THE PROOF CHECKERS PACHECK AND PASTÈQUE FOR THE PRACTICAL ALGEBRAIC CALCULUS

Daniela Kaufmann, Mathias Fleury and Armin Biere
Johannes Kepler University
Linz, Austria

FMCAD 2020
September 24, 2020
Online
Formal Verification using Computer Algebra

Renewed interest in recent years


Algebraic reasoning in combination with SAT solving

Basic Idea of Algebraic Approach

System

Polynomials

Specification

Implication

\[
B = \{ \\
\begin{align*}
x - a_0 & \cdot b_0, \\
y - a_1 & \cdot b_1, \\
s_0 - x & \cdot y, \\
\ldots
\end{align*}
\}
\]

\[
\begin{align*}
\sum_{i=0}^{n-1} 2^i s_i &= 0 \\
\left(\sum_{i=0}^{n-1} 2^i a_i\right) \left(\sum_{i=0}^{n-1} 2^i b_i\right) &\neq 0
\end{align*}
\]

\[
= 0 \checkmark \\
\neq 0 \times
\]
Basic Idea of Algebraic Approach

Multiplier

Polynomials

Specification

Implication

\[ B = \{ \]
\[ x - a_0 \ast b_0, \]
\[ y - a_1 \ast b_1, \]
\[ s_0 - x \ast y, \]
\[ \ldots \] \]

\[ = 0 \checkmark \]
\[ \neq 0 \times \]
Basic Idea of Algebraic Approach

Multiplier

Polynomials

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

Specification

\[ \sum_{i=0}^{2n-1} 2^i s_i - \]

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

Ideal Membership

\[ = 0 \checkmark \]
\[ \neq 0 \times \]
Motivation

Multiplier

Polynomials

Problem: Verification might not be error free

Specification

Ideal Membership

Correct?
Motivation

Polynomials

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

Ideal Membership

$$= 0 \checkmark$$

$$\neq 0 \times$$

Problem: Verification might not be error free

Goal: Validate result of verification process

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

⇒ SC’2 2018: Practical Algebraic Calculus (PAC) based on polynomial calculus
**Motivation**

**Multiplier**

**Specification**

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

**Polynomials**

\[
B = \{ \\
\quad x - a_0 \cdot b_0, \\
\quad y - a_1 \cdot b_1, \\
\quad s_0 - x \cdot y, \\
\quad \ldots \\
\}
\]

**Ideal Membership**

\[
= 0 \checkmark \\
\neq 0 \times
\]

**Problem:** Verification might not be error free

**Goal:** Validate result of verification process

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

⇒ SC’2 2018: Practical Algebraic Calculus (PAC) based on polynomial calculus

**Contribution:**

- Extending PAC: indexing, deletion and extension rules
- Proof checker PACHECK and PASTÈQUE
Ideal

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

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

Ideal membership problem. Given a polynomial $f \in \mathbb{Z}[X]$ and a (finite) set of polynomials $P \subseteq \mathbb{Z}[X]$, 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 $P$. 
\[ P = -b + 1 - a, \quad b = \neg a \]
\[ -c + a \cdot b, \quad c = a \land b = a \land \neg a \]
\[ -a^2 + a \]
\[ a = \bot \lor a = \top \]
\[ \text{Spec} = c \]
\[ c = \bot \]

\[ * : -b + 1 - a, \quad a, \quad -a \cdot b + a - a^2; \]
\[ * : -a^2 + a, \quad -1, \quad a^2 - a; \]
\[ + : -a \cdot b + a - a^2, \quad a^2 - a, \quad -a \cdot b; \]
\[ + : -a \cdot b, \quad -c + a \cdot b, \quad -c; \]
\[ * : -c, \quad -1, \quad c; \]

\[ \forall p, q \in I : p + q \in I \quad \text{and} \quad \forall p \in \mathbb{Z}[X] \forall q \in I : pq \in I \]
 Practical Algebraic Calculus - SC’2 2018

\[
P = -b+1-a, \quad b = \neg a \\
-\neg c + a*\neg b, \quad c = a \land \neg a \\
-\neg a^2 + a, \quad a = \bot \lor \neg a = \top \\
\text{Spec} = c \quad c = \bot
\]

\[
\begin{align*}
* : & \quad -b+1-a, \quad a, \quad -a*b+a-a^2; \\
* : & \quad -a^2+a, \quad -1, \quad a^2-a; \\
+ : & \quad -a*b+a-a^2, \quad a^2-a, \quad -a*b; \\
+ : & \quad -a*b, \quad -c+a*b, \quad -c; \\
* : & \quad -c, \quad -1, \quad c;
\end{align*}
\]

Can be checked by our older proof checker PACTRIM.
1. Boolean Variables

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

\[ P = \neg b + 1 - a, \]
\[ \neg c + a \cdot b, \]
\[ \neg a^2 + a \]

\[ \text{Spec} = c \]

\[
\begin{align*}
* : & \quad \neg b + 1 - a, & a, & -a \cdot b; \\
+ : & \quad -a \cdot b, & \neg c + a \cdot b, & -c; \\
* : & \quad -c, & -1, & c;
\end{align*}
\]
Introduce indices to reduce proof size.

\[ P = 1 -b+1-a; \]
\[ 2 -c+a*b; \]
\[ \text{Spec} = c \]

3  * 1,  a,  -a*b;
4  + 3,  2,  -c;
5  * 4,  -1,  c;
3. Deletion Rule

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

\[ P = \begin{align*} 1 & -b+1-a; \\ 2 & -c+a*b; \end{align*} \]

\[ \text{Spec} = c \]

\[ \begin{align*} 3 & * 1, \ a, \ -a*b; \\ 1 & d; \\ 4 & + 3, \ 2, \ -c; \\ 2 & d; \\ 3 & d; \\ 5 & * 4, \ -1, \ c; \end{align*} \]
4. Extension Rule

The extension rule allows to add model preserving polynomials to the constraint set.

\[
\begin{array}{c}
\bar{x} \lor \bar{y} \\
y \lor z \\
\end{array}
\quad
\begin{array}{c}
\bar{x} \lor z
\end{array}
\]
4. Extension Rule

The extension rule allows to add model preserving polynomials to the constraint set.

\[
\frac{xy}{x(1 - z)} \left(1 - y \right) \left(1 - z \right)
\]
4. Extension Rule

The extension rule allows to add model preserving polynomials to the constraint set.

\[
\begin{align*}
xy \\
(1 - y)(1 - z) \\
x(1 - z)
\end{align*}
\]

\[
P = 1 \ x*y; \\
2 \ y*z-y-z+1;
\]

\[
\text{Spec} = -x*z+x
\]

\[
3 = f, \ -z+1; \\
4 * 3, \ y-1, \ -f*y+f-y*z+y+z-1; \\
5 + 2, \ 4, \ -f*y+f;
\]

\[
\text{EXT}(i, v, p) \quad (X, P) \Rightarrow (X \cup \{v\}, P(i \mapsto -v + p))
\]

provided that \( P(i) = \perp \) and \( v \notin X \) and \( p \in \mathbb{Z}[X]/\langle B(X) \rangle \),

and \( p^2 - p \equiv 0 \mod \langle B(X) \rangle \).
4. Extension Rule

The extension rule allows to add model preserving polynomials to the constraint set.

\[
\frac{xy}{x(1 - z)} \frac{(1 - y)(1 - z)}{\text{Spec} = -x*z+x}
\]

\[
P = 1 \ x*y;
2 \ y*z-y-z+1;
\]

\[
3 = f, -z+1;
4 \ast 3, \ y-1, -f*y+f-y*z+y+z-1;
5 + 2, 4, -f*y+f;
6 \ast 1, \ f, f*x*y;
7 \ast 5, \ x, -f*x*y+f*x;
8 + 6, 7, f*x;
9 \ast 3, \ x, -f*x-x*z+x;
10 + 8, 9, -x*z+x;
\]
PACHECK

- 1700 lines of C code
- supports new and old PAC format

- PACHECK reads three input files `<input>`, `<proof>`, and `<target>`.
- Verifies that the polynomial in `<target>` is contained in the ideal generated by the polynomials in `<input>` using the rules provided in `<proof>`.

Polynomial arithmetic is implemented from scratch.

Coefficients are represented using the GMP library.

Terms are ordered linked lists of variables.
PACHECK

- 1700 lines of C code
- supports new and old PAC format
- PACHECK reads three input files `<input>`, `<proof>`, and `<target>`.
- Verifies that the polynomial in `<target>` is contained in the ideal generated by the polynomials in `<input>` using the rules provided in `<proof>`.

- Polynomial arithmetic is implemented from scratch.
- A polynomial is represented as a linked list of monomials:
  - Coefficients are represented using the GMP library.
  - Terms are ordered linked lists of variables.
Variable ordering

- The order of the variables has an enormous effect on the memory usage:
  - Terms are internally shared
Variable ordering

- The order of the variables has an enormous effect on the memory usage:
  - Terms are internally shared

Assume we want to represent the terms $uxy$ and $vxy$.

$v > u > x > y$

$x > u > y > v$
Variable ordering

- The order of the variables has an enormous effect on the memory usage:
  - Terms are internally shared
- Example: 7 mio. rules, 50% memory increase using a different ordering

Implemented Orderings:

- Default: Variables are ordered using `strcmp`.
- Alternative: Same variable ordering as used in input files.
- Both orderings can also be reversed.
PASTÈQUE

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 8 000 lines of code
Refinement

\[
\text{no error} \implies \text{spec} \in \langle P \rangle \land \langle P \rangle|_X \subseteq \langle P_0 \rangle|_X
\]
Refinement

\[
\text{no error} \implies \text{spec } \in \langle P \rangle \land \langle P \rangle_{X} \subseteq \langle P_{0} \rangle_{X}
\]

Entering nondeterminism monad

check rule by rule for the side conditions
Refinement

\[
\text{no error} \rightarrow \text{spec } \in \langle P \rangle \wedge \langle P \rangle\big|_X \subseteq \langle P_0 \rangle\big|_X
\]

- Entering nondeterminism monad
- Check rule by rule for the side conditions
- Executable polynomials
- Use multisets for polynomials
Refinement

no error $\implies$ \( \text{spec} \in \langle P \rangle \land \langle P \rangle \big|_X \subseteq \langle P_0 \rangle \big|_X \)

- Entering nondeterminism monad
- check rule by rule for the side conditions
- Executable polynomials
- Use multisets for polynomials
- Specify all operations
- Use multisets for lists

Specify all operations

Automatic efficient imperative code

Imperative code

Entering nondeterminism monad

Use multisets for lists

Use multisets for polynomials

Executable polynomials

check rule by rule for the side conditions

no error $\implies$ spec $\in \langle P \rangle \land \langle P \rangle \big|_X \subseteq \langle P_0 \rangle \big|_X$
Refinement

\[
\text{no error} \quad \implies \quad \text{spec} \in \langle P \rangle \land \langle P \rangle_{X} \subseteq \langle P_0 \rangle_{X}
\]

- Entering nondeterminism monad
- Check rule by rule for the side conditions
- Executable polynomials
- Use multisets for polynomials
- Specify all operations
- Use multisets for lists
- Automatic efficient imperative code
- Imperative code
Refinement

- Translation to Standard ML is trusted
- Parser and plumbing (including answer printing) also trusted
Alternative, \texttt{uloop} variant:

- same functions to check the steps
- but: hand-written (trusted) version of the loop that iterates through the steps

Advantage: more memory efficient
Tool Demonstration

\[
x y \frac{(1 - y)(1 - z)}{x(1 - z)} = \begin{align*}
P &= 1 \times y; \\
2 \times z - y - z + 1;
\end{align*}
\]

Spec = -x*z + x

3 = f, -z + 1;
4 * 3, y - 1, -f*y + f*y*z + y - z - 1;
5 + 2, 4, -f*y + f;
2 d;
4 d;
6 * 1, f, f*x*y;
1 d;
7 * 5, x, -f*x*y + f*x;
5 d;
8 + 6, 7, f*x;
6 d;
7 d;
9 * 3, x, -f*x - x*z + x;
3 d;
10 + 8, 9, -x*z + x;
Pacheck Version 001
Practical Algebraic Calculus Proof Checker
Copyright (C) 2020, Daniela Kaufmann, Johannes Kepler University Linz
compressed mode with indices assumed
sorting according to strcmp
checking target enabled
reading target polynomial from 'ex1.target'
read 8 bytes from 'ex1.target'
reading original polynomials from 'ex1.input'
found 2 original polynomials in 'ex1.input'
read 20 bytes from 'ex1.input'
reading polynomial algebraic calculus proof from 'ex1.proof'
found and checked 8 inferences in 'ex1.proof'
read 219 bytes from 'ex1.proof'
found 1 target polynomial inference
proof length 10 (number of polynomials)
proof size 25 (on average 2.5 terms per polynomial)
proof degree 3 (internal maximum degree 3)
searched 32 inferences 0.1 average collisions
10 inferences, 3.2 average searches
original inferences 2 (20% of total rules)
inference rules 8 (80% of total rules)
addition inference rules 3 (38% of inference rules)
multiplication inference rules 4 (50% of inference rules)
extension rules 1 (12% of inference rules)
deletion inference rules 3 (30% of total rules)
maximum 9 of total 10 terms (90%)
searched 52 terms 81% hits 0.3 average collisions
maximum 2261 bytes allocated (0.0 MB)
maximum resident set size 5922816 bytes (5.6 MB)
process time 0.001 seconds
TARGET CHECKED
pasteque exi.{input,proof,target}
c polys parsed
c ******************
c pac parsed
c spec parsed
c Now checking
s SUCCESSFULL

c

c ***** stats *****
c parsing polys file init (nonGC): 0.000 s = 0.000 s (usr) 0.000 s (sys)
c parsing pac file init (nonGC): 0.000 s = 0.000 s (usr) 0.000 s (sys)
c full init (nonGC): 0.000 s = 0.000 s (usr) 0.000 s (sys)
c time solving (nonGC): 0.000 s = 0.000 s (usr) 0.000 s (sys)
c time GC: 0.000 s = 0.000 s (usr) 0.000 s (sys)
c time solving(full): 0.000 s

c Overall (nonGC): 0.000 s = 0.000 s (usr) 0.000 s (sys)
c overall GC: 0.000 s = 0.000 s (usr) 0.000 s (sys)
c Overall(full): 0.000 s
vim ex1.proof
3 = f, -z+1;
4 * 3, y-1, -f+y+z+y+z-1;
5 + 2, 4, -f*y+f;
2 d;
4 d;
6 * 1, f, f*x
1 d;
7 * 5, x, -f*x*y+f*x;
8 + 6, 7, f*x;
9 * 3, x, -f*x-x+z+x;
10 + 8, 9, -x*z+x;
3 = f, -x+1;
4 * 3, y-1, -f*y+f-y*z+y+z-1;
5 + 2, 4, -f*y+f;
2 d;
4 d;
6 * 1, f, f*x + y;
1 d;
7 * 5, x, -f*x*y+f*x;
8 + 6, 7, f*x;
9 * 3, x, -f*x-x*z+x;
10 + 8, 9, -x*z+x;
**pacheck ex1.{input,proof,target}**

[pacheck] Pacheck Version 001
[pacheck] Practical Algebraic Calculus Proof Checker
[pacheck] Copyright (C) 2020, Daniela Kaufmann, Johannes Kepler University Linz
[pacheck] compressed mode with indices assumed
[pacheck] sorting according to strcmp
[pacheck] checking target enabled
[pacheck] reading target polynomial from 'ex1.target'
[pacheck] read 8 bytes from 'ex1.target'
[pacheck] reading original polynomials from 'ex1.input'
[pacheck] found 2 original polynomials in 'ex1.input'
[pacheck] read 8 bytes from 'ex1.target'
[pacheck] reading original polynomials from 'ex1.input'
[pacheck] found 2 original polynomials in 'ex1.input'
[pacheck] read 20 bytes from 'ex1.input'
[pacheck] reading polynomial algebraic calculus proof from 'ex1.proof'

*** 'pacheck' error in polynomial multiplication rule 4 with index 6 in 'ex1.proof' line 7: conclusion polynomial:

\[ f \times x + y \]

does not match expected result:

\[ f \times x + y \]
Evaluation

Comparison between:

PACTRIM the older version
PACHECK with no delete, no index, everything
PASTÈQUE with and without u1oop
Complex adder: Replace it by simpler adder [DATE20,Kaufmann et al.] via extensions.
## Evaluation Results

<table>
<thead>
<tr>
<th>multiplier</th>
<th>( n )</th>
<th>steps ((10^6))</th>
<th>ext deg</th>
<th>( \text{PACTRIM} )</th>
<th>( \text{PACHCK} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no index sec MB</td>
<td>no delete sec MB</td>
</tr>
<tr>
<td>btor</td>
<td>128</td>
<td>0.4</td>
<td>0</td>
<td>3</td>
<td>10   105</td>
</tr>
<tr>
<td>btor</td>
<td>256</td>
<td>1.6</td>
<td>0</td>
<td>3</td>
<td>60   459</td>
</tr>
<tr>
<td>btor</td>
<td>512</td>
<td>6.3</td>
<td>0</td>
<td>3</td>
<td>395  2066</td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>128</td>
<td>0.6</td>
<td>0</td>
<td>4</td>
<td>16   156</td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>256</td>
<td>2.3</td>
<td>0</td>
<td>4</td>
<td>92   687</td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>512</td>
<td>9.4</td>
<td>0</td>
<td>4</td>
<td>587  3107</td>
</tr>
<tr>
<td>sp-ar-cl</td>
<td>32</td>
<td>1.6</td>
<td>2</td>
<td>256</td>
<td>31   405</td>
</tr>
<tr>
<td>sp-dt-lf</td>
<td>32</td>
<td>0.3</td>
<td>1</td>
<td>1684</td>
<td>3    82</td>
</tr>
<tr>
<td>bp-ct-bk</td>
<td>32</td>
<td>0.2</td>
<td>1</td>
<td>405</td>
<td>2    57</td>
</tr>
<tr>
<td>bp-wt-cl</td>
<td>32</td>
<td>5.6</td>
<td>5</td>
<td>740</td>
<td>242  1716</td>
</tr>
</tbody>
</table>
## Evaluation Results

<table>
<thead>
<tr>
<th>multiplier</th>
<th>$n$</th>
<th>steps (10^6)</th>
<th>ext</th>
<th>deg</th>
<th>PACHECK</th>
<th>PASTÈQUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>default</td>
<td>default</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>sec</td>
<td>MB</td>
</tr>
<tr>
<td>btor</td>
<td>128</td>
<td>0.4</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>92</td>
</tr>
<tr>
<td>btor</td>
<td>256</td>
<td>1.6</td>
<td>0</td>
<td>3</td>
<td>25</td>
<td>364</td>
</tr>
<tr>
<td>btor</td>
<td>512</td>
<td>6.3</td>
<td>0</td>
<td>3</td>
<td>141</td>
<td>1461</td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>128</td>
<td>0.6</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>136</td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>256</td>
<td>2.3</td>
<td>0</td>
<td>4</td>
<td>27</td>
<td>541</td>
</tr>
<tr>
<td>sp-ar-rc</td>
<td>512</td>
<td>9.4</td>
<td>0</td>
<td>4</td>
<td>134</td>
<td>2171</td>
</tr>
<tr>
<td>sp-ar-cl</td>
<td>32</td>
<td>1.6</td>
<td>2081</td>
<td>256</td>
<td>21</td>
<td>353</td>
</tr>
<tr>
<td>sp-dt-lf</td>
<td>32</td>
<td>0.3</td>
<td>1684</td>
<td>46</td>
<td>2</td>
<td>73</td>
</tr>
<tr>
<td>bp-ct-bk</td>
<td>32</td>
<td>0.2</td>
<td>1405</td>
<td>25</td>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>bp-wt-cl</td>
<td>32</td>
<td>5.6</td>
<td>5740</td>
<td>764</td>
<td>181</td>
<td>1428</td>
</tr>
</tbody>
</table>
Conclusion & Future Work

- **Proof Checker:**
  - **PACheck** is 30-80% faster than older proof checker.
  - **PASTÈQUE** is $4 \times$-slower, but fully verified.

- **New PAC format:**
  - No need to add extension variables to input polynomials anymore.
  - Indices and assumption of Boolean variables reduces the proof size.
  - Deletion rules reduce the memory usage.

- Investigate whether lifting the restrictions on $E_X T$ is useful.

- Explore connection to Nullstellensatz Proofs to gain shorter proofs:
THE PROOF CHECKERS PACHECK AND PASTÈQUE FOR THE PRACTICAL ALGEBRAIC CALCULUS

Daniela Kaufmann, Mathias Fleury and Armin Biere
Johannes Kepler University
Linz, Austria

FMCAD 2020
September 24, 2020
Online