### COMBINING SAT AND COMPUTER ALGEBRA FOR CIRCUIT VERIFICATION

### Daniela Kaufmann

Johannes Kepler University

Linz, Austria

#### **Beyond Satisfiability Workshop**

Simons Institute for the Theory of Computing (online)

February 16, 2021











	Basic M	ode 🗸		-	- 0 😣	
41958	35+31	45727	_		3338	
1,333	82044	9			1,5550	
7	8	9	+	$\sim$	c	
4	5	6	×	(		
1	2	3	-	x <sup>2</sup>	1	
					-	

					Unt	itled 1	- LibreOf	fice Cal	k		-	D	
File	Edit	View	Inser	t For	mat	Style	s Sheet	Data	Tools	Win	dow Hel	ρ	
	• 🚞	- 4	- 1		9	×	6	- 1	A .	-	• († -	9¢	
U	iberatio	ns •	1	0	٣	в	<u>Ι</u>	<u>A</u> .	ø -	=	= =	=,	
A2			- 1	ľxΣ	• =							•	6
		Α			B			С			D	1	
1	1.33	38204	1491									_	
													1
3													
4												- h	
5												- 15	1
6												- 1	
7												- 17	
8													
8 9													



File Edit V	iew Insert Fo	rmat Styles She	et Data Tool	s Window Hel	0
🖻 • 🚞 •	🛛 - 📓 🖨		🖻 • 🎍 A	y 🐀 • 🕐 -	94
Liberation S	• • 10	• B I L	<u>A</u> - 🖉 -		<b>E</b> 2
AZ	- f <sub>x</sub> Σ	• =			- 6
	A	В	С	D	1
1 1.3338	3204491				
2					4
3					
4					- 112
5					1.4
6					- f
7					10.1
8					
9					
10					
H A P H	+ Sheet1				1







Pentium FDIV bug and AMI GUI BIOS demo https://youtu.be/hE7qMJV115U



Pentium FDIV bug and AMI GUI BIOS demo https://youtu.be/hE7qMJV115U

#### **Intel Pentium Processor**



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

<b>GPU-World</b>	News • CPUs / Chips • Benchmarks • Information • F Identification • Pinouts • S-Spec numbers • Glossary	orum • Links • .	About		
			MARCH (Po #ORL)		
	Unable to connect		Search the site / Identify CPU / Quick CPU Innings		
			NEW BE		
	SX923 (Intel Pentium 90 MHz)				
II S-Specs > Pentium > SX923					
SX923 specifications	CPU / Microprocessor	<ul> <li>Millio and</li> <li>Marcal 1 S</li> <li>Marcal 2 S</li> <li>Marcal 2 S</li> <li>Marcal 2 S</li> </ul>	MARTIN Context Martina		
Part number	A80502-90				
Frequency (MHz) () Bus speed (MHz) () Package type	90 60 SPGA	Unable	to connect		
Architecture / Microarchited	ture / Other	the server at	er all		
CPUID Core stepping Core voltage (V)  Notes on sSpec SX923 This processor has FDIV bug.	0522h 83 3.3 (3.135 - 3.465)	<ul> <li>The alle crossels affect in</li> <li>E provide</li> </ul>	could be temperatify the or temperatify the or tem body. By again to stream. • available to load any pages.		
in the second					

https://www.cpu-world.com/sspec/SX/SX923.html

### Intel Pentium FDIV bug 1994



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 25 years later, verification of arithmetic circuits is considered to be hard. Especially proving the correctness of gate-level integer multiplier circuits is a challenge.



$$3 \cdot 3 = 9$$

$$3 \cdot 3 = 9$$



$$3 \cdot 3 = 9$$



$$3 \cdot 3 = 9$$



$$3 \cdot 3 = 9$$



$$3 \cdot 3 = 9$$



$$3 \cdot 3 = 9$$





$$3 \cdot 3 = 9$$









XOR-Gate fg ⊥⊥









 $f \wedge g = y$ 





XOR-Gate ∫g















XOR-Gate









fg

y



 $f \wedge g = y$ 

gy







**XOR-Gate** fg

y













 $f \oplus g = y$ g

y







### Is the circuit correct?

### **Multiplier circuit**





### **Multiplier circuit**

#### Circuit seems correct!



Is the circuit really correct?

#### **Multiplier Circuits**

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

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

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



### Testing

$a_1a_0$	·	$b_1b_0$
00	•	00
00		01
00		10
00		11
01		00
01		01
01		10
01		11
10		00
10		01
10		10
10		11
11		00
11		01
11		10
11		11



### Testing

	Cases	Bit-width
_	16	2
	64	3
	256	4
	1024	5
	4 096	6
	16 384	7
	65 536	8
	:	:
quintillion	18446744073709551616	32
	:	:
undecillion	340282366920938463463374607431768211456	64

### **Formal Verification**


#### Satisfiability Checking (SAT)

#### SAT 2016 Competition

[Biere SATComp'16]

Exponential run-time of solvers

#### Satisfiability Checking (SAT)

SAT 2016 Competition

[Biere SATComp'16]

Exponential run-time of solvers

#### **Decision Diagrams**

- First technique to detect Pentium bug [ChenBryant DAC'95]
- Cannot be applied fully automatically

#### Satisfiability Checking (SAT)

SAT 2016 Competition

[Biere SATComp'16]

Exponential run-time of solvers

#### **Decision Diagrams**

- First technique to detect Pentium bug [ChenBryant DAC'95]
- Cannot be applied fully automatically

#### **Theorem Proving**

- Used in industry
- Past: Requires manual effort
- 2020: Progress in ACL2

[TemelSlobodovaHunt CAV'20]

#### Satisfiability Checking (SAT)

#### SAT 2016 Competition

[Biere SATComp'16]

Exponential run-time of solvers

#### **Decision Diagrams**

- First technique to detect Pentium bug [ChenBryant DAC'95]
- Cannot be applied fully automatically

#### **Theorem Proving**

- Used in industry
- Past: Requires manual effort
- 2020: Progress in ACL2

[TemelSlobodovaHunt CAV'20]

#### **Algebraic Approach**

#### Seminal work:

[CiesielskiYuBrownLiuRossi DAC'15]

[SayedGroßeKühneSoekenDrechsler DATE'16]

- Polynomial encoding
- Works for non-trivial multiplier designs









**Unsigned Integers:** 

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

**Unsigned Integers:** 

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

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

#### **Unsigned Integers:**

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

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

#### **Unsigned Integers:**

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

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

#### **Unsigned Integers:**

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

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





15

 $1 \mid 0$ 



y





Gate polynomials G(C).

 $s_3 = g_1 \wedge g_4 \qquad -s_3 + g_4 g_1,$ 



$$s_3 = g_1 \wedge g_4 \qquad -s_3 + g_4 g_1, \ s_2 = g_1 \oplus g_4 \qquad -s_2 - 2g_4 g_1 + g_4 + g_1,$$



$$\begin{array}{ll} s_3 = g_1 \wedge g_4 & \quad -s_3 + g_4 g_1, \\ s_2 = g_1 \oplus g_4 & \quad -s_2 - 2 g_4 g_1 + g_4 + g_1, \\ g_4 = g_2 \wedge g_3 & \quad -g_4 + g_2 g_3, \end{array}$$



$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$



$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1+a_1b_1,$



$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1+a_1b_1,$
$g_2 = a_0 \wedge b_1$	$-g_2+a_0b_1,$



$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1+a_1b_1,$
$g_2 = a_0 \wedge b_1$	$-g_2+a_0b_1,$
$g_3 = a_1 \wedge b_0$	$-g_3+a_1b_0,$



$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1+a_1b_1,$
$g_2 = a_0 \wedge b_1$	$-g_2 + a_0 b_1,$
$g_3 = a_1 \wedge b_0$	$-g_3+a_1b_0,$
$s_0 = a_0 \wedge b_0$	$-s_0 + a_0 b_0$



#### Gate polynomials G(C).

$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1+a_1b_1,$
$g_2 = a_0 \wedge b_1$	$-g_2 + a_0 b_1,$
$g_3 = a_1 \wedge b_0$	$-g_3+a_1b_0,$
$s_0 = a_0 \wedge b_0$	$-s_0 + a_0 b_0$

#### Boolean value constraints B(C).

$a_1, a_0 \in \mathbb{B}$	$a_1(1-a_1), a_0(1-a_0),$
$b_1, b_0 \in \mathbb{B}$	$b_1(1-b_1),\ b_0(1-b_0)$



#### Gate polynomials G(C).

$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1+a_1b_1,$
$g_2 = a_0 \wedge b_1$	$-g_2 + a_0 b_1,$
$g_3 = a_1 \wedge b_0$	$-g_3+a_1b_0,$
$s_0 = a_0 \wedge b_0$	$-s_0 + a_0 b_0$

#### Boolean value constraints B(C).

$$a_1, a_0 \in \mathbb{B}$$
  $-a_1^2 + a_1, -a_0^2 + a_0,$   
 $b_1, b_0 \in \mathbb{B}$   $-b_1^2 + b_1, -b_0^2 + b_0$ 



# **Ideals Associated to Circuits**

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers FMSD'20]

#### More circuit relations:

$-s_0 + a_0 b_0$	AND-gate
$-a_1^2 + a_1$	$a_1$ Boolean
$-g_2^2 + g_2$	$g_2$ Boolean
$s_1g_4$	XOR-AND constraint
-	

#### **Polynomial Circuit Constraints.**

A polynomial  $p \in \mathbb{Q}[X]$  is a polynomial circuit constraint (PCC) if for all

$$(a_0,\ldots,a_{n-1},b_0,\ldots,b_{n-1}) \in \{0,1\}^{2n}$$

and resulting values  $g_1, \ldots, g_k, s_0, \ldots, s_{2n-1}$  implied by the gates of the circuit *C* the substitution of these values into *p* gives zero.



### **Ideals Associated to Circuits**

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers FMSD'20]

- The set of all PCCs for C is denoted by I(C).
- $\blacksquare$  *I*(*C*) contains all relations of the circuit *C*.
- $\blacksquare$  I(C) is an ideal.

**Ideal.** A subset  $I \subseteq R[X]$  is called an ideal if

 $\forall \, p,q \in I: p+q \in I \quad \text{ and } \quad \forall \, p \in R[X] \, \forall \, q \in I: pq \in I.$ 

Multiplier. A circuit C is called a multiplier if

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





20

**Basis.** A set  $G = \{g_1, \ldots, g_m\} \subseteq R[X]$  is called a **basis** of an ideal I if

$$I = \{h_1g_1 + \dots + h_mg_m \mid h_1, \dots, h_m \in R[X]\} = \langle G \rangle$$

Given an arbitrary basis G of I it is not obvious how to decide ideal membership.

**Basis.** A set  $G = \{g_1, \ldots, g_m\} \subseteq R[X]$  is called a **basis** of an ideal *I* if

$$I = \{h_1g_1 + \dots + h_mg_m \mid h_1, \dots, h_m \in R[X]\} = \langle G \rangle$$

Given an arbitrary basis G of I it is not obvious how to decide ideal membership.

Solution: We need a basis with certain structural properties, called Gröbner basis.

**Basis.** A set  $G = \{g_1, \ldots, g_m\} \subseteq R[X]$  is called a **basis** of an ideal *I* if

 $I = \{h_1g_1 + \dots + h_mg_m \mid h_1, \dots, h_m \in R[X]\} = \langle G \rangle$ 

Given an arbitrary basis G of I it is not obvious how to decide ideal membership.

Solution: We need a basis with certain structural properties, called Gröbner basis.

Gröbner basis. [Buchberger'65]

Offers decision procedure for ideal membership problem in R[X], where R is a field.

Every ideal of R[X] has a finite Gröbner basis.

Given an arbitrary basis of an ideal, we are able compute a Gröbner basis.

**Basis.** A set  $G = \{g_1, \ldots, g_m\} \subseteq R[X]$  is called a **basis** of an ideal *I* if

 $I = \{h_1g_1 + \dots + h_mg_m \mid h_1, \dots, h_m \in R[X]\} = \langle G \rangle$ 

Given an arbitrary basis G of I it is not obvious how to decide ideal membership.

Solution: We need a basis with certain structural properties, called Gröbner basis.

Gröbner basis. [Buchberger'65]

- Offers decision procedure for ideal membership problem in R[X], where R is a field.
- Every ideal of R[X] has a finite Gröbner basis.
- Given an arbitrary basis of an ideal, we are able compute a Gröbner basis.
- Computing a Gröbner basis over Q and deciding ideal membership is EXPSPACE-complete.[Mayr STACS'89]

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers FMSD'20]

• We can deduce some elements of I(C):

- $\Box$  Gate polynomials G(C)
- $\Box$  Boolean value constraints B(C)
- Let  $J(C) = \langle G(C) \cup B(C) \rangle$ .
- Lexicographic term order: Reverse topological
  Output variable of a gate is greater than input variables [LvKallaEnescu TCAD'13].

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

Proof idea: Application of Buchberger's Product criterion.

### **Soundness and Completeness**

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers FMSD'20]

#### Theorem

For all acyclic circuits C, we have J(C) = I(C).

■  $J(C) \subseteq I(C)$ : soundness ■  $I(C) \subseteq J(C)$ : completeness
#### **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.

$$8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0$$

$$\frac{8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0}{8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0}$$

$$\frac{8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0}{8g_1g_4} + \frac{4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0}{4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0}$$

$$\begin{aligned} &8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ &8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ &4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ &4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \end{aligned}$$

$$\begin{aligned} &8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ &8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ &4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ \hline \hline &4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ &4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \end{aligned}$$

$$\begin{split} &8s_3+4s_2+2s_1+s_0-4a_1b_1-2a_1b_0-2a_0b_1-a_0b_0\\ &8g_1g_4+4s_2+2s_1+s_0-4a_1b_1-2a_1b_0-2a_0b_1-a_0b_0\\ &4g_4+4g_1+2s_1+s_0-4a_1b_1-2a_1b_0-2a_0b_1-a_0b_0\\ &4g_2g_3+4g_1+2s_1+s_0-4a_1b_1-2a_1b_0-2a_0b_1-a_0b_0\\ &4g_1+2g_3+2g_2+s_0-\boxed{4a_1b_1}-2a_1b_0-2a_0b_1-a_0b_0\\ &2g_3+2g_2+s_0-2a_1b_0-2a_0b_1-a_0b_0 \end{split}$$

$$\begin{split} 8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + 2g_2 + s_0 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + s_0 - 2a_1b_0 - a_0b_0 \end{split}$$

$$\begin{split} 8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + 2g_2 + s_0 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + s_0 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ s_0 - a_0b_0 \end{split}$$

$$\begin{aligned} 8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + 2g_2 + s_0 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + s_0 - 2a_1b_0 - a_0b_0 \\ s_0 - \boxed{a_0b_0} \end{aligned}$$

## **Faulty multiplier**

### Gate polynomials G(C).

$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1 + a_1 b_1,$
$g_2 = a_0 \wedge b_1$	$-g_2 + a_0 b_1,$
$g_3 = a_1 \wedge b_0$	$-g_3+a_1b_0,$
$s_0 = a_0 \wedge b_0$	$-s_0 + a_0 b_0$

#### Boolean value constraints B(C).

$a_1, a_0 \in \mathbb{B}$	$a_1(1-a_1), a_0(1-a_0),$
$b_1, b_0 \in \mathbb{B}$	$b_1(1-b_1), \ b_0(1-b_0)$



## **Faulty multiplier**

### Gate polynomials G(C).

$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1 + a_1 b_1,$
$g_2 = a_0 \wedge b_1$	$-g_2 + a_0 b_1,$
$g_3 = a_1 \wedge b_0$	$-g_3+a_1b_0,$
$s_0 = a_0 \wedge b_0$	$-s_0 + a_0 b_0$

#### Boolean value constraints B(C).

$a_1, a_0 \in \mathbb{B}$	$a_1(1-a_1), a_0(1-a_0),$
$b_1, b_0 \in \mathbb{B}$	$b_1(1-b_1), \ b_0(1-b_0)$



## **Faulty multiplier**

### Gate polynomials G(C).

$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1 + a_1 b_1,$
$g_2 = a_0 \wedge b_1$	$-g_2 + a_0 b_1,$
$g_3 = a_1 \wedge b_0$	$-g_3+a_0b_0,$
$s_0 = a_0 \wedge b_0$	$-s_0 + a_0 b_0$

#### Boolean value constraints B(C).

$a_1, a_0 \in \mathbb{B}$	$a_1(1-a_1), a_0(1-a_0),$
$b_1, b_0 \in \mathbb{B}$	$b_1(1-b_1), \ b_0(1-b_0)$



$$\begin{aligned} 8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + 2g_2 + s_0 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + s_0 - 2a_1b_0 - a_0b_0 \end{aligned}$$

$$\begin{aligned} 8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + 2g_2 + s_0 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + s_0 - 2a_1b_0 - a_0b_0 \\ s_0 - 2a_1b_0 + a_0b_0 \end{aligned}$$

$$\begin{aligned} 8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + 2g_2 + s_0 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\ 2g_3 + s_0 - 2a_1b_0 - a_0b_0 \\ s_0 - 2a_1b_0 + a_0b_0 \end{aligned}$$

$$8s_{3} + 4s_{2} + 2s_{1} + s_{0} - 4a_{1}b_{1} - 2a_{1}b_{0} - 2a_{0}b_{1} - a_{0}b_{0}$$

$$8g_{1}g_{4} + 4s_{2} + 2s_{1} + s_{0} - 4a_{1}b_{1} - 2a_{1}b_{0} - 2a_{0}b_{1} - a_{0}b_{0}$$

$$4g_{4} + 4g_{1} + 2s_{1} + s_{0} - 4a_{1}b_{1} - 2a_{1}b_{0} - 2a_{0}b_{1} - a_{0}b_{0}$$

$$4g_{2}g_{3} + 4g_{1} + 2s_{1} + s_{0} - 4a_{1}b_{1} - 2a_{1}b_{0} - 2a_{0}b_{1} - a_{0}b_{0}$$

$$4g_{1} + 2g_{3} + 2g_{2} + s_{0} - 4a_{1}b_{1} - 2a_{1}b_{0} - 2a_{0}b_{1} - a_{0}b_{0}$$

$$2g_{3} + 2g_{2} + s_{0} - 2a_{1}b_{0} - 2a_{0}b_{1} - a_{0}b_{0}$$

$$2g_{3} + 2g_{2} + s_{0} - 2a_{1}b_{0} - 2a_{0}b_{1} - a_{0}b_{0}$$

$$2g_{3} + s_{0} - 2a_{1}b_{0} - a_{0}b_{0}$$

$$s_{0} - 2a_{1}b_{0} + a_{0}b_{0}$$

$$-2a_{1}b_{0} + 2a_{0}b_{0}$$

# **Counter Example**

**Remainder:**  $-2a_1b_0 + 2a_0b_0 \neq 0$ 

## **Counter Example**

**Remainder:**  $-2a_1b_0 + 2a_0b_0 \neq 0$ 

 $a_1 = 0, \quad a_0 = 1$  $b_1 = 0, \quad b_0 = 1$ 

 $01 \cdot 01 = 0001$ 

# **Counter Example**

Remainder: 
$$-2a_1b_0 + 2a_0b_0 \neq 0$$

$$a_1 = 0, \quad a_0 = 1$$
  
 $b_1 = 0, \quad b_0 = 1$ 

 $01 \cdot 01 = 0001$ 



#### **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 increases drastically.

8-bit multiplier cannot be verified within 20 minutes.

[MahzoonGroßeDrechsler ICCAD'18]



## **Incremental Verification**



[RitircBiereKauers FMCAD'17, KaufmannBiereKauers FMSD'20]

Let 
$$P_k = \sum_{k=i+j} a_i b_j$$
.

#### **Column-Wise Checking Algorithm**

Input: Circuit C with sliced Gröbner bases  $G_i$ Output: Determine whether C is a multiplier

 $C_{2n} \leftarrow 0$ 

for  $i \leftarrow 2n-1$  to 0

 $C_i \leftarrow \text{Remainder} (2C_{i+1} + s_i - P_i, G_i)$ 

return  $C_0 = 0$ 

# **AND-Inverter Graphs (AIGs)**



# Rewriting



# Rewriting



# **Variable Elimination**

Variable elimination is based on elimination theory of Gröbner bases.

Identify sub-circuits  $C_S$  in the AIGs and eliminate internal variables

[RitircBiereKauers DATE'18, KaufmannBiereKauers FMSD'20]

- Full-adder rewriting
- Half-adder rewriting
- XOR rewriting
- Eliminating nodes with single dependency

Repeated elimination of variables with single dependency [KaufmannBiereKauers FMCAD'19]:

Does not rely on specific patterns.

## Elimination theory of Gröbner bases

**Elimination order.** Let  $X = Y \cup Z$  and we want to eliminate Z. Order the terms such that for all terms  $\sigma, \tau$  where a variable from Z is contained in  $\sigma$  but not in  $\tau$ , we obtain  $\tau < \sigma$ .

**Elimination ideal.** The elimination ideal *J* where the *Z*-variables are eliminated of  $I \subseteq \mathbb{Q}[X] = \mathbb{Q}[Y, Z]$  is defined by

 $J = I \cap \mathbb{Q}[Y].$ 

**Elimination theorem.** Given an ideal  $I \subseteq \mathbb{Q}[X] = \mathbb{Q}[Y, Z]$ . Further let *G* be a Gröbner basis of *I* with respect to an elimination order Y < Z. Then the set

 $H = G \cap \mathbb{Q}[Y]$ 

is a Gröbner basis of the elimination ideal  $J = I \cap \mathbb{Q}[Y]$ , in particular  $\langle H \rangle = J$ .

## Variable Elimination

Example:  $I \subseteq \mathbb{Q}[o, x, y, a, b].$ Eliminate *y*.

#### Ideal

 $I = \langle$ -o + xy - x - y + 1, -x + ab - a - b + 1.  $-\boldsymbol{u}+ab\rangle$ 

#### Extended ideal

 $I = \langle$ -o + xy - x - y + 1, -o + xab - x - ab + 1, -x + ab - a - b + 1, -x + ab - a - b + 1.  $-\boldsymbol{u}+ab\rangle$ 

## **Rewritten ideal** $I = \langle$ -o + xab - x - ab + 1, $-\boldsymbol{u}+\boldsymbol{a}\boldsymbol{b}\rangle$

### Elimination ideal $I \cap \mathbb{Q}[o, x, a, b] = \langle$ -o + xab - x - ab + 1. $-x + ab - a - b + 1\rangle$

## Local variable elimination

#### Theorem

Let  $G \subseteq \mathbb{Q}[X] = \mathbb{Q}[Y, Z]$  be a Gröbner basis with respect to some term order  $<_G$ . Let  $G_A = G \cap \mathbb{Q}[Y]$  and  $G_B = G \setminus G_A$ . Let  $<_H$  be an elimination order for Z which agrees with  $<_G$  for all terms that are free of Z, i.e., terms free of Z are equally ordered in  $<_G$  and  $<_H$ . Suppose that  $\langle G_B \rangle$  has a Gröbner basis  $H_B$  with respect to  $<_H$  which is such that every leading term in  $H_B$  is free of Z or free of Y. Then  $\langle G \rangle \cap \mathbb{Q}[Y] = (\langle G_A \rangle + \langle G_B \rangle) \cap \mathbb{Q}[Y] = \langle G_A \rangle + (\langle G_B \rangle \cap \mathbb{Q}[Y])$ .

#### Theorem

Let  $G, G_A, G_B, H_B, H_Y, H_Z, <_H, <_G$  be as before. Then  $H = G_A \cup H_Y$  is a Gröbner basis w.r.t. the ordering  $<_H$ .



### Multiplier – Wallace Tree Acc. & Carry Look-Ahead Adder



### Multiplier – Wallace Tree Acc. & Carry Look-Ahead Adder



### Multiplier – Wallace Tree Acc. & Carry Look-Ahead Adder



### **OR Gates**




#### **OR Gates**



 $o = x_0 + x_1 - x_0x_1 + x_2 - x_0x_2 - x_1x_2 + x_0x_1x_2 + x_3 - x_0x_3 - x_1x_3 + x_0x_1x_3 - x_2x_3 + x_0x_2x_3 + x_1x_2x_3 - x_0x_1x_2x_3$ 

 $15 = 2^4 - 1$  monomials

# **SAT & Computer Algebra**



# **SAT & Computer Algebra**



# **SAT & Computer Algebra**

[KaufmannBiereKauers FMCAD'19]





AIG is translated to CNF.

CNF is given to SAT solver.

To show correctness SAT solver needs to return UNSAT.

# AMulet 2.0

Computer Algebra Systems [KaufmannBiereKauers FMSD'20]

- Mathematica, Singular
- Algorithmic power (e.g., Mathematica has > 5000 built-in functions).
- Easy to use.
- Designed for general purposes.

AMULET 2.0 [KaufmannBiere TACAS'21]

- **C++** implementation ( $\sim$  6 000 lines)
- Automatically applies adder substitution and circuit verification.
- Designed to make use of properties of polynomials.
- On-the-fly reduction of Boolean value constraints.



daniela@daninovo:~/algprop/src/amulet2\$ ./amulet ../benchmarks/btor/btor512.aig -verify -v2

dmitiageniewe:-/figrops/src/swittS ./switt ./benchmarks/btor/btor/S12.aig -verify -v2
[awuit2] Adjer multiplicer examination tool
[awuit2] Adjer multiplicer examination tool
[awuit2] Adjer multiplicer examination tool
[awuit2] selected mode: verification
[awuit2] (awuit2] (awuit2) (awuit2) (awuit2) (awuit2)
[awuit2] (awuit2] (awuit2) (awuit2) (awuit2) (awuit2) (awuit2)
[awuit2] (awuit2] (awuit2) (awuit

janiela@daninovo:~/algprop/src/anulet25 ./anulet ../benchmarks/btor/btor512.aig -verify -v2 anulet21 AMulet Version 2.0 amulet21 Aiger multiplier examination tool amulet2] Copyright(C) 2020, Daniela Kaufmann, Johannes Kepler University Linz anulet21 selected mode: verification anulet2] reading '.../benchmarks/btor/btor512.aig amulet21 MTLOA 2092544 1024 0 1024 2091520 amulet21 assuming ordering as in BTOR generated benchmarks amulet2] (a[0], a[1], ..., a[511]) =(input[0], input[2], ..., input[1022] amulet2] (b[0], b[1], ..., b[511]) =(input[1], input[3], ..., input[1023]) anulet2] (s[0], ..., s[1023]) =(output[0], ..., output[1023]) amulet21 allocating 2093568 gates amulet21 found 262144 partial products anulet21 assuming simple pp generator anulet21 found 522752 xor-nates amulet21 marking xor chain gates amulet21 marked 0 xor gates in last slice anulet21 remove internal xor nates anulet21 removed 522752 internal xor gates amulet21 slicing based on xor anulet21 remove single occurence dates amulet21 removed 0 single occurence gates anulet2] moved 0 gates to smaller slices anulet21 remove gates that are not assigned to slices anulet21 removed 0 gates that are not assigned to slices anulet21 anulet21 anulet21 started reducing anulet21 reducing by slice 1023 anulet21 after reducing by slice 1023 amulet21 remainder is -898844567431157953864652595394512366808988489471153286367150405788663379027504815663542386612037680105600569399356966788293948844072083112464237153197320672188883946712432742638151109808623047705972654147604258 28844190753411712314407369565552704136185816752553422931491199736229692398581524176781648121120686088\*14185088+89884656743115795386465259539451236680898848994711532863675866337902750481566354238661203768010560056939935696678829 394884407208311246423715319737062188883946712432742638151109800623047059726541476042502884419075341171231440736956555270413618581675255342293149119973622969239858152417678164812112068608 anulet21 anulet21 reducing by slice 1022 anulet21 after reducing by slice 1022 amule 22] remainder 1s - 440423283715578976032326207697256183404042447355766431835752028043316895137524078317711033060188400952807284030414697442203604155623211857859868531094441973356216371310975554090031152352086327073802125 442209537678585615729368478277635296889299837627671146574559986811484619929876288339882486695663438471557897693232629769725618348644942447355765643183575282894331689513752487831771193386618848965288828465967848339414 3089884894711532863671504057886633790275048156635423866120376801056005693993569667882939488440720831124642371531973706218888394671243274263815110980062304705972654147604250288441907534117123144073695655527041361858167525534229314911 9973622969239858152417678164812112068608 anulet21 anulet2] reducing by slice 1021 anulet21 after reducing by slice 1021 721104768835292807860184239138817603404645418813835573287279993405742309964538104419541203028017152\*141857789488466163148847658091702247122367788423913882156914471658447568762039158855966530094200264001423498392416970 348721101802077811605928829934265547220986678108185659537777450155761764931635369010625721104768835292807860184239138817603404645418813835573287279993405742309964538104419541203028017152\*14185038-224711641857789488466163148848628091 702247122367788321591787601447165844756876203915885596653009420026400142349839241697073487211018020778116059288299342655472209866781081856595377774501557617649316353690140257211047688352928078601842391388176034046454188138355732872 99934057423099645381044195412030260451772509122046163305406741349255733684653984894465458842751667413671033649647753628043414975342706528611747655678995902276405177250912204616330540673343481778648980279664166296003422455697861333 350467285294794906107031877163314306505878423580552717416452810213936256441506719861839980217226929893614313258623609084051456 anulet21 anulet21 reducing by slice 1020 amulet21 after reducing by slice 1020 amulet21 remainder is -112355828928894744233881574424314845851123561183894168795893888723582922378438181919579427983265847169132889671174919628848536743685589818389958829644149671327736184933398548928297688887258778888824658176845853 286055238441764640393860921195694088801762322769406917786643639996762871154982269852209776601514088576\*14184992-112355820928894744233881574424314845851123561183894166795893808723582922378438181957942798326504710813208087117491962084853 67436655999103899580296441496713277361949333995499292768888725875888872587588887258758888725875888872582128695238441764548393999021195694888178232278948854836399967823711549822569827884511498575811484933398157447433481574474314945 851123561183R94166795R93R9B7235R292237R43818195794279R32656471681328689711749196288485367436655998163R995580296441496713277361649333985469282976R8887256778808824658176845853128669523844176464833986921195694888817178564363

999/708/7115498/26965/209776601514086/76\*14184988-1123558709289474423861574256118384160795993806723582922376438[019579427983265647100132000711749195706483567450655908[0389658072964149671327736[049333905409282976888824654] 72507780882465817644697312805212954746493390921195649488017022279694091778064563999702871154982209523707601544085174243155578212322370872561834494942473555649311335758293137242783370133300154982178447493492473555649311335758293137242783370655513724278 5162/5512889126\*1418289+233108/25/311098231/2996/3455393\*010168702/1241129710094-5993533718902176;

amulet2

[amulet2] reducing by slice 1006

[amulet2] after reducing by slice 1006

amulet21 8784388998799381296407897878785181233758875678875678875678875674571938659481254893287655745719386594812548932876557457193865942575625657457193865942158755642893294674411352112378659481254893287655645745719386987387659497447439894929441135211237865948125489328765574571938698738765948125489329467441135211237865948125489328765588755681162756448964811587592994794 188924855782467271995911839564696244204534920166685986672348139681197729828438899879038129647887851812337588756783866694877472399175388818986765779497439894924424111352112378659481254893202653255657457193869873826758922 5767969757581162756449064\*14182630-685765508599211085496992831398401158759299979491541568764000248557824672719959118395646962442045349261166590866723401396811977298284308098798301296478070878745181233758875678306694877472399175308018 PR6/765779497439R949244241113521123786594812548932826557457193869873826755925765796975758116275644R864\*1418255885992118854869928313984811587592998738924541135211237865948125485785486929421839564696244284534928166659866723481 3968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521123786594812548932026532556574571938698730267509225767960757581162756440064\*14182632-68576550859921108540699203139840115521123786594812548932026532555574571938698730267509225767960757581162756440064\*14182632-685765508599211085406992031398401155211237865948125489320265325555574571938698730267509225767960757581162756440064\*14182632-685765508599211085406992031398401155211237865948125489320265325555574571938698730267509225767960757581162756440064\*14182632-68576550859921108540699203139840115521123786594812548932026532555574571938698730267509225767960757581162756440064\*14182632-6857655085992110854069920313984011552112378659481254893202653255555745719386987302675092078 49154158876488924855782467271995911839564696244284284534928168529481396811977298284388899879838129647887887874518123375887587838669487747239917538881896676577049743989492442411135211237865948125489328 730267509225767960757581162756440066\*14182630-68576550859921108546699283139840115875929907949154150876400024875782467271995911839564696244204534920166059066723401396811977298284308098989301296478070878745181233758875078306694877472 76577949743989492442411135211237865948125489328265325565745719386987382675799275767960757581162756440064\*14182628-685765508599211085406992031398401158759299879491541508764000248557824672719959118395646962442045349201666 5996672340130681197729828438890979030129647887087874518123375887587876678866604877472399175388189667657704974398949244241113521123786594812548932025555745719386987392575692575679697575811627554498644\*1418521588594211685469491254943111521123786594812548932025565745719386987392575892257679697575811627554498644\*14185211685469491254943111521123788594812548932025565745719386987392575892257679697575811627554498644\*14185211685469429678544987392575812975894295769497439994924424111352112378659481254993292556574571938698739257589225767969755811627554498644\*141852116854694211685469421297854498444\*141852110386987392575811627564498644\*1411152111237865948125499329255657457193869873925758925758921168546987457594874598 83139R401158759299079491541588764060248557024672719959118395646962442045349201660590667234013968119772982843080987903012964780708787451812337588750783066948774723991753080189067657794974398949244241113521237865948125489320265325565 74571938698730267509225767960757581162756440064\*14182674-6857655085992110854069920313984011587592990794915415087640002485570246727199591183956469624420453492016605906672340139681197729828430809879030129647807087874518123375887507830 45349201660572340139681197729828430809879030129647807687874518123375887507830669487747239917530880149244241113521123786594812548932026552457193869873025757940743080929424241113521123786594812548932026557457193869873025765960757581162756440664\*14182620-6857655685992 186546600293130846115875020067040154158976486924855787447771005011830564666242944534029166850966773481306811077208284388608708381206478478782875887547838660497747230017538881806676577040743080402442111352117378650481254802 2026532556574571938698730267509225767960757581162756440064\*14182618-68576550859921108540699203139840115875929907949154150876400024855702467271995911839564696244204534920166059066723401396811977298284368098790301296478070878745181233 \$948774723991753080189667657794974398949244241113521123786594812548932026532556574571938698730267509225767966757581162756440864\*14182616+68576550859921108540699203139840115875929907949154150876400024855702467271995911835 n45340201666672340130681107720828430869979030129647807878758181233758875678386604877472390175308018966765770407439804024424111352112378650481254803202565745719386087380257657457193865048725811627564408644141826114-68 5765588500211885486092813398481158750299879491541588750299879491183956469244285782467271995911839564696244284584288768787487887887587487887587534672349811573758875878346744787887587534672349811573758875875467487878787538875875346723498747339849244241113521123788 5948125489328265325565745719386987382675892257679687575811627564488649878921188548699263139848115875929987949154158876488824855782467271995911839564696244285578246727199591183956469672348139681197729828388987993812964788788 14182668-68576558859921168546699283139846115875929987949154158876468692485576246727199591183956469624420453492016665598667234013968119772982843886987903012964788768787451812337588756783066948774723991753088189667657794974398949244241 13521123786594812548932626532556574571938698736267509225767960757581162750440004\*14182606+123437791547857995373258565651712268576673834308477471577520044740204441089592641311216453239568162856298906320102122514261559136911754577822 42333666527581741326228765975146952658779458318515554434832178483095391818863963409843388228158786625888776477586818342294896577144815166863823293636468929615921152

[amulet2]

[amulet2] reducing by slice 1005

[amulet2] after reducing by slice 1005

amulet2 remainder is 34/28827542996055427034960156992005793796495397457707543820001242785123363599795591978234812210226746008302953336170069840598864914215404939515064823903543937259061687943753915334743873619958765400945338288974 x7190474622128556768561893297486274286581397545678285296934513375461288398837879858133754612883988378122182263336178 91421546493951566482398354393755966168794375391533474387361995876548699453387889748719947462212855676656189329748662742466613266278287285969349365133754612883988378798581378296832\*14182358-342882754299665542783496815692685793796495397 2883080378700581378220032414182348-3428827542006055427034060156002005503745400530745707543820001242785123363500705501078234812210253356170068874540142154040305156040823003554230472500615870437530153347438736100587654000 45338288974871994746221285567685618932974862774466013266278287285969349365133754612883980378798581378220032\*14182346-34288275429968554270349660156992805793796495397457787543820001242785123363599795591978234812210226746008302951336170 6984059886491421540493951506482390354393725906168794375391533474387361995876540094533828897487199474622120556760561893797406627446601326627828596934936513375461288398037879058137754612883980378790581378220032\*14182344-34288275429960554270349601569920057 74577875438280012427851233635097055019782348122102267265018820253336178060848598864914215484039515864823983543937259861687943753915334743873619058765480945338288974872190474622128556768561803297486627344668132662782822 365133754612883980378790581378220032\*14182342-3428827542996055427034960156992005793796495397457707543820001242785123363599795591978234812210226746000830295333617006984059886491421540493951506482390354393725906168794375391533474387361 828807487100474622120556764056180320746527426561326527828728505034035513375461288308037870058137820803187005813784.342882754290565542783406415600208530745708754382080124278512335350070550107823481221822674588928 2953336176 nn69840598864914215464939515864823903543937259061687943753915334743873619958765400945338288974871994746221205567605618932974062744660132662782872859693493651337546128839893782905813775461288398978790581377546128839897879058137754612883989787905813775461288398978790581377546128839897879058137754612883989787905813775461288398978790581377546128839897879058137754612883989787905813775461288398978790581377546128839897879058137754612883989787828975470398785477905813775461288398978790581377546128839897878790581377546128839897878790581377546128839897878790581377546128839897878790581377546128839897879058137754612883989787854790581377546128839897878547905813775461288978787905813775461288398978787905813775461288978 87285969349365133754612883980378790581378220032\*14182336-34288275429960554270349601569920057937964953974577075438200012427851233635997955919782348122102267460083029533361700c9840598864914215404939515064823903543937259061687943753915 33474387361995876540094533828897487199474622120556760561893297406274466013266278287969349365133754612883980378790581378220032\*14182334-3428827542996055427034960156992005793796495397457207543820001242785123363599795591978234812210 226746688382953336178864914215484939515664823983543937259861687943753915334743873619958765488942533828897487199474652128556768561893297486527845661326652782872859693493651337546128839889378799581378228932\*3428827542996 6695793796495397457787543821661242785123363599795591978234812218226746688838295333617086984859886491421546493951566482398354393725986168794375391533474387361995876548699453382889748719947462212855676656189329748627446 68132662782872859693493651337546128839883787985813782269324713482389-3428827542996955427834966156992695793796495397457785438266912427851233635997955919782348122169267466988649349325158648239935439372596616 674689830295333617006984959886491421548493951596482390354393725590516879437539153347438736199587654909453382889748719947462212855676055189329748662744668132662782872859693493651337554612883989378799581378220932\*14182326-34 297486274466013266278287285969349365133754612883980378799581378220032\*14182324-342882754299605542703496015699200579379649539745770754382000124278512336359979559197823481221022674600830295333617006984059886491421540493951586482390354 14182320-342882754299665542783496615699289579379649537539765438286691242785123363599795591978234912216257666882829953361786698864914215464939515864823993543937559661687943755915334743873619958765486945338288974871994746221265 0648/230035430372500616870437530153347438736190587654000453382880748719047462212855576056180329744662734466213266278287285060349365133754612883980378709581378220832\*14182316+65147773316925953113664242982848110082133412516964433325800

b410\*a428-18328898567823975176828821717551896634061913878189217618842527889479880963425858661988981930127762168274798885960159828847613998884526961 401\*1415.18128898507821975170828021717551894614061913070189217618842527809479 r9682560526831695838459555430831205351382962374814354898944+b399\*+439-18328898507823975170828021717551806634061913070189217618842527809479088963425858661980981930129766 441679682568526831695838459555438831205351382962374814354898944\*b392\*6446-183288985678239751708280217175518966340619130701892176188 100571282105574645075288774445075288774445444167068754452278094700809634258786651084991035518820623748143548080944+2)31\*a447-183288095507823075170528821717551896634061011070180217418842527809473088096342587869470880963425878641358865108498103120315118206237454143548080944+2)31\*a447-18328809557424627388452788443081598288 4526961088284816666661138057128230557106665745450752807346546416796825605288316058384595554308312851382962374814354898044+b386+a452-1832889850782307517882862171755180661486101307018 10002440 1000023 100057 55430831205351382062374814354808944+b382+a456-183288985078230751788280217175518966346619130701892176188425278894706 0734654641670682566526831605818450555430831205351812062374814354808044\*5176\*a462-18128808507821075170828021717551806614061011070180217618842527880470 101012776216827470008509815082084761100088452606108228481666666011005712821055716669574545075288714655646054831605838450555408312055513829427414143548098944+b)71+a467-18728808507817555106629417175510669446101107512821055716669574545075288714655610832941717551066947481000512821055118209517414143548080944+b)71+a467-18728808507817055106828481666660110057128210557166695481566546100828445010088452600810008 260610828481666666133095712823955710669574545075280734654641679682560526831695838459555430831205351382062374814354898944+b370\*a460+18328890562371755180663406191307521892176188425278004790809634258586619809819381205351382062374814354898944+b370\*a460+1832889056237175518066340619130771282395571066957458450752807346546416796825603247900859081598 Mark 12 (A 1 ) Bar 1 (A 1 ) Bar 1 (A 1 ) Bar 1 08044+b369+a478-18328085078230751768280217175518966340610130701802176188425278004790800634258586610800810301277621602747908859001508288476130 25605768116958184595554108112051511820623748143154808044+5154+5484-1812880850782107517082807521085710652987145546416 508/268476110008845/26061088754491666666011100571282105571066057454597528871465968256052681160593845955543083120513182062174914354898944+b351==+487.1812889859782307517892802171755 452606108828481666660133005712823055718660574530875288752887528875288750873455464167058256855683160581845055543083120513182062374814354808044=h340=a480\_1832880850745370828021717551806634865013070188221761882217618842527 454507578071465464167068756057687160558184505558308312053513870673748143548085044+5141+3407-181288085078731075178758821717551896514861011878180217518842527806479880954425856610860810381277475 51382062374814354808044+h330+ad00.183288085478230751788288217175518066348610138781802174 16605745458975289734654641679682568526831605838459555439831285351382962374814354898044+1337+4591.1832 1641670682568526831605838450555438831285351382062374814354808044+b336+a582.18328808587823075178828821717551806634861013878180 





```
amulet21 remainder is -4096*13040-4096*13038-4096*13036-4096*13034-4096*13032-4096*13030-4096*13028-4096*13026-4096*13024-4096*13022+4096*13016+40960
anulet21
amulet21 reducing by slice 11
anulet21 after reducing by slice 11
anulet21 remainder is -2048*12870-2048*12868-2048*12866-2048*12864-2048*12862-2048*12860-2048*12858-2048*12856-2048*12854+2048*12854+2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12854-2048*12
anulet21
anulet21 reducing by slice 10
anulet2] after reducing by slice 10
amulet2] remainder is -1024*12716-1024*12714-1024*12712-1024*12710-1024*12708-1024*12706-1024*12706-1024*12704-1024*12702+1024*12696+8192:
anulet21
anulet21 reducing by slice 9
amulet2] after reducing by slice 9
anulet21 remainder is -512*12578-512*12576-512*12574-512*12572-512*12578-512*12568-512*12566+512*12560+3584;
anulet21
anulet21 reducing by slice 8
amulet2] after reducing by slice 8
amulet2] remainder is -256*12456-256*12454-256*12452-256*12458-256*12448-256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12446+256*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*12456*
amulet2] reducing by slice 7
amulet21 after reducing by slice 7
amulet2] remainder is -128*12350-128*12348-128*12346-128*12344-128*12344-128*12344-128*12346+648
anulet21
anulet21 reducing by slice 6
anulet21 after reducing by slice 6
amulet21 remainder is -64*12260-64*12258-64*12256-64*12256+64*12248+256:
anulet21
anulet2] reducing by slice 5
amulet2] after reducing by slice 5
anulet2] remainder is -32*12186-32*12184-32*12182+32*12176+96:
anulet21
amulet2] reducing by slice 4
anulet21 after reducing by slice 4
anulet2] remainder is -16*l2128-16*l2126+16*l2120+32:
anulet21
anulet21 reducing by slice 3
anulet21 after reducing by slice 3
anulet21 remainder is -8*12086+8*12080+8:
amulet2] reducing by slice 2
anulet21 after reducing by slice 2
amulet21 remainder is 4*12056:
anulet21
anulet21 reducing by slice 1
anulet21 after reducing by slice 1
anulet21 remainder is 0:
anulet21 reducing by slice 0
amulet21 after reducing by slice 0
anulet21 remainder is 0:
anulet21
anulet2
anulet21 CORRECT MULTIPLIER
anulet21
anulet21 maximum resident set size:
                                                                                     1966.98 MB
anulet21 used time for initializing:
                                                                                         1.33 seconds
amulet21 used time for slicing/elimination:
                                                                                         7.62 seconds
anulet21 used time for reduction:
                                                                                        30.72 seconds
anulet21 used time for freeing memory:
                                                                                         3.40 seconds
anulet21 total process time:
                                                                                        43.07 seconds
```

Is the circuit really really correct?

# **Proofs**



#### Problem:

- Can we trust our own implementation?
- Is the verification process correct?

# **Proofs**



#### Problem:

- Can we trust our own implementation?
- Is the verification process correct?

#### Solution:

Validate result of verification process.

- Generate machine-checkable proofs.
- Check by independent proof checkers.

#### **Practical Algebraic Calculus**



$$\begin{array}{ll} G(C) &= \{-b+1-a, \; -c+ab\} \\ B(C) &= \{-a^2+a\} \\ {\rm Spec} &= c \end{array}$$

[RitircBiereKauers SC2'18]

[KaufmannBiereKauers FMCAD'19]

[KaufmannFleuryBiere FMCAD'20]

1036 ... 120085 -1540+1536\*1534\*1520-1536\*1534-1536\*1520-1534\*1520+1534+1520;

94937 \*\* 129989

94938 +: 129988.

94939 \*\* 129982

194939. -\$2+1544\*1548\*1542-1544\*1548-1544\*1542-1548\*1542+1548\*1542+ 94949 11 194934

94941 + 129914

94942 \*: 129916

194944 +: 129928, 129921, -s8+b8\*a8;

0000904 (112)93200-333372000372948855892746252171976063317496166418141009864396001978282409984\* 254 32\*1259822+28948822309329048855892746252171976963317496166418141009864396081978282409984\*1259834\*1259834\*2949864396081978282409984\*1259834\*2949842309329048855892746252171976963317496166418141009864396081978282409984\*1259834\*2949842309329048855892746252171976963317496166418141009864396081978282409984\*1259834\*2948842309329048855892746252171976963317496166418141009864396081978282409984\*1259834\*2948842309329048855892746252171976963317496166418141009864396081978282409984\*1259834\*2948842309329048855892746252171976963317496166418141009864396081978282409984\*1259834\*2948842309329048855892746252171976963317496166418141009864396081978282409984\*1259834\*2948842309844\*1259834\*2948842309844\*1259834\*1259834\*2948842309844\*1259834\*2948842309844\*125983 

1259847 \*\*\* 578969644618658097711785492594343953926634992333282082819728792083955564819968\*1259824\*125982 578966446186580977117854925641439539266349923328/002820197287920039565648199681325804451855809771178549256444196580977117854925644419539266349923328/0028201972879200395656481996812580445185580977117854925644419539266349923328/002820197287920039565648199681258047117854925644419539266349923328/002820197287920039565648199681258047117854925644419539266349923328/0028201972879200395656481996812580477117854925644618580977117854925644419539266349923328/0028201972879200395656481996812580477117854925644419539266349923328/0028201972879200395656481996812580477117854925644618580977117854925644618580977117854925644419539266349923328/0028201972879200395656481996812580477117854925644419539266349923328/002820197287920039565648199681258047117854925644199538

94948 \*: 7. ...28948807239172944855582746/5217197694311749616641811409964396081978282409984\*1259827471718549530433353926614992133282804393592661499233318282043353959266149923318282049984 259822+28948022309329048855892746252171976963317496166410141049864396001978282409984, 28948022309329048855892746252171976963317496166410141009864396001978282409984, 259832+25982+259832+25982+259832+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982+25982 34\*1259828\*221091792825492121091792825499584\*1259834\*12 9812\*1.25982.8\*125982.2\*5789064461865889771178549256914395392/66149923332820282019728792083956564819968\*1259832\*1.25982.8\*125982.9\*129048855892746252171976963317496166410141809864396801978282201978282409984\*1259832\*1.25982.9\*129048855892746252171976963317496166410141809864396 01978282489984\*1259832

9884. 57896644618658097711785492594343953926614992332826282019728792001956564819968\*125982\*257879604461858097711785492594343953926614992332826282019728792001956564819968\*125982\*2598644195392465589274652717956933174961664101410093043960019782 78282489984

84\*\259828-28948822389329848855892746252171976963317496166410141809864396081978282489984\*\259826\*\28982\*\259826\*\259826\*\259826\*\259826\*\259826\*\259826\*\259826\*\259826\*\29982\*\259826\*\259825\*\259826\*\259826\*\259826\*\259826\*\259826\*\259826\*\259826\*\259826\*

396001978282409984\*1259834+28948822309329048855892746252171976963317496166418141009864396001978282409984\*1259832\*12598263499233282028201972872003956564819968\*1259828-2894882230932904885589274625217197696331749616641814100986439600 978282489984\*1259822.28948822389329848855892746252171976963317496166418141889864396881978282489984

194953 +: 194948. 194949. 28948827289724052217197060331749616641810499884\*1259834\*1259822+5789668446186588977117854925843439539266349923328280197287920839565648199684\*1259834\*1259834\*1259822+5789668446186588977117854925843439539266349923328280197287920839565648199684\*1259834\*1259834\*1259822+578966844618658897711785492584343953926634992332828019787287920839565648199684\*1259834\*125834\*1259834\*1259834\*125834\*12 0141099864396001978282489984\*1259828\*1259824\*28948022389329648855892746252171976963317496166410141009864396001978282489984\*1259828\*1259824\*289480223893796252171976963317496166410141009864396001978282489984\*1259828\*289480223893796252171976963317496166410141009864396001978282489984\*1259828\*28948022389379648855892746252171976963317496166410141009864396001978282489984\*1259828\*289480223893796252171976963317496166410141009864396001978282489984\*1259828\*289480223893796252171976963317496166410141009864396001978282489984\*1259828\*28948022389379648855892746252171976963317496166410141009864396001978282489984\*1259828\*28948022389379648855892746252171976963317496166410141009864396001978282489984\*1259828\*28948022389379648855892746252171976963317496166410141009864396001978282489984\*1259828\*2892\*28928\*28928\*2892\*28928\*28928\*2892\*28928\*2892\*28928\*2892\*28928\*2892\*28 

141089864396001978282409984\*1259828-28948822309329048855892746252171976963317496166410141099864396001978282409984\*1259824\*28948822309329048855892746252171976963317496166410141099864396001978282409984\*1259824\*28948822309329048855892746252171976963317496166410141099864396001978282409984\*1259824\*28948822309329048855892746252171976963317496166410141099864396001978282409984\*1259824\*28948822309329048855892746252171976963317496166410141099864396001978282409984\*1259824\*28948822309329048855892746252171976963317496166410141099864396001978282409984\*1259824\*28948822309329048855892746252171976963317496166410141099864396001978282409984\*1259824\*28948822309329048855892746252171976963317496166410141099864396001978282409984\*1259824\*28984822309329048855892746252171976963317496166410141009864396001978282409984\*1259824\*1259

4\*1259828\*1259826\*1259825\*1259825\*1259825\*1259825\*1259825\*1259825\*1259825\*1259825\*1259825\*1259825\*1259825\*1259825\*12598 123/264 (2)/26

194956 +: 194955, 194954, -28948022309329048855892746252171976963317496166410141009864396001978282409984\*s234+578960440186580977117854925043439539266349923328200975856548199068\*1259849-2894802230932904885589274625217197696331749616641014100986439600197828240998 

194957 \*: 129935, 14474811154664574427946373126885988481658748883205070584932198809989141204992. -14474811154664574427946373126885988481658748883205070584932198809989141204992. -14474811154664574427946373126885988481658748883205070584932198809989141204992. 14\*125978#-14474011154664524477946373126085984481658748883205076564932198000989141204992\*1259816\*1259788-14474011154664524477946373126085984481658748883205076564932198000989141204992\*125 9814\*1259788+14474011154664524427946373126085988481658748083205070504932198009989141204992\*1259788+14474011154664524427946373126085988481658748083205070504932198000989141204992\*

194959\*: 16. · 14474011154664524427946373126085988481658748083205070504932198000989141204992\*1259814\*\259788-28948022309329048855892746252711976963317496166410141009864396001978282409904\*\259810+14474011154664524427946373126085988481658748083205070504932198000989141204992\*1259814\*\259788-28948022309329048855892746252711976963317496166410141009864396001978282409904\*\259810+14474011154664524427946373126085988481658748083205070504932198000989141204992\*1259814\*\259788-28948022309329048855892746252711976963317496166410141009864396001978282409904\*\259810+14474011154664524427946373126085988481658748083205070504932198000989141204992\*1259814\*\259788-2894802230932904885589274625271976963317496166410141009864396001978282409904\*\259818+164740111546645244279463731260859884816587480832050786439219800989141204992\* 2 2597/88.1447/49111546645244279465171156986481658748081765748081751269859864816587480817512698598648165874808321980009891412608598648165414141009864396001572828499841259 816\*1259810-14474011154664524427946373126085988481658748083205070504932198000989141204992\*1259816\*12649\*1264992\*1259816\*12649816\*1264992\*1259816\*12649\*12649816\*12649\*126 Start 4\*12 Start 1\*1 (Start 4\*1) Start 4\*1 ( 00000141204002#1250014+

2992.28948822319329048855892746/352171976903317496166418141089864396001978282409984\*1259818-1859816\*1259810\*125910\*125910\*125910\*125910\*125910\*125910\*125910\*125910\*125910\*125910\*125910\*125910\*125910\*125910\* 9880893141264992\*1259818\*14259888\*14474011154664574427946373126085988481658748083285870504932198060989141264992\*1259806\*14474011154664574427946373126085988481658748083285870504932198060989141264992\*1259806\*14474011154664574427946373126085988481658748083285870504932198060989141264992\*1259806\*1457401154664574427946373126085988481658748083285870504932198060989141264992\*1259806\*145748083785870504932198060989141264992\*1259806\*14574011154664574427946373126085988481658748083785870504932198060989141264992\*1259806\*14574011154664574427946373126085988481658748083785870504932198060989141264992\*1259806\*14574011154664574427946373126085988481658748083785870504932198060989141264992\*1259806\*1457401115466457442794637312608598848165874808378587050493219800098914126492\*1259806\*1457401115466457442794637312608598848165874808378587050492 198888989141284992\*1259888

259808-2894882330932904885589274625217197696331749616641814069864396801978282409984 ...1447401115466452442794617312608598481658748092198806989141284992\*1259886\*125986\*125988\*125988\*125988\*125988\*125988\*125988\*125988\*125988\*125988\*12598\*125988\*125988\*125988\*125 810\*1259886+14474011154664524427946373126885988481658748883285874808328587481658748883285874825217197866331749916418141989841155874888328427946373126859884816587488832858748252171978663317499164111546641244729463731268598848165874888328587488832858748883285874888328587488832858748883285874888328587488832858748883285874888328587488832858748883285874888328587488832858748838285874883828587488382858748838285874883828587488382858748838288388388388588748838838858874883874883883885874883887488388388588748838838

5,000 (1)900, (2)900, 6+78948827389329848825892746252171976963317496166418141899864396681978282489984\*1259802746252171976963317496166418141899864396681978282489984\*1259802746252171976963317496166418141899864396681978282489984\*

259800\*1259790+14474011154664524427946373126085988481658748083205070504932198000989141204992\*1259790+14474011154664524427946373126085988481658748083205070504932198000989141204992\*

1259790+14474011154664574477946771156985998746751796677115698598481558749983165778669711569869712598099712598097125980971259809712598097125980971259809712598097125980 802\*\259796-1447401115466452442794637312688598848165874808328587874625217197696331749616641014100986439600197828240994\*\259796-14474011154664524427946373126885988481658748083205070564932198000989141204992\*\259804885589274625217197696331749616641014100986439600197828240994\*\2 Septem 12 SQ794+12 SQ79+12 SQ79+12 SQ79+12 SQ79+12 SQ79+12 SQ79+12 SQ79+12 00989141204992\*1259880

# **Proof Checker**

PACTRIM [RitircBiereKauers SC2'18]

- **W**ritten in C ( $\sim 2000$  lines of code).
- Supports older PAC formats, where antecedents are given explicitly.

PACHECK [KaufmannFleuryBiere FMCAD'20]

• Written in C ( $\sim 1700$  lines of code).

Supports new calculus.

Backward compatible to older PAC formats.

Is the circuit really really really correct?

# PASTÈQUE

[KaufmannFleuryBiere FMCAD'20]

#### Theorem Prover Isabelle/HOL



Refinement Approach, relying on Isabelle's Refinement Framework

abstract specification on ideals: specification in ideal

■ final step: executable checker

Isabelle's Archive of Formal Proofs 8000 lines of code

## **Recent Related Work**

- M. Ciesielski, C. Yu, W. Brown, D. Liu, and A. Rossi. Verification of Gate-level Arithmetic Circuits by Function Extraction. In DAC, 2015.
- C. Yu, M. Ciesielski, and A. Mishchenko. Fast Algebraic Rewriting Based on And-Inverter Graphs. IEEE TCAD, 2018.
- M. Ciesielski, T. Su, A. Yasin, and C. Yu. Understanding Algebraic Rewriting for Arithmetic Circuit Verification: a Bit-Flow Model. IEEE TCAD, 2019.
- 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 DATE'16.
- A. Mahzoon, D. Große, and R. Drechsler. PolyCleaner: Clean your Polynomials before Backward Rewriting to verify Million-gate Multipliers. In ICCAD'18.
- A. Mahzoon, D. Große, and R. Drechsler. RevSCA: Using Reverse Engineering to Bring Light into Backward Rewriting for Big and Dirty Multipliers. In DAC'19.
- A. Mahzoon, D. Große, C. Scholl, and R. Drechsler. Towards Formal Verification of Optimized and Industrial Multipliers. In DATE, 2020.

## **Evaluation**



Comparing our approaches to most recent related work.

- Our work
   Mathematica [KaufmannBiereKauers FMSD'20]
   AMULET [KaufmannBiereKauers FMCAD'19]
   AMULET 2.0 [KaufmannBiere TACAS'21]
- Yu et al. [ABC-based]
- Mahzoon et al.

[RevSCA] [RevSCA-2.0]

- 384 different multipliers
- Input bit-width n = 64
- Time-limit: 300 sec

## **Evaluation**



Comparing our approaches to most recent related work.

- Our work
   Mathematica [KaufmannBiereKauers FMSD'20]
   AMULET [KaufmannBiereKauers FMCAD'19]
   AMULET 2.0 [KaufmannBiere TACAS'21]
- Yu et al. [ABC-based]
- Mahzoon et al.

[RevSCA] [RevSCA-2.0]

- Array ripple-carry multiplier
- Input bit-width n in [128, 2048]
- Time-limit: 86 400 sec (24h)

#### Conclusion

#### Is the circuit correct?

Yes, because we have tested some cases.

#### Is the circuit really correct?

Yes, because we applied formal verification.

#### Is the circuit really really correct?

Yes, because we generated and checked a proof certificate.

#### Is the circuit really really really correct?

Yes, because we used a verified proof checker.









# COMBINING SAT AND COMPUTER ALGEBRA FOR CIRCUIT VERIFICATION

# Daniela Kaufmann

Johannes Kepler University

Linz, Austria

#### **Beyond Satisfiability Workshop**

Simons Institute for the Theory of Computing (online)

February 16, 2021











#### **Proofs**



#### Problem:

- Can we trust our own implementation?
- Is the verification process correct?

#### Solution:

Validate result of verification process.

- Generate machine-checkable proofs.
- Check by independent proof checkers.

#### Ideal

Ideal membership problem. Given a polynomial  $f \in R[X]$  and a (finite) set of polynomials  $P \subseteq R[X]$ , decide whether  $f \in \langle P \rangle$ , where  $\langle P \rangle$  is the smallest ideal containing all elements of P, also known as the ideal generated by P.

**Ideal.** A nonempty subset  $I \subseteq R[X]$  is called an ideal if

 $\forall p, q \in I : p + q \in I$  and  $\forall p \in R[X] \ \forall q \in I : pq \in I$ 

**Basis.** A set  $P = \{p_1, \ldots, p_m\} \subseteq R[X]$  is called a **basis** of an ideal *I* if

$$I = \{q_1p_1 + \dots + q_mp_m \mid q_1, \dots, q_m \in R[X]\} = \langle P \rangle$$

## **Ideal Membership Problem**

Given an ideal  $I = \langle p_1, \dots, p_s \rangle \subset R[X]$  we can decide whether a given polynomial  $f \in R[X]$  lies in I as follows:

Compute a Gröbner basis G for I using Buchberger's algorithm.

Given a Gröbner basis G, there is a computable function  $\operatorname{red}_G \colon R[X] \to R[X]$ such that  $\forall f \in R[X] : \operatorname{red}_G(f) = 0 \iff f \in \langle G \rangle$ .

# **Polynomial Calculus**

Let  $G \subseteq R[X]$  and  $f \in R[X]$ .

**Proof:** Sequence  $P = (p_1, \dots, p_n)$ , where each  $p_k$  is obtained by one of the rules:

Boolean Axiom
$$\overline{x_j^2 - x_j}$$
 $x_j \in X$ Addition $\frac{p_i}{\alpha p_i + \beta p_j}$  $\alpha, \beta \in R$ Addition $\frac{p_i}{\alpha p_i + \beta p_j}$  $p_i, p_j$  appearing earlier in the proof  
or are contained in  $G$ Multiplication $\frac{p_i}{qp_i}$  $p_i$  appearing earlier in the proof  
or is contained in  $G$ 

If  $p_n = f$  we write  $G \vdash f$  or in algebraic terms  $f \in \langle G \rangle$ .

**Refutation:**  $G \vdash 1$  or  $1 \in \langle G \rangle$ .

Example



$$R = \mathbb{Q}$$

$$G = \{ -b+1-a, \quad b = \neg a$$

$$-c+ab \} \quad c = a \land b = a \land \neg a$$

$$f = c$$

$$\operatorname{red}_G(f) = 0$$

$$+ \frac{\frac{-b+1-a}{-ab+a-a^2}}{+ \frac{-ab}{c}} \frac{a^2-a}{-c+ab}$$

$$P = (-ab + a - a^2, a^2 - a, -ab, c)$$

We translate the polynomial calculus into a concrete proof format:

given gate and boolean value polynomials serve as axioms
 instances of addition or multiplication rule derive polynomials

#### **Practical Algebraic Calculus (PAC)**

allows automated proof checking
# **PAC Syntax**

```
letter ::= 'a' | 'b' | ... | 'z' | 'A' | 'B' | ... | 'Z'
   number ::= '0' | '1' | ... | '9'
  constant ::= (number)^+
  variable ::= letter (letter | number)*
    power ::= variable ['^' constant]
      term ::= power ('*' power)*
 monomial ::= constant | [constant '*'] term
  operator ::= '+' | '-'
polynomial ::= ['-'] monomial (operator monomial)*
    given ::= (polynomial ': ')*
      rule ::= ('+' | '*') ': ' polynomial ', ' polynomial ', ' polynomial '; '
     proof ::= (rule '; ')^*
```

# **Example - PAC**





f = c

-b+1-a; \* : -b+1-a, -a\*b+a-a^2; a, -c+ab: \* : -a^2+a. -1, a^2-a; -a^2+a; + : -a\*b+a-a^2, a^2-a, -a\*b; + : -a\*b, -c+a\*b, -c; \* : -c. -1. c;

# **Proof Checking**

A proof rule contains four components:

o:v,w, p;

### **Proof checking:**

- Connection property: v, w are given polynomials or conclusions  $p_i$  of previous rules
- Inference property: verify correctness of each rule, e.g. p = v + w for o = " + "

**Target check:** at least one  $p_i$  is equal to target f

# **Proof Checking Algorithm**

- input G sequence of given polynomials
  - $r_1 \cdots r_k$  sequence of PAC proof rules
  - f target polynomial
- output "incorrect", "target-checked", or "correct-proof"

```
P_0 \leftarrow G
for i \leftarrow 1 \dots k
      let r_i = (o_i, v_i, w_i, p_i)
       case o_i = +
           if v_i \in P_{i-1} \land w_i \in P_{i-1} \land p_i = v_i + w_i then P_i \leftarrow \text{append}(P_{i-1}, p_i)
           else return "incorrect"
       case o_i = *
           if v_i \in P_{i-1} \land p_i = v_i * w_i then P_i \leftarrow \text{append}(P_{i-1}, p_i)
           else return "incorrect"
for p_i \in P_k
       if p_i = f then return "target-checked"
```

```
return "correct-proof"
```

### Gate polynomials G(C).

$s_3 = g_1 \wedge g_4$	$-s_3+g_4g_1,$
$s_2 = g_1 \oplus g_4$	$-s_2 - 2g_4g_1 + g_4 + g_1,$
$g_4 = g_2 \wedge g_3$	$-g_4+g_2g_3,$
$s_1 = g_2 \oplus g_3$	$-s_1 - 2g_2g_3 + g_2 + g_3,$
$g_1 = a_1 \wedge b_1$	$-g_1 + a_1 b_1,$
$g_2 = a_0 \wedge b_1$	$-g_2 + a_0 b_1,$
$g_3 = a_1 \wedge b_0$	$-g_3 + a_1 b_0,$
$s_0 = a_0 \wedge b_0$	$-s_0 + a_0 b_0$

### Boolean value constraints B(C).

$a_1, a_0 \in \mathbb{B}$	$a_1(1-a_1), a_0(1-a_0),$
$b_1, b_0 \in \mathbb{B}$	$b_1(1-b_1),\ b_0(1-b_0)$



### Algorithm: Preprocessing

Input : Gröbner basis  $G = \{g_1, \ldots, g_m\}$ Output: Simplified Gröbner basis  $G' = \{g'_1, \ldots, g'_n\}$ 1  $H \leftarrow G;$ 2 while  $\exists q \in H : \operatorname{lt}(q)$  occurs in only one  $p \in H$  with  $p \neq q$  do 3  $r \leftarrow \operatorname{Reduce}(p, q);$ 4  $H \leftarrow (H \setminus \{p, q\}) \cup \{r\};$ 

5 return  $G' \leftarrow H$ 

### Algorithm: Preprocessing

Input : Gröbner basis  $G = \{g_1, \dots, g_m\}$ Output: Simplified Gröbner basis  $G' = \{g'_1, \dots, g'_n\}$ 

- 1  $H \leftarrow G;$
- 2 while  $\exists q \in H: \mathrm{lt}(q)$  occurs in only one  $p \in H$  with  $p \neq q \ \mathbf{do}$
- $r \leftarrow \mathsf{Reduce}(p,q);$

4 
$$H \leftarrow (H \setminus \{p,q\}) \cup \{r\};$$

5 return  $G' \leftarrow H$ 

### Algorithm: Spec-Reduction

Input : Ordered Gröbner basis  $G = \{g_1, \dots, g_m\}$ , Specification fOutput: Determine whether  $f \in \langle G \rangle$ 1  $r \leftarrow f$ ; 2 for i = 1 to m do 3  $\mid r \leftarrow \text{Reduce}(r, g_i)$ ; 4 return r = 0

### Algorithm: Preprocessing

**Input** : Gröbner basis  $G = \{g_1, \ldots, g_m\}$ **Output:** Simplified Gröbner basis  $G' = \{g'_1, \ldots, g'_n\}$ 

- 1  $H \leftarrow G;$
- 2 while  $\exists q \in H : \operatorname{lt}(q)$  occurs in only one  $p \in H$  with  $p \neq q$  do
- $r \leftarrow \mathsf{Reduce}(p,q);$
- 5 return  $G' \leftarrow H$

### Algorithm: Spec-Reduction

Input : Ordered Gröbner basis  $G = \{g_1, \ldots, g_m\}$ , Specification fOutput: Determine whether  $f \in \langle G \rangle$ 1  $r \leftarrow f$ ; 2 for i = 1 to m do 3  $\mid r \leftarrow \text{Reduce}(r, g_i)$ ; 4 return r = 0

### Algorithm: Reduce(p, q)

**Input** : Polynomials p and q**Output:** Remainder r of reducing p modulo q

- 1  $p_d \leftarrow \text{Divide-by-lt}(p, q);$
- 2  $p_m \leftarrow \mathsf{Multiply}(p_d, q);$
- $s r \leftarrow \mathsf{Add}(p, p_m);$
- 4 return r

# **Preprocessing Example**

**Example:**  $I \subseteq \mathbb{Q}[o, x, y, a, b].$ 

Eliminate y.

### Ideal

$$\begin{split} I &= \langle \\ &-o+xy-x-y+1, \\ &-x+ab-a-b+1, \\ &-y+ab \rangle \end{split}$$

# Elimination ideal $I \cap \mathbb{O}[a, r, a, b] = ($

$$-o + xab - x - ab + 1,$$
  
 $-x + ab - a - b + 1
angle$ 

### Algorithm: Reduce(p, q)

2 
$$p_m \leftarrow \mathsf{Multiply}(p_d, q)$$

- $s r \leftarrow \mathsf{Add}(p, p_m);$
- 4 return r

Reduce (-o + xy - x - y + 1, -y + ab)

$$p_d = x - 1$$
  

$$p_m = (x - 1)(-y + ab) = -xy + y + xab - ab$$
  

$$r = (-o + xy - x - y + 1) + (-xy + y + xab - ab)$$
  

$$= -o + xab - x - ab + 1$$

### Algorithm: Preprocessing

**Input** : Gröbner basis  $G = \{g_1, \ldots, g_m\}$ **Output:** Simplified Gröbner basis  $G' = \{g'_1, \ldots, g'_n\}$ 

- 1  $H \leftarrow G;$
- 2 while  $\exists q \in H : \operatorname{lt}(q)$  occurs in only one  $p \in H$  with  $p \neq q$  do
- $r \leftarrow \mathsf{Reduce}(p, q);$
- 5 return  $G' \leftarrow H$

### Algorithm: Spec-Reduction

Input : Ordered Gröbner basis  $G = \{g_1, \ldots, g_m\}$ , Specification fOutput: Determine whether  $f \in \langle G \rangle$ 1  $r \leftarrow f$ ; 2 for i = 1 to m do 3  $\mid r \leftarrow \text{Reduce}(r, g_i)$ ; 4 return r = 0

### **Algorithm:** Reduce(*p*, *q*)

**Input** : Polynomials p and q**Output:** Remainder r of reducing p modulo q

- 1  $p_d \leftarrow \text{Divide-by-Im}(p, q);$
- 2  $p_m \leftarrow \mathsf{Multiply}(p_d, q);$
- $s r \leftarrow \mathsf{Add}(p, p_m);$
- 4 return r

### Algorithm: Preprocessing

**Input** : Gröbner basis  $G = \{g_1, \ldots, g_m\}$ **Output:** Simplified Gröbner basis  $G' = \{g'_1, \ldots, g'_n\}$ 

- 1  $H \leftarrow G;$
- 2 while  $\exists q \in H : \operatorname{lt}(q)$  occurs in only one  $p \in H$  with  $p \neq q$  do
- $r \leftarrow \mathsf{Reduce}(p,q);$

$$\mathbf{4} \qquad H \leftarrow (H \setminus \{p,q\}) \cup \{r\}$$

5 return  $G' \leftarrow H$ 

### Algorithm: Spec-Reduction

Input : Ordered Gröbner basis  $G = \{g_1, \ldots, g_m\}$ , Specification fOutput: Determine whether  $f \in \langle G \rangle$ 1  $r \leftarrow f$ ; 2 for i = 1 to m do 3  $\mid r \leftarrow \text{Reduce}(r, g_i)$ ; 4 return r = 0

# Algorithm: Reduce(p, q) Input : Polynomials p and q Output: Remainder r of reducing p modulo q 1 $p_d \leftarrow$ Divide-by-Im(p, q);

- 2  $p_m \leftarrow \mathsf{Multiply}(p_d, q);$
- **3** Print-Mult-Rule $(q, p_d, p_m)$ ;
- 4  $r \leftarrow \mathsf{Add}(p, p_m);$
- s Print-Add-Rule $(p, p_m, r)$ ;
- 6 return r

- // \*: q, pd, pm;
- // +: p, p<sub>m</sub>, r;

### Algorithm: Preprocessing

**Input** : Gröbner basis  $G = \{g_1, \ldots, g_m\}$ **Output:** Simplified Gröbner basis  $G' = \{g'_1, \ldots, g'_n\}$ 

- 1  $H \leftarrow G;$
- 2 while  $\exists q \in H : \operatorname{lt}(q)$  occurs in only one  $p \in H$  with  $p \neq q$  do
- $r \leftarrow \mathsf{Reduce}(p,q);$

$$\mathbf{4} \qquad H \leftarrow (H \setminus \{p,q\}) \cup \{r\}$$

5 return  $G' \leftarrow H$ 

### Algorithm: Spec-Reduction

Input : Ordered Gröbner basis  $G = \{g_1, \ldots, g_m\}$ , Specification fOutput: Determine whether  $f \in \langle G \rangle$ 1  $r \leftarrow f$ ; 2 for i = 1 to m do 3  $\mid r \leftarrow \text{Reduce}(r, g_i)$ ; 4 return r = 0

### Algorithm: Reduce(p, q)

 $\begin{array}{ll} \mbox{Input} & : \mbox{Polynomials } p \mbox{ and } q \\ \mbox{Output: Remainder } r \mbox{ of reducing } p \mbox{ modulo } q \\ \mbox{1} & p_d \leftarrow \mbox{Divide-by-Im}(p,q); \\ \mbox{2} & p_m \leftarrow \mbox{Multiply}(p_d,q); \\ \mbox{3} & \mbox{Print-Mult-Rule}(q,p_d,p_m); \\ \mbox{4} & r \leftarrow \mbox{Add}(p,p_m); \end{array}$ 

- **5** Print-Add-Rule $(p, p_m, r)$ ;
- 6 return r

- // \*: q, pd, pm;
- // +: p, p<sub>m</sub>, r;

### Algorithm: Preprocessing

**Input** : Gröbner basis  $G = \{g_1, \ldots, g_m\}$ **Output:** Simplified Gröbner basis  $G' = \{g'_1, \ldots, g'_n\}$ 

- 1  $H \leftarrow G;$
- 2 while  $\exists q \in H : \operatorname{lt}(q)$  occurs in only one  $p \in H$  with  $p \neq q$  do

3 
$$r \leftarrow \text{Reduce}(p, q, \mathbf{1});$$

$$4 \qquad H \leftarrow (H \setminus \{p,q\}) \cup \{r\}$$

5 return  $G' \leftarrow H$ 

# Algorithm:Spec-ReductionInput: Ordered Gröbner basis $G = \{g_1, \ldots, g_m\}$ ,<br/>Specification fOutput:Determine whether $f \in \langle G \rangle$ 1 $r \leftarrow f$ ;2for i = 1 to m do3 $| r \leftarrow \text{Reduce}(r, g_i, \mathbf{0});$ 4Print-Add-Rules-for-Factor-Stack();5return r = 0

### Algorithm: Reduce(p, q, print)

# **Reduction Example**

Reduction	Stack	
$r_1 = f$	oradin	
$r_2 = r_1 + p_{d_1}g_1$	$p_{d_1}g_1$	
$r_3 = r_2 + p_{d_2}g_2$	$p_{d_2}g_2$	
$\vdots$ $r_{m+1} = r_m + p_{d_m} g_m = 0$	$\vdots \\ p_{d_m} g_m$	
	Reduction $r_1 = f$ $r_2 = r_1 + p_{d_1}g_1$ $r_3 = r_2 + p_{d_2}g_2$ $\vdots$ $r_{m+1} = r_m + p_{d_m}g_m = 0$	

# **Reduction Example**

Algorithm: Spec-Reduction in AMULET	Beduction	Stack	
<b>Input</b> : Ordered Gröbner basis $G = \{g_1, \ldots, g_m\}$ , Specification $f$	$r_1 = f$	oradin	
<b>Output:</b> Determine whether $f \in \langle G \rangle$	$r_2 = r_1 + p_{d_1}g_1$	$p_{d_1}g_1$	
1 $r \leftarrow f;$	$r_3 = r_2 + p_{d_2} q_2$	$p_{d_2} q_2$	
2 for $i=1$ to $m$ do	10 12 1 page2	$Pa_{2}52$	
$r \leftarrow Reduce(r, g_i, 0);$	:	:	
<pre>4 Print-Add-Rules-Factor-Stack();</pre>	•	·	
5 return $r = 0$	$r_{m+1} = r_m + p_{d_m}g_m = 0$	$p_{d_m}g_m$	

$$0 = r_{m+1} = f + p_{d_1}g_1 + p_{d_2}g_2 + \ldots + p_{d_m}g_m$$

# **Reduction Example**

Algorithm: Spec-Reduction in AMULET	Beduction	
<b>Input</b> : Ordered Gröbner basis $G = \{g_1, \ldots, g_m\}$ , Specification $f$	$r_1 = f$	
<b>Output:</b> Determine whether $f \in \langle G \rangle$	$r_2 = r_1 + p_{d_1}g_1$	
1 $r \leftarrow f;$	$r_3 = r_2 + p_{d_2} q_2$	
2 for $i=1$ to $m$ do	10 2 1 I d <u>2</u> J 2	
$r \leftarrow Reduce(r, g_i, 0);$	:	
4 Print-Add-Rules-Factor-Stack();	•	
5 return $r = 0$	$r_{m+1} = r_m + p_{d_m} g_m = 0$	

$$f = -1 \cdot (p_{d_1}g_1 + p_{d_2}g_2 + \ldots + p_{d_m}g_m)$$

Stack

 $p_{d_1}g_1$   $p_{d_2}g_2$   $\vdots$   $p_{d_m}g_m$ 

### **Practical Algebraic Calculus**

[RitircBiereKauers SC2'18]



 $\forall p, q \in I : p + q \in I$  and  $\forall p \in R[X] \ \forall q \in I : pq \in I$ 

# **Practical Algebraic Calculus**

[RitircBiereKauers SC2'18]



Can be checked by our older proof checker PACTRIM.

# **1. Boolean Variables**

Handle Boolean-value constraints implicitly to reduce number of proof steps.



*	:	-b+1-a,	a,	-a*b;
+	:	-a*b,	-c+a*b,	-c;
*	:	-c,	-1,	c;

# 2. Indices

Introduce indices to reduce proof size.



# 3. Deletion Rule

Introduce a deletion rule to reduce the memory usage of the proof checker.

$$a \xrightarrow{P = 1 -b+1-a;} 2 -c+a*b;$$
  
Spec = c



$$\frac{\overline{x} \vee \overline{y} \quad y \vee z}{\overline{x} \vee z}$$

$$\frac{xy}{x(1-z)} \frac{(1-y)(1-z)}{x(1-z)}$$

$$\frac{xy \quad (1-y)(1-z)}{x(1-z)} \qquad P = 1 \text{ x*y;} \\ 2 \text{ y*z-y-z+1;} \\ \text{Spec} = -x*z+x \end{cases}$$

$$\begin{split} \textbf{Ext}(i,v,p) \quad & (X,P) \Rightarrow (X \cup \{v\}, P(i \mapsto -v + p)) \\ & \text{provided that } P(i) = \bot \text{ and } v \notin X \text{ and } p \in R[X]/\langle B(X) \rangle, \\ & \text{and } p^2 - p \equiv 0 \mod \langle B(X) \rangle. \end{split}$$

$$\begin{array}{rll} xy & (1-y)(1-z) \\ \hline x(1-z) \end{array} & \begin{array}{rll} {\rm P} &=& 1 \ {\tt x} {\tt *y} {\tt ;} \\ & 2 \ {\tt y} {\tt *z} {\tt -y} {\tt -z} {\tt +1} {\tt ;} \\ & \\ {\rm Spec} &=& -{\tt x} {\tt *z} {\tt +x} \end{array} \\ \end{array}$$

# From DRUP to PAC

1. Polynomial encoding of CNF



Add bit-flipping polynomials (similar to "Polynomial Calculus with Resolution"): Clause  $x \lor y \lor z$  can be translated to (1 - x)(1 - y)(1 - z) = 0, which generates  $2^3$  monomials.

If we introduce  $-f_x + 1 - x = 0$ ,  $-f_y + 1 - y = 0$ ,  $-f_z + 1 - z = 0$ , the same equation can be depicted as  $f_x f_y f_z = 0$ .

# From DRUP to PAC

1 1 -2 -3 0 0

2 1 2 0 0

3 -1 -2 0 0

4 -1 2 0 0 5 3 0 0

6 - 2 0 3 1 5 0

7 0 4 2 6 0

### 2. Encoding of resolution steps

Encode resolution using the traces provided in TraceCheck:

```
Let a = 1, b = 2 and c = 3.
```

We encode the first resolution step of rule 6 (resolving clause 3 and 1).

Thus from  $a \lor \overline{b} \lor \overline{c}$  and  $\overline{a} \lor \overline{b}$  we resolve the clause  $\overline{b} \lor \overline{c}$ .

The corresponding PAC encoding is:

\* : b\*a, c, c\*b\*a; + : -c\*b\*a+c\*b, c\*b\*a, c\*b;

### 3. Merge PAC proofs

PAC proofs can be merged by combining constraint sets and proof rules.

# From DRUP to PAC











### **Incremental Verification** $G_3$ $G_2$ $G_1$ $G_0$ $a_1b_1$ $a_0 b_0$ $a_1b_0$ $a_0b_1$ P11 p10 P01 p00 84 82 81 80

[RitircBiereKauers FMCAD'17, KaufmannBiereKauers FMSD'19]

Let 
$$P_k = \sum_{k = i+j} a_i b_j$$

#### **Column-Wise Checking Algorithm**

Input: Circuit C with sliced Gröbner bases  $G_i$ Output: Determine whether C is a multiplier

$$C_{2n} \leftarrow 0$$

for  $i \leftarrow 2n-1$  to 0

$$C_i \leftarrow \text{Remainder} (2C_{i+1} + s_i - P_i, G_i)$$

return  $C_0 = 0$ 

### Order

### **Row-Wise**

### **Column-Wise**



# Slicing

Partial Products. Let 
$$P_k = \sum_{k=i+j} a_i b_j$$
.

**Input Cone.** For each output bit  $s_i$  we determine its input cone

 $I_i = \{ \text{gate } g \mid g \text{ is in input cone of output } s_i \}$ 

**Slice.** Slices  $S_i$  are defined as the difference of consecutive cones  $I_i$ :

$$S_0 = I_0 \qquad S_{i+1} = I_{i+1} \setminus \bigcup_{j=0}^i S_j$$

**Sliced Gröbner Bases.** Let  $G_i$  be the set of gate and boolean value polynomials in  $S_i$ .

# **Carry Recurrence Relation**

### **Carry Recurrence Relation.**

A sequence of 2n + 1 polynomials  $C_0, \ldots, C_{2n}$  is called a **carry sequence** if

 $-C_i + 2C_{i+1} + s_i - P_i \in I(C)$  for all  $0 \le i < 2n + 1$ .

Then  $R_i = -C_i + 2C_{i+1} + s_i - P_i$  are the carry recurrence relations for  $C_0, \ldots, C_{2n}$ .

### Theorem

Let *C* be a circuit where all carry recurrence relations are contained in I(C). Then *C* is a multiplier, iff  $C_0 - 2^{2n}C_{2n} \in I(C)$ .

# **Incremental Algorithm**

### Incremental Checking Algorithm.

input: Circuit C with sliced Gröbner bases  $G_i$  output: Determine whether C is a multiplier

 $C_{2n} \leftarrow 0$ for  $i \leftarrow 2n - 1$  to 0 $C_i \leftarrow$  Remainder (  $2C_{i+1} + s_i - P_i, G_i$  )

return  $C_0 = 0$ 


## **Adder Substitution**



# **Adder Substitution**



7	Algorithm:	Identifying	GP	adders	in	
	AMULET					
<b>Input</b> : Circuit C in AIG format						
<b>Output:</b> Determine whether $C$ might contain a GP						
adder						
$1  j \leftarrow 2n-2, \ \tau \leftarrow 1;$						
2 while $ au$ and $j\geq 0$ do						
3	$\tau, c_j, p_j \leftarrow$					
	Check-if-XOR-and-Identify-p <sub>j</sub> -and-c <sub>j</sub> $(s_j)$ ;					
4	$x_j, y_j \leftarrow Declare-Adder-Inputs(p_j, \tau);$					
5	$j \leftarrow j-1;$					
$6 c_{in} \leftarrow c_j;$						
7 for $i \leftarrow j$ to $2n-1$ do						
8	$m \leftarrow Follow\text{-and-Mark-Paths}(s_i);$					
9 I	9 return $m=0$					



AIG is translated to CNF.

CNF is given to SAT solver.

To show correctness SAT solver needs to return UNSAT.

## **Generated Circuit**



#### **References I**

- [ABC-based] M. J. Ciesielski, T. Su, A. Yasin and C. Yu. Understanding Algebraic Rewriting for Arithmetic Circuit Verification: a Bit-Flow Model. In IEEE TCAD, vol. 39, no. 6, pages 1346–1357, 2020.
- [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.
- [Buchberger'65] B. Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. PhD Thesis, University of Innsbruck, 1965.
- [CleggEdmondsImpagliazzo STOC'96] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Groebner Basis Algorithm to Find Proofs of Unsatisfiability. In Proc. of STOC'96, pages 174–183, ACM, 1996.
- [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**

- [KaufmannBiere TACAS'21] D. Kaufmann and A. Biere. AMulet 2.0 for Verifying Multiplier Circuits. Aceppted at TACAS'21, 2021.
- [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.
- [KaufmannBiereKauers Vampire'19] D. Kaufmann, A. Biere and M. Kauers. SAT, Computer Algebra, Multipliers. In Proc. of Vampire Workshop'19, EPiC series, vol. 71, pages 1–18, EasyChair, 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.
- [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.

## **References III**

- [Mayr STACS'89] E. Mayr. Membership in polynomial ideals over Q is exponential space complete. In Proc. of STACS'89, pages 400–406, Springer, 1989
- [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.
- [RevSCA] 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.
- [RevSCA-2.0] A. Mahzoon, D. Große and R. Drechsler. RevSCA-2.0. https://github.com/amahzoon/RevSCA-2.0
- [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**

[RitircBiereKauers SC2'18] D. Ritirc, A. Biere and M. Kauers. A Practical Polynomial Calculus for Arithmetic Circuit Verification. In Proc. of SC'2, pages 61–76, CEUR-WS, 2018.

[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.

[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.