RECYCLING ALGEBRAIC PROOF CERTIFICATES

Daniela Kaufmann and **Clemens Hofstadler**

TU Wien, Austria Johannes Kepler University Linz, Austria

SC^2 Workshop
Stuttgart, Germany

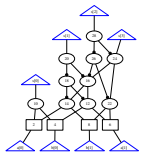
August 2, 2025

TU WIEN Informatics

FWF Austrian Science Fund

Circuit Verification using Algebraic Proofs

Circuit



Polynomials

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

Specification

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

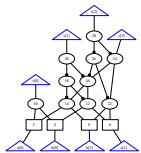
Implication

$= 0$ ✓

$\neq 0$ ✗

Circuit Verification using Algebraic Proofs

Circuit



Polynomials

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

Specification

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

Ideal Membership

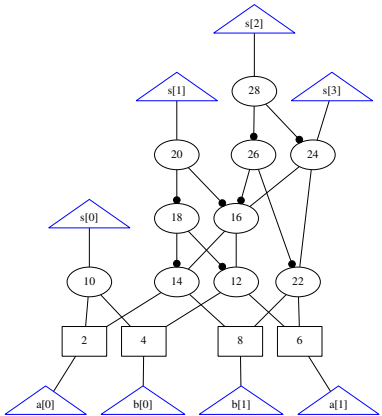
$= 0$ ✓

$\neq 0$ ✗

Circuit Verification using Algebraic Proofs

Gate polynomials $G(C)$.

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



Circuit Verification using Algebraic Proofs

Gate polynomials $G(C)$.

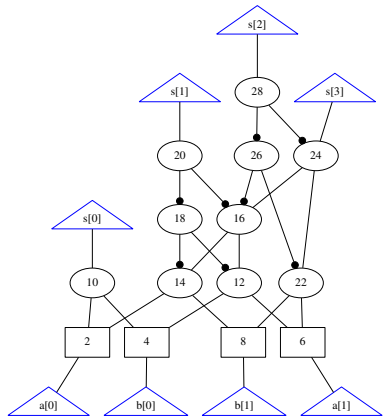
$$-s_3 + l_{24}$$
$$-s_2 + l_{28}$$
$$-s_1 + l_{20}$$
$$-s_0 + l_{10}$$
$$-l_{28} + l_{26}l_{24} - l_{26} - l_{24} + 1$$
$$-l_{26} + l_{22}l_{16} - l_{22} - l_{16} + 1$$
$$-l_{24} + l_{22}l_{16}$$
$$-l_{22} + a_1b_1$$
$$-l_{20} + l_{18}l_{16} - l_{18} - l_{16} + 1$$
$$-l_{18} + l_{14}l_{12} - l_{14}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

Boolean input constraints $B(C)$.

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

Specification $\mathcal{S}$.

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



Circuit Verification using Algebraic Proofs

Gate polynomials $G(C)$.

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

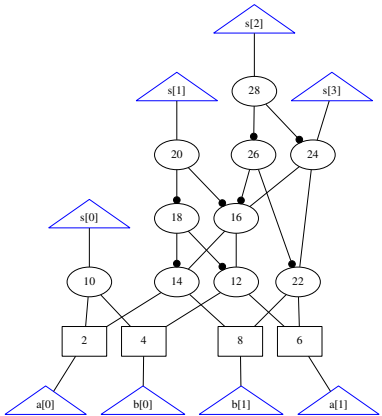
Boolean input constraints $B(C)$.

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

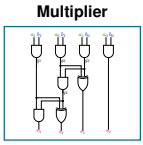
Specification $\mathcal{S}$.

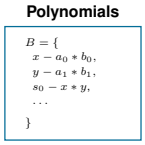
$$8s_3 + 4s_2 + 2s_1 + s_0 - 4b_1 a_1 - 2b_1 a_0 - 2b_0 a_1 - b_0 a_0$$

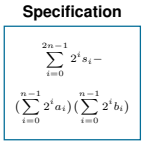
Verification: $\mathcal{S} \in \langle G(C) \cup B(C) \rangle \Leftrightarrow \mathcal{S} \xrightarrow{G(C) \cup B(C)} 0$

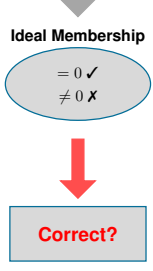


Proof Logging



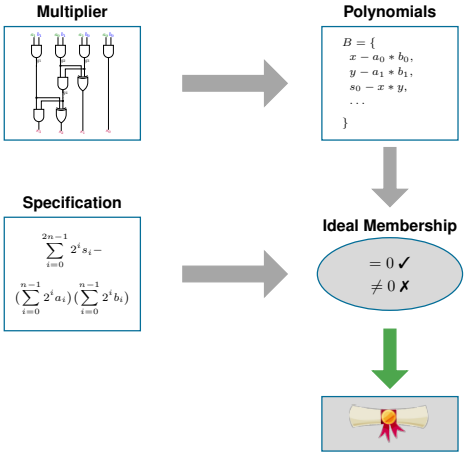






Problem: Verification might not be error free

Proof Logging

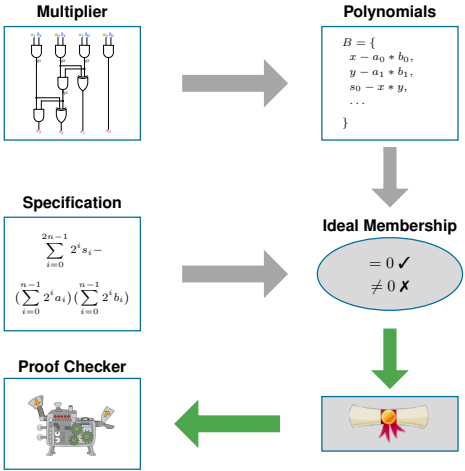


Problem: Verification might not be error free

Goal: Validate result of verification process

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

Proof Logging



Problem: Verification might not be error free

Goal: Validate result of verification process

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

Algebraic Proof Logging

Setting: Let X be a set of variables, $\mathbb{K}$ a field. Let $G \subseteq \mathbb{K}[X]$, $f \in \mathbb{K}[X]$. The goal is to derive a proof certificate validating that $f \in \langle G \rangle$.

Algebraic Proof Logging

Setting: Let X be a set of variables, $\mathbb{K}$ a field. Let $G \subseteq \mathbb{K}[X]$, $f \in \mathbb{K}[X]$. The goal is to derive a proof certificate validating that $f \in \langle G \rangle$.

Disclaimer: Although our proposed idea of proof recycling works in this general setting, our main focus is on applications where all variables X are Boolean.

Hence, we have $B(X) = \{x^2 - x \mid x \in X\} \subseteq G$.

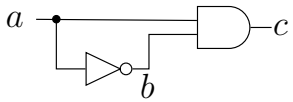
Nullstellensatz Proofs

Input: $f_1, \ldots, f_s, f$
Output: $a_1, \ldots, a_s, r$
$a_1 := 0; \ldots; a_s := 0; r := 0$
$p := f$
WHILE $p \neq 0$ DO
    $i := 1$
    divisionoccurred := false
    WHILE $i \leq s$ AND divisionoccurred = false DO
        IF LT(f_i) divides (p) THEN
            $a_i := a_i + \text{LT}(p)/\text{LT}(f_i)$
            $p := p - (\text{LT}(p)/\text{LT}(f_i))f_i$
            divisionoccurred:= true
        ELSE
            $i := i + 1$
    IF divisionoccurred = false THEN
        $r := r + \text{LT}(p)$
        $p := p - \text{LT}(p)$

- Provides list of co-factors $a_1, \ldots, a_s$.
- Correctness is checked by expanding linear combination $f = \sum a_i f_i$.
- Condensed proof format.
- Not ideal for debugging.

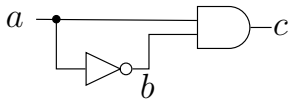
Beame, P., Impagliazzo, R., Krajicek, J., Pitassi, T., Pudlák, P.: Lower Bounds on Hilbert's Nullstellensatz and Propositional Proofs. In: Proc. London Math. Society. vol. s3-73, pp. 1-26 (1996)

Example



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

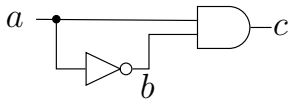
Example



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

$$f = c = a(-b + 1 - a) - 1(-c + ab) + 1(a^2 - a)$$

Example



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

$$f = c = a(-b+1-a) - 1(-c+ab) + 1(a^2-a)$$

$$P = (a, -1, 1)$$

Polynomial Calculus*

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

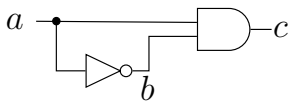
Proof: Sequence $P = (p_1, \ldots p_n)$, where each p_k is obtained by one of the two rules:

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 $q \in \mathbb{K}[X]$ being arbitrary

If $p_n = f$ we have $f \in \langle G \rangle$.

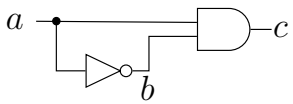
Clegg, M., Edmonds, J., Impagliazzo, R.: Using the Groebner basis algorithm to find proofs of unsatisfiability. In: STOC. pp. 174–183. ACM (1996)

Example



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

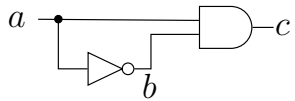
Example



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

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

Example



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

$$
\begin{array}{lcccc}
* & \frac{-b+1-a}{} & & & \\
+ & \frac{-ab+a-a^2}{} & & a^2-a & \\
& + & \frac{-ab}{} & & \frac{-c+ab}{} \\
& & & * \ \frac{-c}{c} &
\end{array}
$$

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

Practical Algebraic Calculus

We translated 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

D. Ritirc, A. Biere, M. Kauers, A Practical Polynomial Calculus for Arithmetic Circuit Verification, SC2-Workshop, 2018.

LPAC

Switch from explicit addition and multiplication rules to linear combinations.

LPAC

AXIOM (i, p)

$$\frac{(X,P) \quad P(i)=\bot \quad p\in\mathbb{K}[X]}{(X\cup\mathrm{Var}(p),P(i\mapsto p))}$$

LPAC

AXIOM (i, p)

$$\frac{(X,P) \qquad P(i)=\bot \qquad p\in\mathbb{K}[X]}{(X\cup\mathrm{Var}(p),P(i\mapsto p))}$$

DELETION (i)

$$\frac{(X,P)}{(X,P(i\mapsto\bot))}$$

LPAC

AXIOM (i, p)

$$\frac{(X,P) \quad P(i)=\bot \quad p\in\mathbb{K}[X]}{(X\cup \mathrm{Var}(p), P(i\mapsto p))}$$

DELETION (i)

$$\frac{(X,P)}{(X,P(i\mapsto \bot))}$$

LINCOMB $(i, (j_1, \ldots, j_n), (q_1, \ldots, q_n), p)$

$$\frac{(X,P) \quad P(j_1)\neq\bot,\ldots,P(j_n)\neq\bot, P(i)=\bot \quad p,q_1,\ldots,q_n\in\mathbb{K}[X] \quad p=q_1\cdot P(j_1)+\cdots+q_n\cdot P(j_n) \bmod \langle B(X)\rangle}{(X,P(i\mapsto p))}$$

LPAC

AXIOM (i, p)

$$\frac{(X,P) \quad P(i)=\bot \quad p\in\mathbb{K}[X]}{(X\cup \mathrm{Var}(p), P(i\mapsto p))}$$

DELETION (i)

$$\frac{(X,P)}{(X,P(i\mapsto \bot))}$$

LINCOMB $(i, (j_1, \ldots, j_n), (q_1, \ldots, q_n), p)$

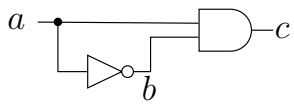
$$\frac{(X,P) \quad P(j_1)\neq\bot,\ldots,P(j_n)\neq\bot, P(i)=\bot \quad p,q_1,\ldots,q_n\in\mathbb{K}[X] \quad p=q_1\cdot P(j_1)+\cdots+q_n\cdot P(j_n) \bmod \langle B(X)\rangle}{(X,P(i\mapsto p))}$$

EXT (i, v, q)

$$\frac{(X,P) \quad P(i)=\bot \quad v\notin X \quad q\in\mathbb{K}[X] \quad q^2-q\in\langle B(X)\rangle}{(X\cup\{v\}, P(i\mapsto -v+q))}$$

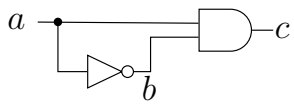
D. Kaufmann, M. Fleury, A. Biere, M. Kauers, Practical Algebraic Calculus and Nullstellensatz with the Checkers Pacheck and Pastèque and Nuss-Checker FMSD, 2022.

LPAC



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

LPAC

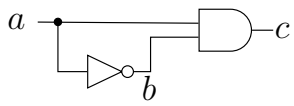


$$
\begin{aligned}
G &= \{ \quad -b + 1 - a, && b = \neg a \\
&\qquad\; -c + ab, && c = a \wedge b = a \wedge \neg a \\
&\qquad\; a^2 - a, \} && a \in \mathbb{B} \\
f &= \quad c
\end{aligned}
$$

LPAC simulates PC

```
1   a -b+1-a;
2   a -c+a*b;
3   % 1*(a), -a*b;
1   d;
4   % 3+2, -c;
2   d;
3   d;
5   % 4*(-1), c;
```

LPAC



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

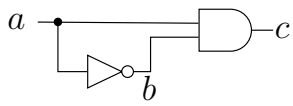
LPAC simulates PC

```
1   a -b+1-a;
2   a -c+a*b;
3   % 1*(a), -a*b;
1   d;
4   % 3+2, -c;
2   d;
3   d;
5   % 4*(-1), c;
```

LPAC

```
1   a -b+1-a;
2   a -c+a*b;
3   % 1*(a)+2, -c;
1   d;
2   d;
4   % 4*(-1), c;
```

LPAC



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

LPAC simulates PC

```
1   a -b+1-a;
2   a -c+a*b;
3   % 1*(a), -a*b;
1   d;
4   % 3+2, -c;
2   d;
3   d;
5   % 4*(-1), c;
```

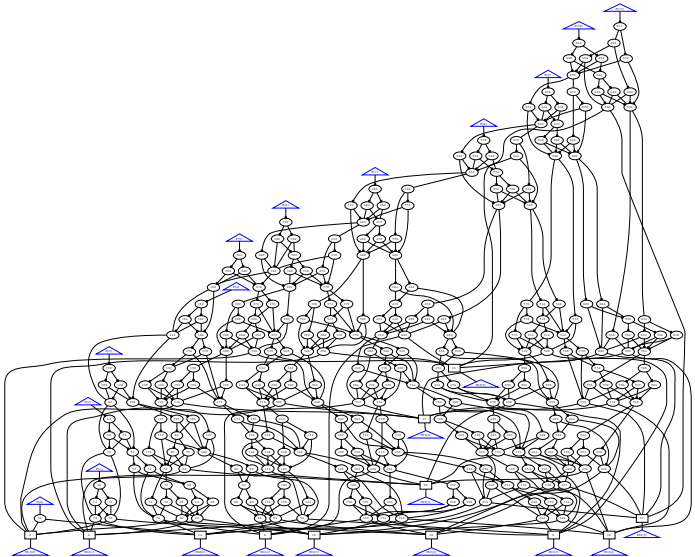
LPAC

```
1   a -b+1-a;
2   a -c+a*b;
3   % 1*(a)+2, -c;
1   d;
2   d;
4   % 4*(-1), c;
```

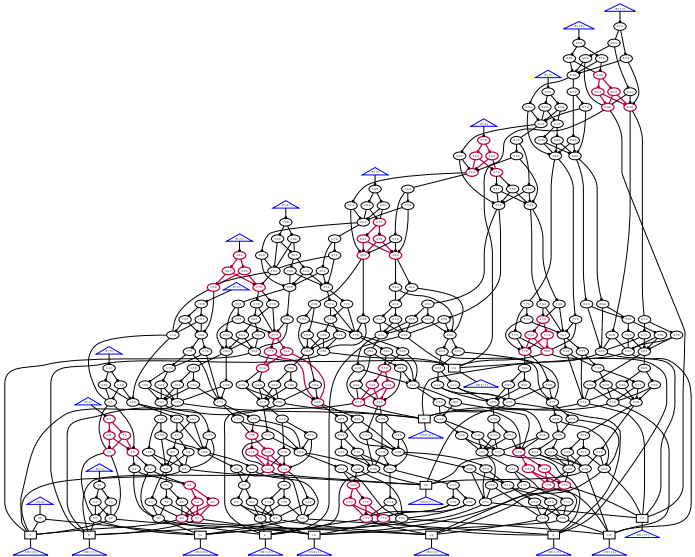
LPAC simulates NSS

```
1   a -b+1-a;
2   a -c+a*b;
3   % 1*(-a)+2*(-1), -c;
```

Repeated Calculations for Building Blocks



Repeated Calculations for Building Blocks



Repeated Calculations for Building Blocks

```
338 % 6 *(132-1) + 7, -136-132*130*128+132*130-132+130*128-130+1;
339 % 5 *(-130*128+130-1) + 338, -136+2*130*128-130-128+1;
338 d;
340 % 12 *(144-1) + 13, -148-144*142*140+144*142-144+142*140-142+1;
341 % 11 *(-142*140+142-1) + 340, -148+2*142*140-142-140+1;
340 d;
348 % 34 *(188-1) + 35, -192-188*186*184+188*186-188+186*184-186+1;
349 % 33 *(-186*184+186-1) + 348, -192+2*186*184-186-184+1;
348 d;
352 % 41 *(1102-1) + 42, -1106-1102*1100*182+1102*182-1102+1100*182-182+1;
353 % 40 *(-1100*182+182-1) + 352, -1106+2*1100*182-1100-182+1;
352 d;
360 % 68 *(1156-1) + 69, -1160-1156*1154*1152+1156*1154-1156+1154*1152-1154+1;
361 % 67 *(-1154*1152+1154-1) + 360, -1160+2*1154*1152-1154-1152+1;
360 d;
364 % 75 *(1170-1) + 76, -1174-1170*1168*1150+1170*1150-1170+1168*1150-1150+1;
365 % 74 *(-1168*1150+1150-1) + 364, -1174+2*1168*1150-1168-1150+1;
364 d;
366 % 78 *(1176-1) + 79, -1180-1176*1174*1132+1176*1132-1176+1174*1132-1132+1;
367 % 77 *(-1174*1132+1132-1) + 366, -1180+2*1174*1132-1174-1132+1;
366 d;
```

Idea: Use Patterns to Recycle Proof Steps

Idea: Use Patterns to Recycle Proof Steps

SC^2 - Approach:

Idea: Use Patterns to Recycle Proof Steps

SC^2 - Approach:

- **S**elect subcircuits in the larger circuit

Idea: Use Patterns to Recycle Proof Steps

SC^2 - Approach:

- **S**elect subcircuits in the larger circuit
- **C**apture their corresponding proofs as reusable patterns

Idea: Use Patterns to Recycle Proof Steps

SC^2 - Approach:

- **S**elect subcircuits in the larger circuit
- **C**apture their corresponding proofs as reusable patterns
- **S**earch for isomorphic subcircuits

Idea: Use Patterns to Recycle Proof Steps

SC^2 - Approach:

- **S**elect subcircuits in the larger circuit
- **C**apture their corresponding proofs as reusable patterns
- **S**earch for isomorphic subcircuits
- **C**onnect them to existing proof patterns via instantiation

New Rules

PATTERNNEW (i, I, S, O)

$$\frac{(X,P,C) \quad C(i)=\bot \quad \text{Correct-LPAC-Proof}(I,S) \quad O\subseteq\text{Conclusions}(S) \quad V_{\text{ext}}=\text{ExtensionVar}(O)}{(X,P,C\cup(i,I,O,V_{\text{ext}}))}$$

New Rules

PATTERNNEW (i, I, S, O)

$$\frac{(X,P,C) \quad C(i){=}\bot \quad \text{Correct-LPAC-Proof}(I,S) \quad O{\subseteq}\text{Conclusions}(S) \quad V_{\text{ext}}{=}\text{ExtensionVar}(O)}{(X,P,C{\cup}(i,I,O,V_{\text{ext}}))}$$

PATTERNAPPLY $(i, X_{\text{ext}}, \varphi, (j_1, \ldots, j_m), \{(k_1, p_1), \ldots, (k_n, p_n)\})$

$$\frac{\begin{array}{c}(X, P, C) \quad C(i) \neq \bot \quad X_{\text{ext}} \not\subseteq X \quad \forall v \in C(i).V_{\text{ext}} : \varphi(v) \in X_{\text{ext}} \\ \forall v \in \text{Var}(C(i)) : \varphi(v)^2 - \varphi(v) \in \langle B(X \cup X_{\text{ext}})\rangle \quad P(j_1) \neq \bot, \ldots, P(j_m) \neq \bot \quad C(i).I =_\varphi \{P(j_1), \ldots, P(j_m)\} \\ P(k_1) = \bot, \ldots, P(k_n) = \bot \quad p_1, \ldots, p_n \in \mathbb{K}[X \cup X_{\text{ext}}] \quad C(i).O =_\varphi \{p_1, \ldots, p_n\}\end{array}}{(X{\cup}X_{\text{ext}},P(k_1{\mapsto}p_1,\ldots,k_n{\mapsto}p_n),C)}$$

Repeated Calculations for Building Blocks

From:

```
338 % 6 *(l32-1) + 7, -l36-l32*l30*l28+l32*l30-l32+l30*l28-l30+1;
339 % 5 *(-l30*l28+l30-1) + 338, -l36+2*l30*l28-l30-l28+1;
```

Repeated Calculations for Building Blocks

From:

```
338 % 6 *(l32-1) + 7, -l36-l32*l30*l28+l32*l30-l32+l30*l28-l30+1;
339 % 5 *(-l30*l28+l30-1) + 338, -l36+2*l30*l28-l30-l28+1;
```

To:

1 PatternNew(P_1, [(i_1, $-v_1 - v_3v_2 + v_2$), (i_2, $-v_4 - v_3v_2 + v_3$), (i_3, $-v_0 + v_4v_1 - v_4 - v_1 + 1$)],
[p_1 % $i_2 * (v_1 - 1) + i_3$, $-v_0 - v_1 * v_3 * v_2 + v_1 * v_3 - v_1 + v_3 * v_2 - v_3 + 1$;
p_2 % $i_1 * (-v_3 * v_2 + v_3 - 1) + p_1$, $-v_0 + 2 * v_2 * v_3 - v_2 - v_3 + 1$;],
{p_2})

2 PatternApply(P_1, {}, [$v_0 \mapsto l_{36}$, $v_1 \mapsto l_{32}$, $v_2 \mapsto l_{30}$, $v_3 \mapsto l_{28}$],
(5,6,7), {(339, $-l_{36} + 2l_{30}l_{28} - l_{30} - l_{28} + 1$)})

Repeated Calculations for Building Blocks

```
338 % 6 *(132-1) + 7, -136-132*130*128+132*130-132+130*128-130+1;
339 % 5 *(-130*128+130-1) + 338, -136+2*130*128-130-128+1;
338 d;
340 % 12 *(144-1) + 13, -148-144*142*140+144*142-144+142*140-142+1;
341 % 11 *(-142*140+142-1) + 340, -148+2*142*140-142-140+1;
340 d;
348 % 34 *(188-1) + 35, -192-188*186*184+188*186-188+186*184-186+1;
349 % 33 *(-186*184+186-1) + 348, -192+2*186*184-186-184+1;
348 d;
352 % 41 *(1102-1) + 42, -1106-1102*1100*182+1102*182-1102+1100*182-182+1;
353 % 40 *(-1100*182+182-1) + 352, -1106+2*1100*182-1100-182+1;
352 d;
360 % 68 *(1156-1) + 69, -1160-1156*1154*1152+1156*1154-1156+1154*1152-1154+1;
361 % 67 *(-1154*1152+1154-1) + 360, -1160+2*1154*1152-1154-1152+1;
360 d;
364 % 75 *(1170-1) + 76, -1174-1170*1168*1150+1170*1150-1170+1168*1150-1150+1;
365 % 74 *(-1168*1150+1150-1) + 364, -1174+2*1168*1150-1168-1150+1;
364 d;
366 % 78 *(1176-1) + 79, -1180-1176*1174*1132+1176*1132-1176+1174*1132-1132+1;
367 % 77 *(-1174*1132+1132-1) + 366, -1180+2*1174*1132-1174-1132+1;
366 d;
```

Repeated Calculations for Building Blocks

1 PatternNew(P_1, [(i_1, $-v_1 - v_3 v_2 + v_2$), (i_2, $-v_4 - v_3 v_2 + v_3$), (i_3, $-v_0 + v_4 v_1 - v_4 - v_1 + 1$)],
[p_1 % $i_2 * (v_1 - 1) + i_3$, $-v_0 - v_1 * v_3 * v_2 + v_1 * v_3 - v_1 + v_3 * v_2 - v_3 + 1$;
p_2 % $i_1 * (-v_3 * v_2 + v_3 - 1) + p_1$, $-v_0 + 2 * v_2 * v_3 - v_2 - v_3 + 1$;],
{p_2})

2 PatternApply(P_1, {}, [$v_0 \mapsto l_{36}$, $v_1 \mapsto l_{32}$, $v_2 \mapsto l_{30}$, $v_3 \mapsto l_{28}$],
(5,6,7), {(339, $-l_{36} + 2 l_{30} l_{28} - l_{30} - l_{28} + 1$)})

3 PatternApply(P_1, {}, [$v_0 \mapsto l_{48}$, $v_1 \mapsto l_{44}$, $v_2 \mapsto l_{42}$, $v_3 \mapsto l_{40}$],
(11,12,13), {(341, $-l_{48} + 2 l_{42} l_{40} - l_{42} - l_{40} + 1$)})

4 PatternApply(P_1, {}, [$v_0 \mapsto l_{92}$, $v_1 \mapsto l_{88}$, $v_2 \mapsto l_{86}$, $v_3 \mapsto l_{84}$],
(33,34,35), {(349, $-l_{92} + 2 l_{86} l_{84} - l_{86} - l_{84} + 1$)})

5 PatternApply(P_1, {}, [$v_0 \mapsto l_{106}$, $v_1 \mapsto l_{102}$, $v_2 \mapsto l_{100}$, $v_3 \mapsto l_{82}$],
(40,41,42), {(353, $-l_{106} + 2 l_{100} l_{82} - l_{100} - l_{82} + 1$)})

6 PatternApply(P_1, {}, [$v_0 \mapsto l_{160}$, $v_1 \mapsto l_{156}$, $v_2 \mapsto l_{154}$, $v_3 \mapsto l_{152}$],
(67,68,69), {(361, $-l_{160} + 2 l_{154} l_{152} - l_{154} - l_{152} + 1$)})

7 PatternApply(P_1, {}, [$v_0 \mapsto l_{174}$, $v_1 \mapsto l_{170}$, $v_2 \mapsto l_{168}$, $v_3 \mapsto l_{150}$],
(74,75,76), {(365, $-l_{174} + 2 l_{168} l_{150} - l_{168} - l_{150} + 1$)})

8 PatternApply(P_1, {}, [$v_0 \mapsto l_{180}$, $v_1 \mapsto l_{176}$, $v_2 \mapsto l_{174}$, $v_3 \mapsto l_{132}$],
(77,78,79), {(367, $-l_{180} + 2 l_{174} l_{132} - l_{174} - l_{132} + 1$)})

Proof Checking

PATTERNNEW (i, I, S, O)

$$\frac{(X,P,C) \quad C(i)=\bot \quad \text{Correct-LPAC-Proof}(I,S) \quad O\subseteq\text{Conclusions}(S) \quad V_{\text{ext}}=\text{ExtensionVar}(O)}{(X,P,C\cup(i,I,O,V_{\text{ext}}))}$$

1. Do we already have a pattern with index i?
2. Do I and S form a correct standalone LPAC proof?
3. Are all polynomials in O conclusion polynomials of proof steps in I or S?

Proof Checking

PATTERNAPPLY $(i, X_{\text{ext}}, \varphi, (j_1, \ldots, j_m), \{(k_1, p_1), \ldots, (k_n, p_n)\})$

$$\frac{\begin{array}{c}(X, P, C) \qquad C(i) \neq \bot \qquad X_{\text{ext}} \not\subseteq X \qquad \forall v \in C(i).V_{\text{ext}} : \varphi(v) \in X_{\text{ext}} \\ \forall v \in \text{Var}(C(i)) : \varphi(v)^2 - \varphi(v) \in \langle B(X \cup X_{\text{ext}}) \rangle \qquad P(j_1) \neq \bot, \ldots, P(j_m) \neq \bot \qquad C(i).I =_\varphi \{P(j_1), \ldots, P(j_m)\} \\ P(k_1) = \bot, \ldots, P(k_n) = \bot \qquad p_1, \ldots, p_n \in \mathbb{K}[X \cup X_{\text{ext}}] \qquad C(i).O =_\varphi \{p_1, \ldots, p_n\}\end{array}}{(X \cup X_{\text{ext}}, P(k_1 \mapsto p_1, \ldots, k_n \mapsto p_n), C)}$$

1. Does pattern $C(i)$ exist?
2. Are all variables in X_{ext} fresh variables that are not contained in X?
3. Does φ map the extension variables of the pattern to the fresh variables in X_{ext}?
4. Does the mapping φ comply with the Boolean axioms $B(X \cup X_{\text{ext}})$?
5. Does I correspond to the input polynomials of $C(i)$ after the mapping φ is applied?
6. Does O correspond to the output polynomials of $C(i)$ after φ is applied?

Example

We algebraically encode Boolean resolution: from $\neg x \vee \neg y$ and $y \vee z$, we derive $\neg x \vee z$.

Let $G = \{xy, yz - y - z + 1\}$. The goal is to derive $x\bar{z} \in \langle G \cup \{\bar{z} - (1 - z)\}\rangle$, with $\bar{z}$ being a new variable representing $\bar{z} = 1 - z$.

```
1 xy
2 yz − y − z + 1
3 PatternNew(1, [(p1 v1v2), (p2 v2v3 − v2 − v3 + 1)],
            [Ext(p3, (w3), (1 − v3)),
             LinComb(p4, (p1, p2, p3), (w3, v1, v1v2 − v1), v1w3)], {p3, p4})
4 PatternApply(1, {z̄}, [v1 ↦ x, v2 ↦ y, v3 ↦ z, w3 ↦ z̄], (1, 2),
              {(l3, −z̄ + 1 − z), (l4, xz̄)})
```

PACHECK2

- `https://github.com/d-kfmnn/pacheck2`
- implemented in C++

- PACHECK2 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>`.

Evaluation: Circuit Verification

Name	Axioms (10^3)	LPAC without Patterns				LPAC with Patterns						
		Steps (10^3)	File (MB)	Mem (MB)	Time (s)	Steps (10^3)	#	Apply	max $\|S\|$	File (MB)	Mem (MB)	Time (s)
abc	64	196	454	982	10.17	48	9	4032	46	439	913	9.61
abc-rsn	64	196	454	982	10.35	48	12	4033	65	439	913	9.94
abc-cmp	64	196	454	982	10.45	48	12	4033	46	439	913	9.90
sp-ar-rc	96	270	676	1377	15.86	107	18	6509	83	663	1315	14.87
sp-bd-rc	98	284	1130	2098	32.33	111	42	6520	87	1117	2038	29.45
sp-dt-rc	96	292	1789	3112	56.63	108	58	7082	84	1776	3056	55.75
sp-os-rc	99	289	1314	2380	38.62	112	52	6542	87	1300	2321	38.48
sp-ar-cl	108	2043	959	1867	25.98	1820	79	3999	421	924	1749	24.37

Conclusion and Outlook

Conclusion:

- Extend LPAC with patterns to recycle proof steps
- Patterns can be used to avoid repeated calculations
- Reduction of proof steps of more than 50% in some cases

Future Work:

- Improve proof checking performance
- Investigate techniques developed for finding shorter proofs