\mathcal{S}_{\text{lin}} = 0$

Linear Gröbner Basis Reduction Algorithm

Algorithm: Linear Gröbner basis reduction

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\text{init}} \leftarrow$ Gate-Polynomials$(C) \cup$ Boolean-Input-Polynomials(C);

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize($\mathcal{S}$);

$G_{\text{drl}} \leftarrow$ Compute-$\prec_{\text{drl}}$-Gröbner-Basis($G_{\text{init}} \cup G_{\text{ext}}$)

$G_1 \leftarrow \{g \mid g \in G_{\text{drl}} \wedge \deg(g) \leq 1\}$;

while $\text{lm}(\mathcal{S}_{\text{lin}}) \in \{\text{lm}(g) | g \in G_1\}$ **do**

  $p_{\text{lin}} \leftarrow g \in G_1$ such that $\text{lm}(g) = \text{lm}(\mathcal{S}_{\text{lin}})$;

  if $\nexists p_{\text{lin}}$ **then return** $\perp$;

  $\mathcal{S}_{\text{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\text{lin}}, p_{\text{lin}}$);

return $\mathcal{S}_{\text{lin}} = 0$

Linear Gröbner Basis Reduction Algorithm

Algorithm: Linear Gröbner basis reduction

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\text{init}} \leftarrow$ Gate-Polynomials$(C) \cup$ Boolean-Input-Polynomials(C);

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize($\mathcal{S}$);

$G_{\text{drl}} \leftarrow$ Compute-$\prec_{\text{drl}}$-Gröbner-Basis($G_{\text{init}} \cup G_{\text{ext}}$)

$G_1 \leftarrow \{g \mid g \in G_{\text{drl}} \wedge \deg(g) \leq 1\}$;

while $\text{lm}(\mathcal{S}_{\text{lin}}) \in \{\text{lm}(g) | g \in G_1\}$ **do**

  $p_{\text{lin}} \leftarrow g \in G_1$ such that $\text{lm}(g) = \text{lm}(\mathcal{S}_{\text{lin}})$;

  if $\nexists p_{\text{lin}}$ **then return** $\perp$;

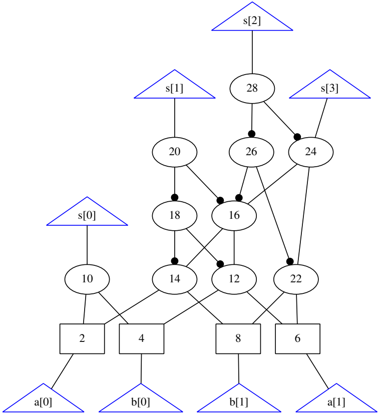
  $\mathcal{S}_{\text{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\text{lin}}, p_{\text{lin}}$);

return $\mathcal{S}_{\text{lin}} = 0$

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize($\mathcal{S}$)

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

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



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

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4b_1a_1 - 2b_1a_0 - 2b_0a_1 - b_0a_0$$

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize($\mathcal{S}$)

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

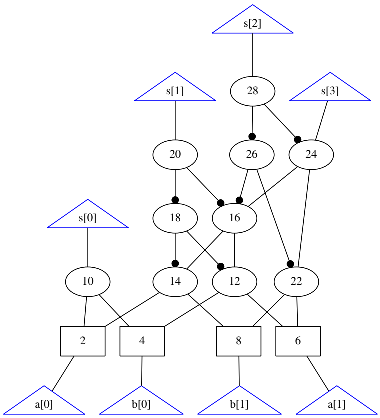
$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14}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

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

$$-t_{11} + a_1 b_1 \qquad -t_{01} + a_0 b_1$$
$$-t_{10} + a_1 b_0 \qquad -t_{00} + a_0 b_0$$

Linear Specification $\mathcal{S}_{\text{lin}} \in \mathbb{Z}[X]$.

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



$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ **Linearize($\mathcal{S}$)**

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

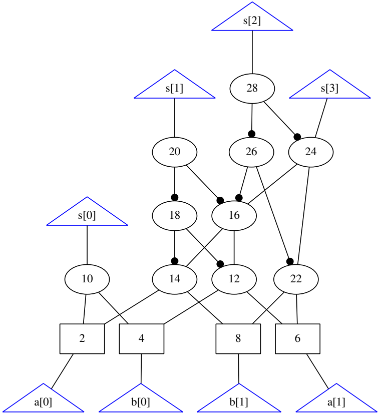
$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1 b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14}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

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

$$-t_{11} + a_1 b_1 \qquad -t_{01} + a_0 b_1$$
$$-t_{10} + a_1 b_0 \qquad -t_{00} + a_0 b_0$$

Linear Specification $\mathcal{S}_{\text{lin}} \in \mathbb{Z}[X]$.

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



$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize($\mathcal{S}$)

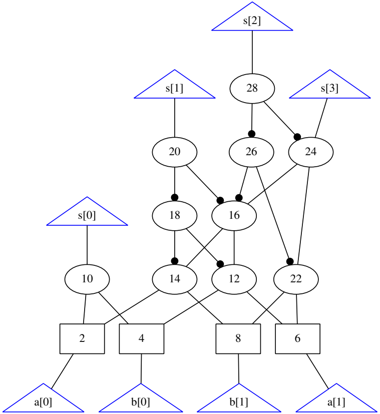
Gate polynomials $G(C) \subseteq \mathbb{Z}[X]$.

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

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

Linear Specification $\mathcal{S}_{\text{lin}} \in \mathbb{Z}[X]$.

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



Linear Gröbner Basis Reduction Algorithm

Algorithm: Linear Gröbner basis reduction

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\mathrm{init}} \leftarrow$ Gate-Polynomials$(C) \cup$ Boolean-Input-Polynomials(C);

$\mathcal{S}_{\mathrm{lin}}, G_{\mathrm{ext}} \leftarrow$ Linearize($\mathcal{S}$);

$G_{\mathrm{drl}} \leftarrow$ Compute-$\prec_{\mathrm{drl}}$-Gröbner-Basis($G_{\mathrm{init}} \cup G_{\mathrm{ext}}$)

$G_1 \leftarrow \{g \mid g \in G_{\mathrm{drl}} \wedge \deg(g) \leq 1\}$;

while $\mathrm{lm}(\mathcal{S}_{\mathrm{lin}}) \in \{\mathrm{lm}(g) | g \in G_1\}$ **do**

  $p_{\mathrm{lin}} \leftarrow g \in G_1$ such that $\mathrm{lm}(g) = \mathrm{lm}(\mathcal{S}_{\mathrm{lin}})$;

  if $\nexists p_{\mathrm{lin}}$ **then return** $\perp$;

  $\mathcal{S}_{\mathrm{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\mathrm{lin}}, p_{\mathrm{lin}}$);

return $\mathcal{S}_{\mathrm{lin}} = 0$

Linear Gröbner Basis Reduction Algorithm

Algorithm: Linear Gröbner basis reduction

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\text{init}} \leftarrow$ Gate-Polynomials$(C) \cup$ Boolean-Input-Polynomials(C);

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize($\mathcal{S}$);

$G_{\text{drl}} \leftarrow$ Compute-$\prec_{\text{drl}}$-Gröbner-Basis($G_{\text{init}} \cup G_{\text{ext}}$);

$G_1 \leftarrow \{g \mid g \in G_{\text{drl}} \wedge \deg(g) \leq 1\}$;

while $\operatorname{lm}(\mathcal{S}_{\text{lin}}) \in \{\operatorname{lm}(g) | g \in G_1\}$ **do**

  $p_{\text{lin}} \leftarrow g \in G_1$ such that $\operatorname{lm}(g) = \operatorname{lm}(\mathcal{S}_{\text{lin}})$;

  if $\nexists p_{\text{lin}}$ **then return** $\perp$;

  $\mathcal{S}_{\text{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\text{lin}}, p_{\text{lin}}$);

return $\mathcal{S}_{\text{lin}} = 0$

Linear Gröbner Basis Reduction Algorithm

Algorithm: Linear Gröbner basis reduction

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\text{init}} \leftarrow$ Gate-Polynomials$(C) \cup$ Boolean-Input-Polynomials(C);

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize($\mathcal{S}$);

$G_{\text{drl}} \leftarrow$ Compute-$\prec_{\text{drl}}$-Gröbner-Basis$(G_{\text{init}} \cup G_{\text{ext}})$; `// Double exponential`

$G_1 \leftarrow \{g \mid g \in G_{\text{drl}} \wedge \deg(g) \leq 1\}$;

while $\text{lm}(\mathcal{S}_{\text{lin}}) \in \{\text{lm}(g) | g \in G_1\}$ **do**

  $p_{\text{lin}} \leftarrow g \in G_1$ such that $\text{lm}(g) = \text{lm}(\mathcal{S}_{\text{lin}})$;

  if $\nexists p_{\text{lin}}$ **then return** $\perp$;

  $\mathcal{S}_{\text{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\text{lin}}, p_{\text{lin}}$);

return $\mathcal{S}_{\text{lin}} = 0$

Linear Gröbner Basis Reduction Algorithm

Algorithm: Linear Gröbner basis reduction

Input : Circuit C in AIG format, Specification polynomial $\mathcal{S}$

Output: Determine whether C fulfills the specification

$G_{\text{init}} \leftarrow$ Gate-Polynomials$(C) \cup$ Boolean-Input-Polynomials(C);

$\mathcal{S}_{\text{lin}}, G_{\text{ext}} \leftarrow$ Linearize($\mathcal{S}$);

Preprocessing(G_{ext});

while $\text{lm}(\mathcal{S}_{\text{lin}}) \in \{\text{lm}(g) | g \in G\}$ **do**

  $p \leftarrow g \in G$ such that $\text{lm}(g) = \text{lm}(\mathcal{S}_{\text{lin}})$;

  $p_{\text{lin}} \leftarrow$ Linearize-Single-Polynomial(p, G) ; // On-the-fly

  if p_{lin} = 0 **then return** $\perp$;

  $\mathcal{S}_{\text{lin}} \leftarrow$ Linear-Reduce($\mathcal{S}_{\text{lin}}, p_{\text{lin}}$);

return $\mathcal{S}_{\text{lin}} = 0$

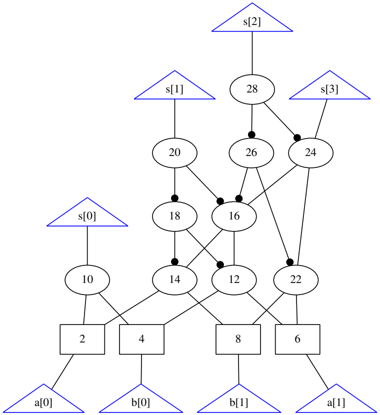
On-the-fly Linearization

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

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

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

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



On-the-fly Linearization

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

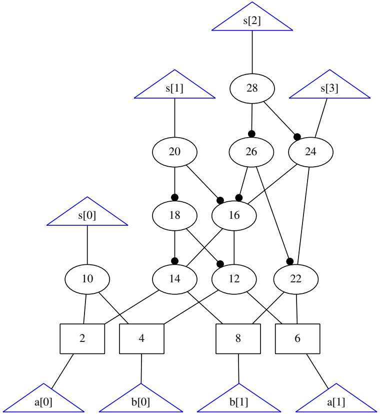
$-s_3 + l_{24}$
$-s_2 + l_{28}$
$-s_1 + l_{20}$
$-s_0 + l_{10}$
$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$
$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$
$-l_{24} + l_{22}l_{16}$

$-l_{22} + a_1b_1$
$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$
$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$
$-l_{16} + l_{14}l_{12}$
$-l_{14} + a_0b_1$
$-l_{12} + a_1b_0$
$-l_{10} + a_0b_0$

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

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

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



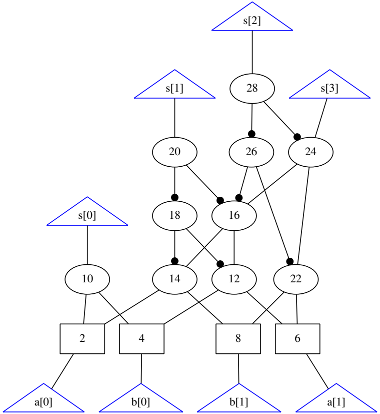
On-the-fly Linearization

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

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

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

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



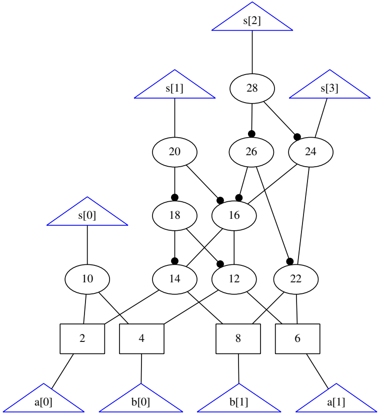
On-the-fly Linearization

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

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

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

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



On-the-fly Linearization

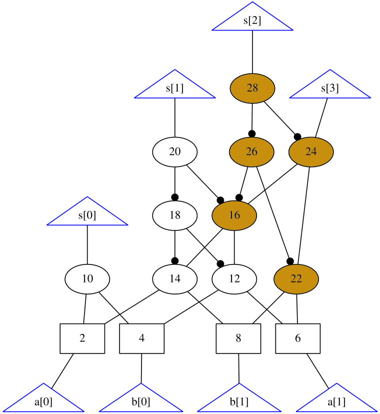
Gate polynomials $G(C) \subseteq \mathbb{Z}[X]$.

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

$-l_{22} + a_1b_1$
$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$
$-l_{18} + l_{14}l_{12} - l_{14} - l_{12} + 1$
$-l_{16} + l_{14}l_{12}$
$-l_{14} + a_0b_1$
$-l_{12} + a_1b_0$
$-l_{10} + a_0b_0$

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

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



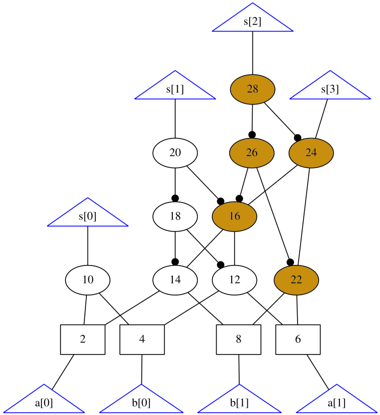
On-the-fly Linearization

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

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

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

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



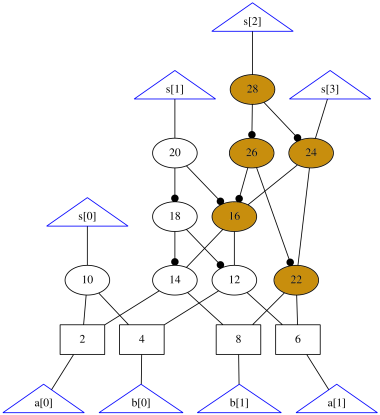
On-the-fly Linearization

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

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

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

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



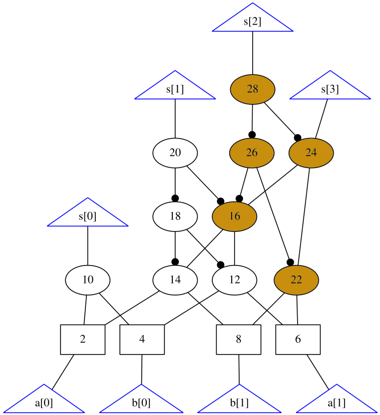
On-the-fly Linearization

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

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

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

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



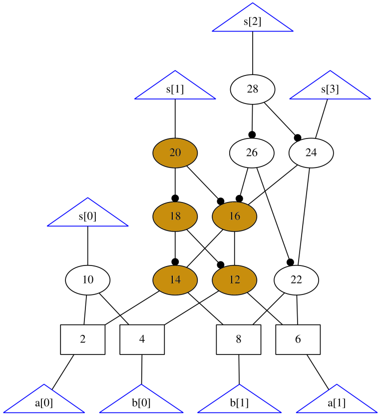
On-the-fly Linearization

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

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

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

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



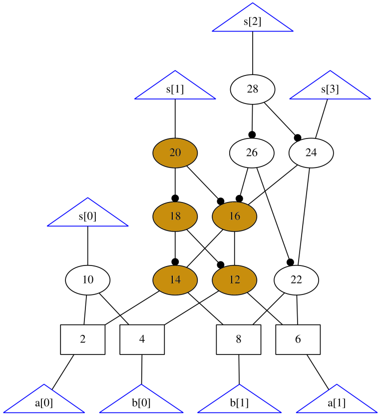
On-the-fly Linearization

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

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

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

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



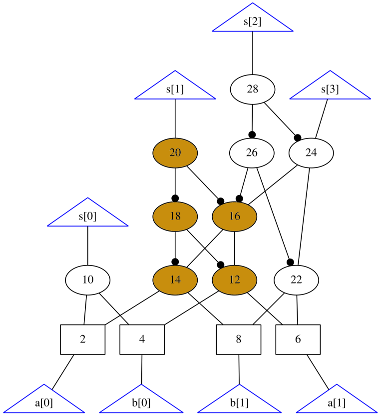
On-the-fly Linearization

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

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

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

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



MultiLing

- Builds on AMulet 2.2, written in C++
- Variables are sorted based on minimum distance to primary inputs
- Gröbner basis engine: msolve[2]
- Non-linear rewriting as fall-back, when distance is below 6.

[2] J. Berthomieu, C. Eder, and M. Safey El Din. msolve: A Library for Solving Polynomial Systems. ISSAC, 2021

Evaluation - Optimized Multipliers

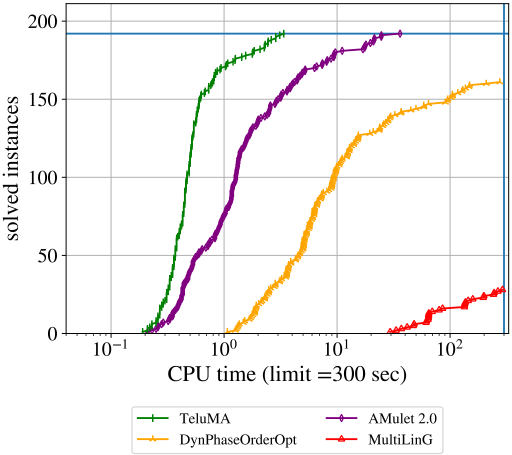
ABC-benchmarks			Related work			MultiLinG		
n	Optimization	Nodes	TeluMA	AMulet 2.2	DPOO	Time	PP-Nodes	#GB
64	resyn	32064	0.3	TO	1.0	5.6	7996	10
64	resyn3	32064	0.3	0.2	1.0	5.6	8000	0
64	dc2	32064	0.2	0.3	1.0	5.8	8000	0
64	complex[3]	32063	TO	TO	1.0	6.3	7996	9
128	resyn	129664	1.3	TO	5.7	200.6	32380	10
128	resyn3	129664	1.2	TO	7.7	209.3	32384	0
128	dc2	129664	1.1	TO	6.6	214.6	32384	0
128	complex	129663	TO	TO	5.8	214.1	32380	9

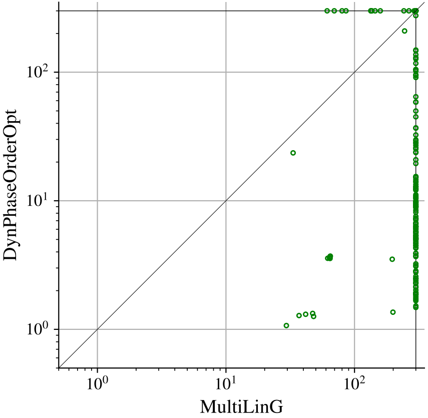
time in sec, TO = 1200 sec, DPOO = DynPhaseOrderOpt

[3] `-c "logic; mfs2 -W 20; ps; mfs; st; ps; dc2 -l; ps; resub -l -K 16 -N 3 -w 100; ps; logic; mfs2 -W 20; ps; mfs; st; ps; iresyn -l; ps; resyn; ps; resyn2; ps; resyn3; ps; dc2 -l; ps;"`

Evaluation - 64-bit Multipliers

Verification of 192 unsigned 64-bit multipliers





Conclusion

If the specification polynomial is linear,

a Gröbner basis with respect to a
degree reverse lexicographic term ordering

contains linear polynomials that suffice
to derive correctness of the circuit.

- Full Gröbner basis computation is hard
- Our approach linearizes polynomials on the fly
- Robust on optimized benchmarks and complements existing $\prec_{\text{lex}}$ techniques.

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.

References II

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

[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.

References III

[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.

[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.

References IV

[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.

[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.

References V

[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.