A PRACTICAL POLYNOMIAL CALCULUS FOR ARITHMETIC CIRCUIT VERIFICATION

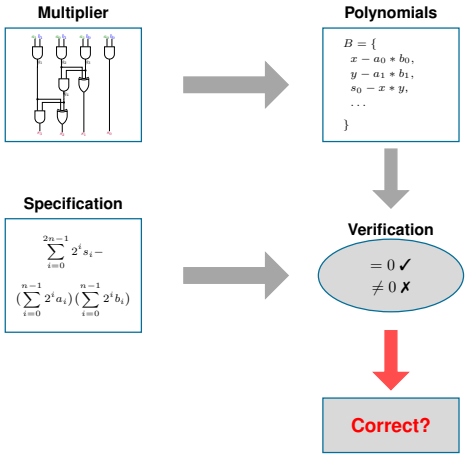
Daniela Ritirc, Armin Biere and Manuel Kauers
Johannes Kepler University
Linz, Austria

SC-Square Workshop 2018
July 11, 2018
Oxford, United Kingdom

JOHANNES KEPLER UNIVERSITY LINZ

RiSE
Rigorous Systems Engineering

Motivation



Problem: Verification might not be error free

Goal: Validate result of verification process

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

Contribution:

- Proof format based on polynomial calculus
- Independent proof checker
- Validate previous verification results

Related Work

- **Reasoning with polynomial equations**
  - D. Kapur. Using **Gröbner Bases** to **Reason** About Geometry Problems. In JSC, 1986.
- **Algebraic proof systems**
  - M. Clegg, J. Edmonds and R. Impagliazzo. Using the **Groebner basis algorithm** to find proofs of unsatisfiability. In STOC, 1996.
  - P. Beame, R. Impagliazzo, J. Krajícek, T. Pitassi and P. Pudlák. Lower bounds on **Hilbert's Nullstellensatz** and propositional proofs. In Proc. London Math. Soc., 1996.
- **Proof complexity**
  - R. Impagliazzo, P. Pudlák and J. Sgall. **Lower bounds** for the polynomial calculus and the Gröbner basis algorithm. In Computational Complexity, 1999.
  - M. Lauria and J. Nordström. **Graph Colouring** is Hard for Algorithms Based on Hilbert's Nullstellensatz and Gröbner Bases. In LIPIcs, 2017.

Preliminaries

Let $f \in \mathbb{F}[X]$.

Polynomial function. $X \mapsto f(X)$

Polynomial equation. $f(X) = 0$ solutions are the roots of f

Ideal. A nonempty subset $I \subseteq \mathbb{F}[X]$ is called an ideal if

$$\forall\, p, q \in I : p + q \in I \qquad \text{and} \qquad \forall\, p \in \mathbb{F}[X]\ \forall\, q \in I : pq \in I$$

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

$$I = \{q_1 p_1 + \cdots + q_m p_m \mid q_1, \ldots, q_m \in \mathbb{F}[X]\} = \langle P \rangle$$

Polynomial Calculus

Let $G \subseteq \mathbb{F}[X]$ and $f \in \mathbb{F}[X]$.

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

Addition	$\dfrac{p_i \qquad p_j}{p_i + p_j}$	p_i, p_j appearing earlier in the proof or are contained in G
Multiplication	$\dfrac{p_i}{qp_i}$	p_i appearing earlier in the proof or is contained in G and $q \in \mathbb{F}[X]$ being arbitrary

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

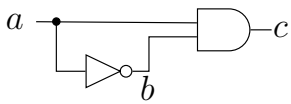
Refutation: $G \vdash 1$ or $1 \in \langle G \rangle$.

Gröbner Basis

Because of Gröbner bases the polynomial calculus is decidable:

- Buchberger's algorithm computes a Gröbner basis $G = \{g_1, \ldots, g_m\}$ for the ideal $\langle H \rangle$ generated by H.
- Given a Gröbner basis G, there is a computable function $\mathrm{red}_G \colon \mathbb{F}[X] \to \mathbb{F}[X]$ such that $\forall\, f \in \mathbb{F}[X] : \mathrm{red}_G(f) = 0 \iff f \in \langle G \rangle$.

Example



$$
\begin{aligned}
G &= \{ \quad -b+1-a, && b=\neg a \\
&\qquad -c+ab, && c=a\wedge b=a\wedge\neg a \\
&\qquad a^2-a,\ b^2-b,\ c^2-c\} && a,b,c\in\mathbb{B} \\
f &= c
\end{aligned}
$$

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

$$
\begin{array}{lcl}
 & * & \dfrac{-b+1-a}{-ab+a-a^2} \\
+ & & \dfrac{a^2-a}{\quad} \\
 & + & \dfrac{-ab}{\quad} \qquad \dfrac{-c+ab}{\quad} \\
 & & * \ \dfrac{-c}{c}
\end{array}
$$

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

Extended Calculus

Radical $\dfrac{p_i^m}{p_i}$ $m \in \mathbb{N} \setminus \{0\}$ and p_i^m appearing earlier in the proof or is contained in G.

If f can be deduced from polynomials $g \in G$, we write $G \vdash^+ f$ or $f \in \sqrt{\langle G \rangle}$.

Radical ideal. $\sqrt{\langle G \rangle} = \{f \in \mathbb{F}[X] \mid G \vdash^+ f\} = \{f \in \mathbb{F}[X] \mid f^m \in \langle G \rangle \text{ for } m \in \mathbb{N} \setminus \{0\}\}$

The extended calculus $\vdash^+$ is decidable. It can be reduced to $\vdash$ with the Rabinowitsch trick:

$$f \in \sqrt{\langle G \rangle} \iff 1 \in \langle G \cup \{yf - 1\} \rangle \quad \text{or} \quad G \vdash^+ f \iff G \cup \{yf - 1\} \vdash 1$$

Models

Let $G \subseteq \mathbb{F}[X]$ and $f \in \mathbb{F}[X]$.

- A **model** for G is a point $u = (u_1, \ldots, u_n) \in \mathbb{F}^n$ such that for all $g \in G$ it holds that $g(u_1, \ldots, u_n) = 0$.
- We write $G \models f$ if every model for G is also a model for $\{f\}$.
- For an algebraically closed field $\mathbb{F}$ we can deduce with Hilbert's Nullstellensatz that $G \models f \iff G \vdash^+ f$.
- PC including the radical rule is correct ("$\Leftarrow$") and complete ("$\Rightarrow$").
- In combination with Rabinowitsch's trick, we can decide the existence of models and produce certificates for the non-existence of models.

Models in $\{0,1\}^n$

For our applications, only models $u \in \{0,1\}^n \subseteq \mathbb{F}^n$ matter.

- Let us write $G \models_{\mathsf{bool}} f$ if every model $u \in \{0,1\}^n$ of G is also a model of $\{f\}$.
- We have $G \models_{\mathsf{bool}} f \iff G \cup B \models f$, where $B = \{x_i(x_i - 1) : i = 1, \ldots, n\}$.
- $G \cup B \models f \iff G \cup B \vdash^+ f$ holds also when $\mathbb{F}$ is not algebraically closed.
- The finiteness of $\{0,1\}^n$ also implies that $G \cup B \vdash^+ f \iff G \cup B \vdash f$ (Seidenberg's lemma).

Practical Algebraic Calculus

We translate the polynomial calculus into a more concrete proof format:

- For correctness it is important to know how the polynomials in the proof where derived
- Usually known $\rightarrow$ store this information

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 ::= ‘+’ | ‘-’

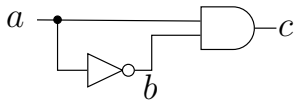
polynomial ::= [‘-’] monomial (operator monomial)*

given ::= (polynomial ‘;’)*

rule ::= (‘+’ | ‘*’) ‘:’ polynomial ‘,’ polynomial ‘,’ polynomial ‘;’

proof ::= (rule ‘;’)*

Example - PAC



$$G = \{ \; -b+1-a, \; -c+ab, \; a^2-a, \; b^2-b, \; c^2-c \}$$

$$f = c$$

```
* : -b+1-a,        a,          -a*b+a-a^2;
+ : -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 $o = " + "$
- Refutation check: at least one p_i is a non-zero constant

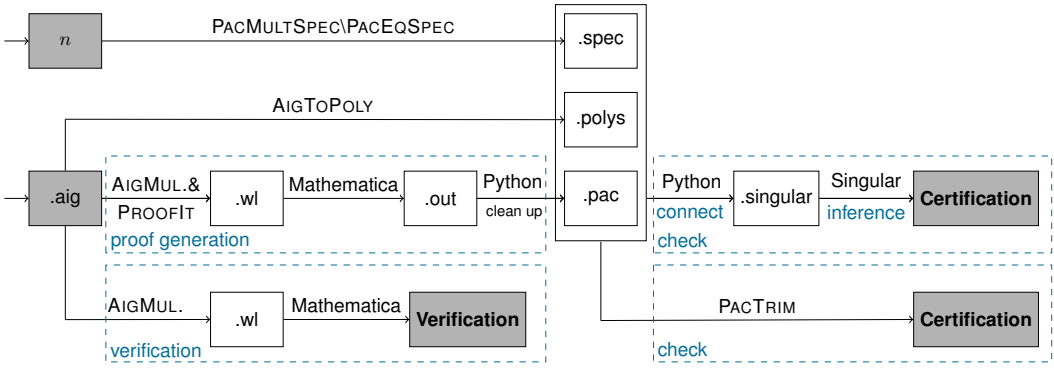
Proof Checking Algorithm

input G sequence of given polynomials
$r_1 \cdots r_k$ sequence of PAC proof rules

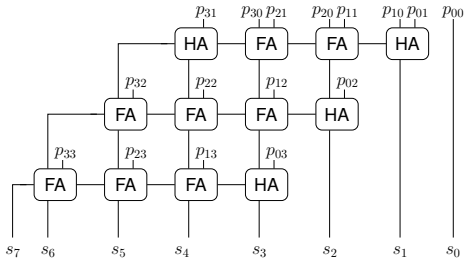
output "incorrect", "correct-proof", or "correct-refutation"

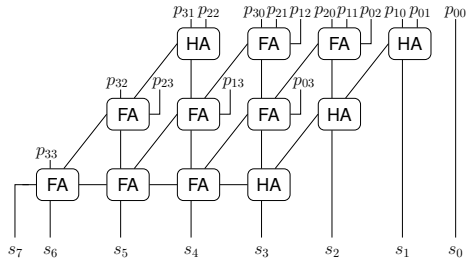
$P_0 \leftarrow G$
for $i \leftarrow 1 \ldots k$
  let $r_i = (o_i, v_i, w_i, p_i)$
  case $o_i = +$
    if $v_i \in P_{i-1} \ \wedge \ w_i \in P_{i-1} \ \wedge \ p_i = v_i + w_i$ **then** $P_i \leftarrow \text{append}(P_{i-1}, p_i)$
    else return "incorrect"
  case $o_i = *$
    if $v_i \in P_{i-1} \ \wedge \ p_i = v_i * w_i$ **then** $P_i \leftarrow \text{append}(P_{i-1}, p_i)$
    else return "incorrect"
for $i \leftarrow 1 \ldots k$
  if p_i is a non zero constant polynomial **then return** "correct-refutation"
return "correct-proof"

Toolflow


n
PacMultSpec\PacEqSpec
.spec
AigToPoly
.polys
.aig
AigMul.&
ProofIt
.wl
Mathematica
.out
Python
clean up
.pac
proof generation
Python
connect
.singular
Singular
inference
Certification
check
AigMul.
.wl
Mathematica
Verification
verification
PacTrim
Certification
check

Correctness and Equivalence Checking





Tailored heuristics:

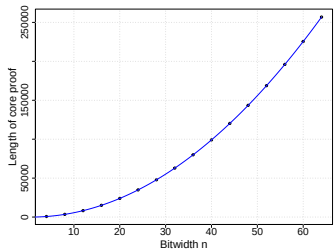
- Split verification problem into sub-problems $\rightarrow$ incremental verification [FMCAD'17]
- Rewrite and simplify Gröbner basis $\rightarrow$ adder-rewriting [DATE'18]

Verifying Correctness of FMCAD'17 and DATE'18 Results

n	mult	option	verification	proof generation	PACTRIM	Python/ Singular	length	core
16	btor	inc	4	70	0	4	14998	64%
16	btor	inc-add	1	37	0	4	12738	61%
16	btor	noninc	4	78	0	10	14966	64%
32	btor	inc	44	1631	1	83	63254	64%
32	btor	inc-add	7	801	1	67	53122	61%
32	btor	noninc	65	1811	5	551	63190	64%
64	btor	inc	622	49638	4	5125	259606	63%
64	btor	inc-add	121	22378	4	4650	216834	61%
64	btor	noninc	MO	MO	-	-	-	-
16	sparrc	inc	8	134	0	7	17445	62%
16	sparrc	inc-add	1	112	0	20	17804	63%
16	sparrc	noninc	11	2696	0	17	17413	62%
32	sparrc	inc	104	3582	1	175	73301	62%
32	**sparrc**	**inc-add**	**8**	**2611**	**2**	**457**	**75244**	**63%**
32	sparrc	noninc	351	TO	-	-	-	-
64	sparrc	inc	1575	TO	-	-	-	-
64	sparrc	inc-add	133	80906	12	EE	309164	62%
64	sparrc	noninc	MO	-	-	-	-	-

Equivalence Checking - DATE'18

n	mult	verification	proof generation	PACTRIM	Python/ Singular	length	core
16	btor-btor	2	75	1	14	25410	59%
32	btor-btor	27	1632	3	277	106114	59%
64	btor-btor	502	45155	15	EE	433410	59%
16	btor-sparrc	3	148	1	48	30476	61%
32	btor-sparrc	28	3456	7	1011	128236	60%
16	sparrc-sparrc	2	223	2	82	35542	61%
32	sparrc-sparrc	29	5363	11	1899	150358	61%



Proof length: $\mathcal{O}(n^2)$

[BeameLiew-CAV'17]: $\mathcal{O}(n^7 \log n)$
upper bound for resolution proof size

Conclusion & Future Work

Conclusion:

- Practical proof checking for algebraic reasoning
- Translation of polynomial calculus to a more concrete proof format
- Certification of correctness of certain multipliers

Future Work:

- Speed-up proof generation
- Explore connection between PAC and clausal proof systems (RUP, DRAT)

A PRACTICAL POLYNOMIAL CALCULUS FOR ARITHMETIC CIRCUIT VERIFICATION

Daniela Ritirc, Armin Biere and Manuel Kauers
Johannes Kepler University
Linz, Austria

SC-Square Workshop 2018
July 11, 2018
Oxford, United Kingdom

JOHANNES KEPLER UNIVERSITY LINZ

RiSE
Rigorous Systems Engineering

Engineering

Use `PolynomialReduce` for reduction $f \xrightarrow{G} 0$ and deriving co-factors $h_1, ..., h_k$ such that $f = h_1 g_1 + \ldots + h_k g_k$.

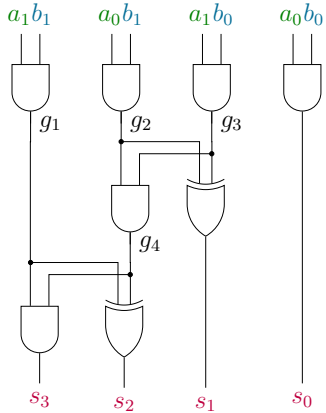
First we generate a multiplication proof rule for each product $h_i g_i$.

$$* : g_1, h_1, h_1 g_1; \qquad \cdots \qquad * : g_k, h_k, h_k g_k;$$

These products are now simply added together as follows:

$$\begin{array}{lll} + : & h_1 g_1, & h_2 g_2, \quad h_1 g_1 + h_2 g_2; \\ + : & h_1 g_1 + h_2 g_2, & h_3 g_3, \quad h_1 g_1 + h_2 g_2 + h_3 g_3; \\ & & \vdots \\ + : & h_1 g_1 + \ldots + h_{k-1} g_{k-1}, & h_k g_k, \quad f; \end{array}$$

Circuit Verification



$$
\begin{aligned}
G = \{ \quad & -s_3 + g_1 g_4, \\
& -s_2 + g_1 + g_4 - 2g_1 g_4, \\
& -g_4 + g_2 g_3, \\
& -s_1 + g_2 + g_3 - 2g_2 g_3, \\
& -g_1 + a_1 b_1, \\
& -g_2 + a_0 b_1, \\
& -g_3 + a_1 b_0, \\
& -s_0 + a_0 b_0, \\
& -a_1^2 + a_1, -a_0^2 + a_0, \\
& -b_1^2 + b_1, -b_0^2 + b_0 \}
\end{aligned}
$$

For all possible $a_i, b_i \in \mathbb{B}$: $(2a_1 + a_0) * (2b_1 + b_0) = 8s_3 + 4s_2 + 2s_1 + s_0$?