SATISFIABLE ALGEBRAIC CIRCUIT VERIFICATION

Daniela Kaufmann
Johannes Kepler University, Linz, Austria

Dagstuhl Seminar
New Perspectives in Symbolic Computation and Satisfiability Checking
Dagstuhl, Germany & online

February 16, 2022

✉ daniela.kaufmann@jku.at

Bugs in hardware are expensive!

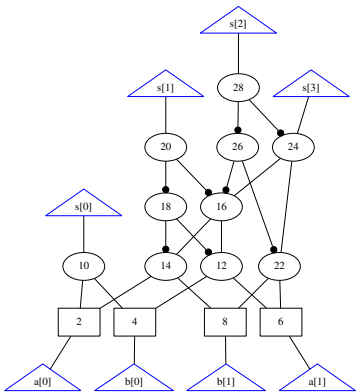
Circuit verification prevents issues like the famous Pentium FDIV bug.

Multiplier verification

Given: Gate-level integer multiplier for fixed bit-width.
Input format: AND-Inverter Graph

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

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



Formal Verification Techniques

Satisfiability Checking (SAT)

- SAT 2016 Competition
- Exponential run-time of solvers

Decision Diagrams

- First technique to detect Pentium bug
- Rely on manual decomposition

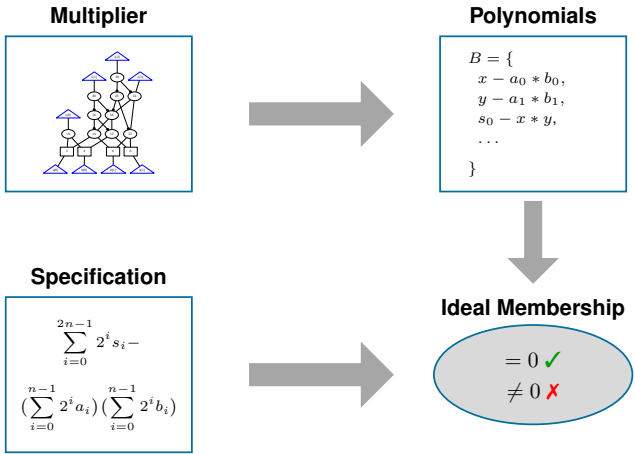
Theorem Proving

- Used in industry
- Requires manual effort
- Automated techniques rely on hierarchical information.

Algebraic Approach

- Great progress since 2015
- Polynomial encoding
- Works for non-trivial multiplier designs

Basic Idea of Algebraic Approach



From Circuits to Polynomials

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

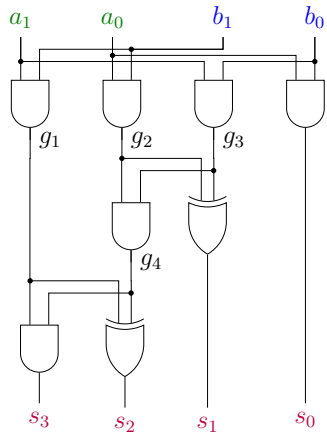
$$
\begin{array}{ll}
s_3 = g_1 \wedge g_4 & -s_3 + g_4 g_1, \\
s_2 = g_1 \oplus g_4 & -s_2 - 2g_4 g_1 + g_4 + g_1, \\
g_4 = g_2 \wedge g_3 & -g_4 + g_2 g_3, \\
s_1 = g_2 \oplus g_3 & -s_1 - 2g_2 g_3 + g_2 + g_3, \\
g_1 = a_1 \wedge b_1 & -g_1 + a_1 b_1, \\
g_2 = a_0 \wedge b_1 & -g_2 + a_0 b_1, \\
g_3 = a_1 \wedge b_0 & -g_3 + a_1 b_0, \\
s_0 = a_0 \wedge b_0 & -s_0 + a_0 b_0
\end{array}
$$

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

$$
\begin{array}{ll}
a_1, a_0 \in \mathbb{B} & a_1(1-a_1),\ a_0(1-a_0), \\
b_1, b_0 \in \mathbb{B} & b_1(1-b_1),\ b_0(1-b_0)
\end{array}
$$

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

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



Verification Technique

Verification Algorithm

Reduce specification $\sum_{i=0}^{2n-1} 2^i s_i - (\sum_{i=0}^{n-1} 2^i a_i)(\sum_{i=0}^{n-1} 2^i b_i)$ by elements of $G(C) \cup B(C)$

based on a fixed variable order until no further reduction is possible.

Then C is a multiplier iff the final remainder is zero.

Easy: Multipliers containing a ripple-carry adder

Hard: Multipliers containing a generate-and-propagate adder, e.g., carry-lookahead adder

Verification

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

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

Verification

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

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

$$8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0$$

Verification

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0
\end{aligned}
$$

Verification

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0
\end{aligned}
$$

Verification

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0
\end{aligned}
$$

Verification

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0
\end{aligned}
$$

Verification

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + s_0 - 2a_1 b_0 - a_0 b_0
\end{aligned}
$$

Verification

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + s_0 - 2a_1 b_0 - a_0 b_0 \\
s_0 - a_0 b_0
\end{aligned}
$$

Verification

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\
8g_1g_4 + 4s_2 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\
4g_2g_3 + 4g_1 + 2s_1 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1b_1 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1b_0 - 2a_0b_1 - a_0b_0 \\
2g_3 + s_0 - 2a_1b_0 - a_0b_0 \\
s_0 - a_0b_0 \\
0
\end{aligned}
$$

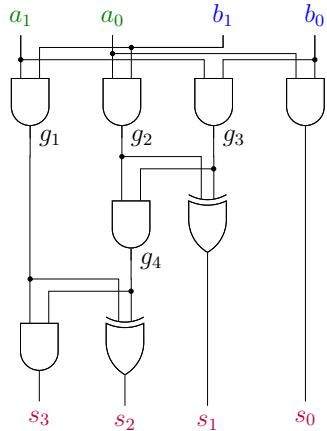
Faulty multiplier

Gate polynomials $G(C)$.

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

Boolean value constraints $B(C)$.

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



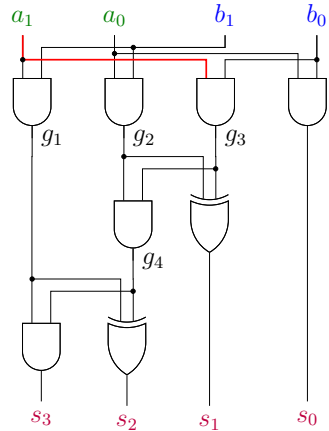
Faulty multiplier

Gate polynomials $G(C)$.

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

Boolean value constraints $B(C)$.

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



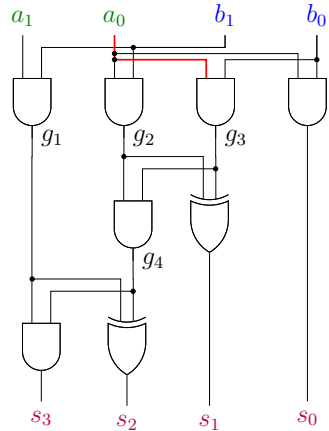
Faulty multiplier

Gate polynomials $G(C)$.

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

Boolean value constraints $B(C)$.

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



Verification of a faulty multiplier

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + s_0 - 2a_1 b_0 - a_0 b_0
\end{aligned}
$$

Verification of a faulty multiplier

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + s_0 - 2a_1 b_0 - a_0 b_0 \\
s_0 - 2a_1 b_0 + a_0 b_0
\end{aligned}
$$

Verification of a faulty multiplier

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + s_0 - 2a_1 b_0 - a_0 b_0 \\
s_0 - 2a_1 b_0 + a_0 b_0
\end{aligned}
$$

Verification of a faulty multiplier

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

$$
\begin{aligned}
8s_3 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
8g_1 g_4 + 4s_2 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_4 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_2 g_3 + 4g_1 + 2s_1 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
4g_1 + 2g_3 + 2g_2 + s_0 - 4a_1 b_1 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + 2g_2 + s_0 - 2a_1 b_0 - 2a_0 b_1 - a_0 b_0 \\
2g_3 + s_0 - 2a_1 b_0 - a_0 b_0 \\
s_0 - 2a_1 b_0 + a_0 b_0 \\
-2a_1 b_0 + 2a_0 b_0
\end{aligned}
$$

Counter Example

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

Counter Example

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

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

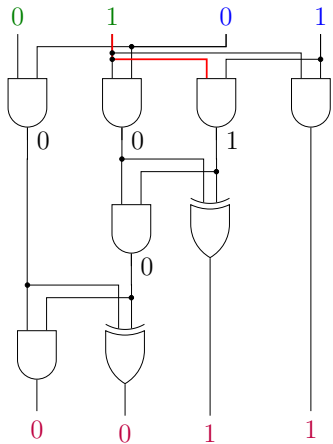
$01 \cdot 01 = 0001$

Counter Example

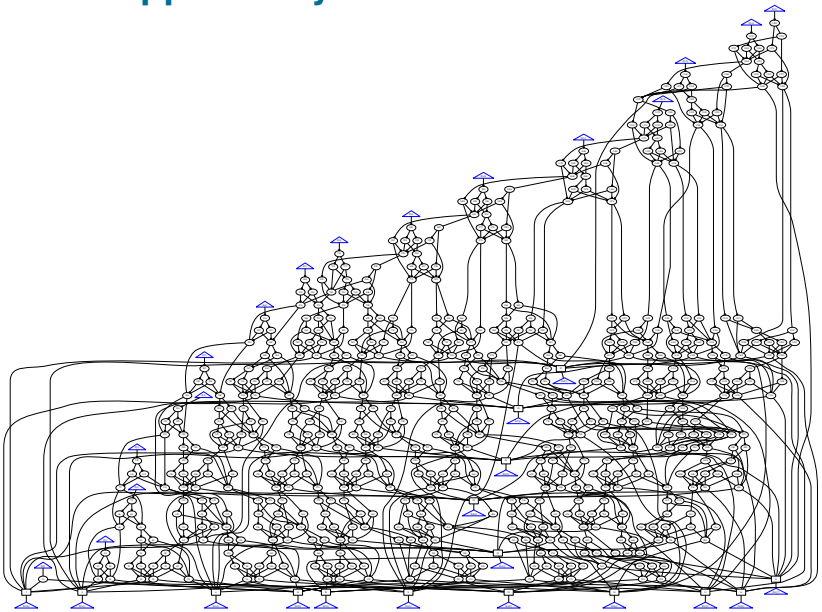
Remainder: $-2a_1b_0 + 2a_0b_0 \neq 0$

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

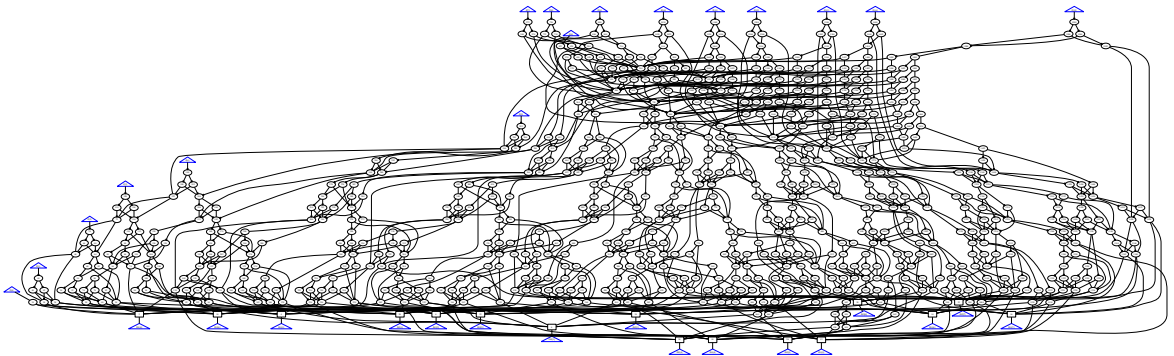
$01 \cdot 01 = 0001$



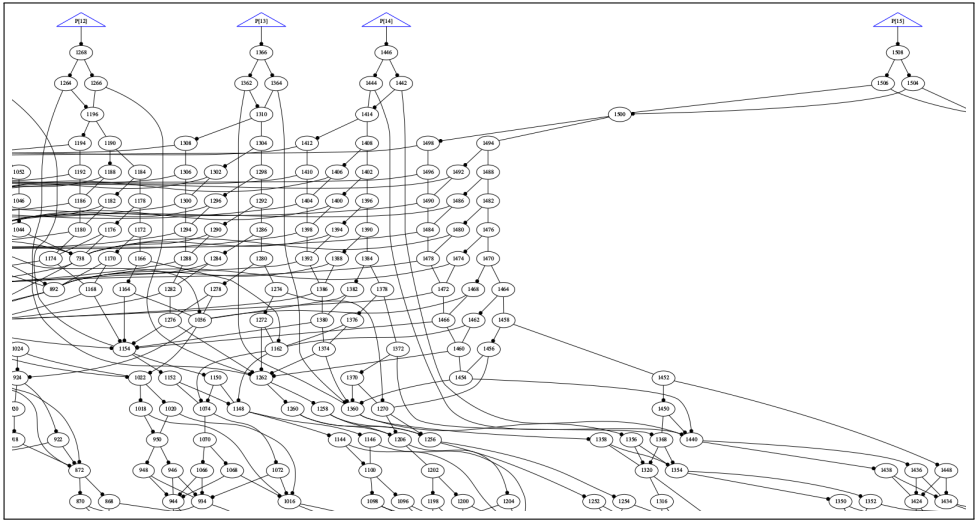
Multiplier – Ripple-Carry Adder



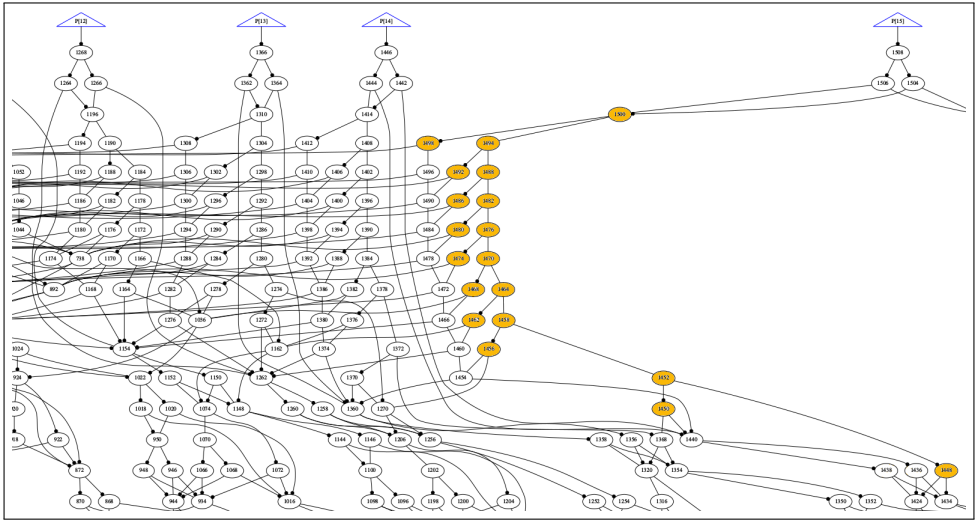
Multiplier – Carry-Lookahead Adder



Multiplier – Carry-Lookahead Adder

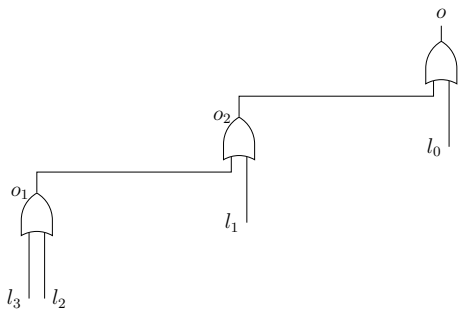


Multiplier – Carry-Lookahead Adder



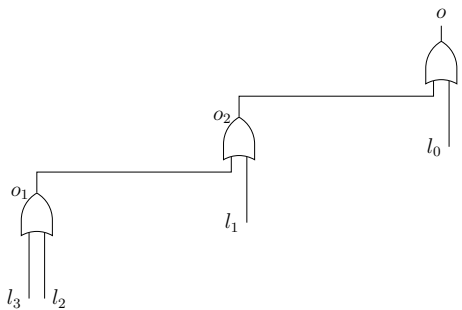
OR Gates

$$
\begin{aligned}
o = o_2 \vee x_0 \quad & -o + o_2 + l_0 - o_2 l_0, \\
o_2 = o_1 \vee l_1 \quad & -o_2 + o_1 + l_1 - o_1 l_1, \\
o_1 = l_3 \vee l_2 \quad & -o_1 + l_3 + l_2 - l_3 l_2
\end{aligned}
$$



OR Gates

$$
\begin{aligned}
o = o_2 \vee x_0 \quad & -o + o_2 + l_0 - o_2 l_0, \\
o_2 = o_1 \vee l_1 \quad & -o_2 + o_1 + l_1 - o_1 l_1, \\
o_1 = l_3 \vee l_2 \quad & -o_1 + l_3 + l_2 - l_3 l_2
\end{aligned}
$$



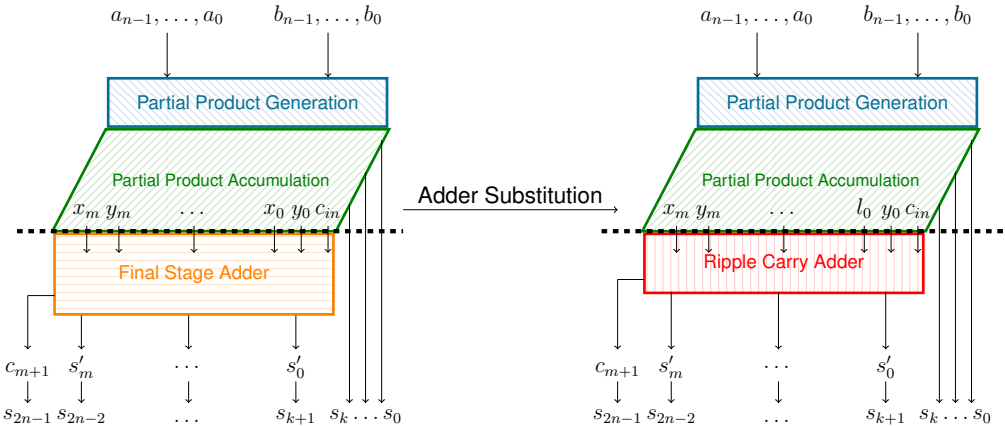
$$o = l_0 + l_1 - l_0 l_1 + l_2 - l_0 l_2 - l_1 l_2 + l_0 l_1 l_2 + l_3 - l_0 l_3 - l_1 l_3 + l_0 l_1 l_3 - l_2 l_3 + l_0 l_2 l_3 + l_1 l_2 l_3 - l_0 l_1 l_2 l_3$$

$15 = 2^4 - 1$ monomials

n OR Gates $\rightarrow 2^{n+1} - 1$ monomials

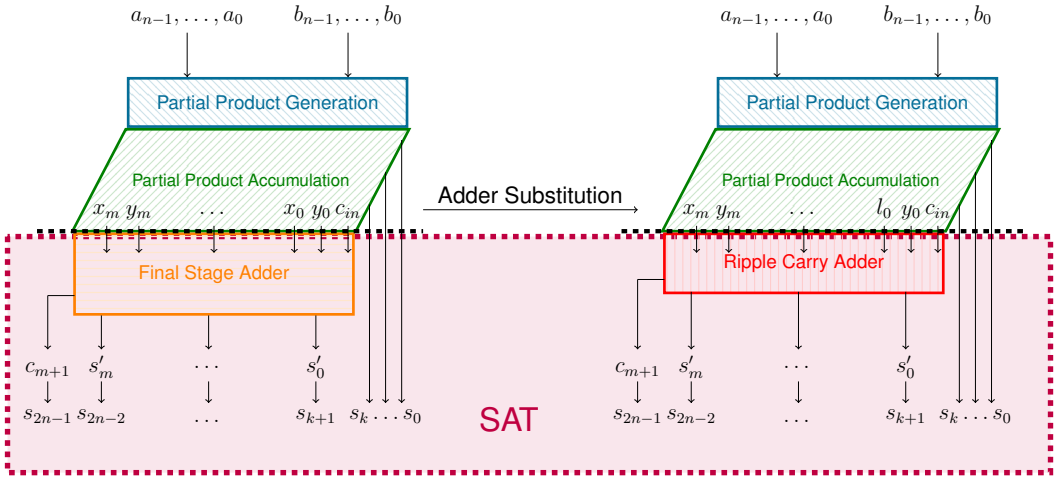
Previous Approach: SAT & Computer Algebra

[Kaufmann et al., 2019]



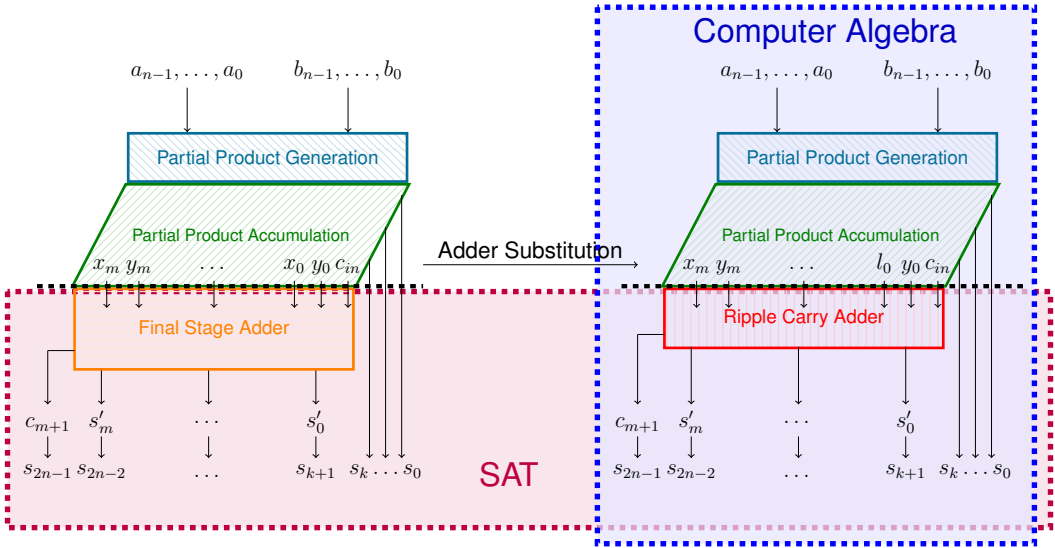
Previous Approach: SAT & Computer Algebra

[Kaufmann et al., 2019]

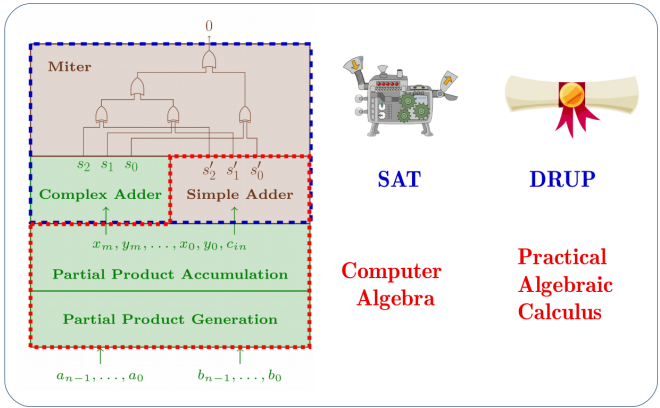


Previous Approach: SAT & Computer Algebra

[Kaufmann et al., 2019]



Problem: Proof Certificates



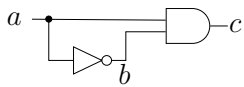
It is possible to simulate DRUP proofs in PAC, but it does not scale [Kaufmann et al., 2020a].

Challenges

How to combine DRUP and PAC into a single proof?

Practical Algebraic Calculus

[Ritirc et al., 2018]



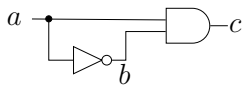

```
P =   -b+1-a,        b = ¬a
      -c+a*b,        c = a ∧ b = a ∧ ¬a
      -a^2+a         a = ⊥ ∨ a = ⊤
Spec = c             c = ⊥
```

```
* : -b+1-a,          a,          -a*b+a-a^2;
* : -a^2+a,          -1,         a^2-a;
+ : -a*b+a-a^2,      a^2-a,      -a*b;
+ : -a*b,            -c+a*b,     -c;
* : -c,              -1,         c;
```

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

Practical Algebraic Calculus

[Ritirc et al., 2018]



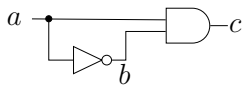

```
P =   -b+1-a,      b = ¬a
      -c+a*b,      c = a ∧ b = a ∧ ¬a
      -a^2+a       a = ⊥ ∨ a = ⊤
Spec = c           c = ⊥
```

```
* : -b+1-a,        a,          -a*b+a-a^2;
* : -a^2+a,        -1,         a^2-a;
+ : -a*b+a-a^2,    a^2-a,      -a*b;
+ : -a*b,          -c+a*b,     -c;
* : -c,            -1,         c;
```

Can be checked by our older proof checker PACTRIM.

1. Boolean Variables

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

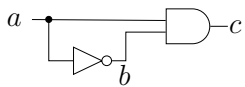



```
P =   -b+1-a,
      -c+a*b,
      -a^2+a
Spec = c
```

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

2. Indices

Introduce indices to reduce proof size.

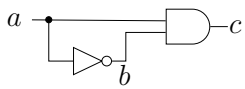



```
P =   1 -b+1-a;
      2 -c+a*b;
Spec = c
```

```
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.

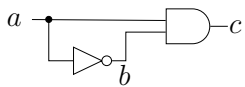



```
P =   1 -b+1-a;
      2 -c+a*b;
Spec = c
```

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

4. Linear Combination Rule

Introduce a linear combination rule to replace explicit addition and multiplication rules.




```
P =   1 -b+1-a;
      2 -c+a*b;
Spec = c
```

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

5. Extension Rule

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

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

5. Extension Rule

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

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

5. Extension Rule

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

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

```
P =   1 x*y;
      2 y*z-y-z+1;
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;
```

Ext(i, v, p) $(X, P) \Rightarrow (X \cup \{v\}, P(i \mapsto -v + p))$
provided that $P(i) = \bot$ and $v \notin X$ and $p \in \mathbb{Z}[X]/\langle B(X)\rangle$,
and $p^2 - p \equiv 0 \mod \langle B(X)\rangle$.

5. Extension Rule

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

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

```
P =   1 x*y;
      2 y*z-y-z+1;
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;
 6  *  1,    f, f*x*y;
 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 & PASTÈQUE

[Kaufmann et al., 2020b]

PACHECK C++ implementation

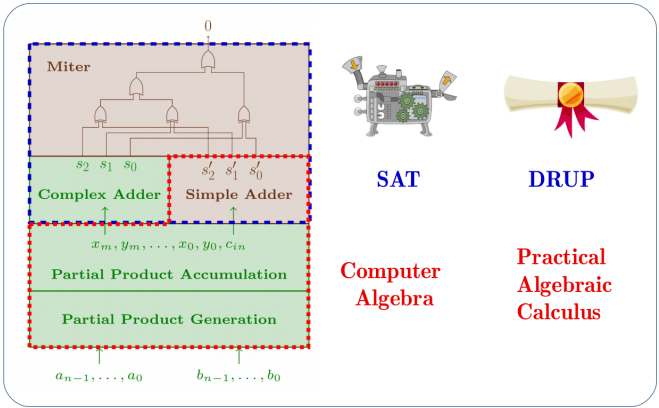
PASTÈQUE Isabelle/HOL

- relying on Isabelle's Refinement Framework
- abstract specification on ideals: specification in ideal
- final step: executable checker

Challenges

How to generalize **EXT** in a meaningful way?

Problem: Proof Certificates

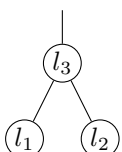


It is possible to simulate DRUP proofs in PAC, but it does not scale [Kaufmann et al., 2020a].

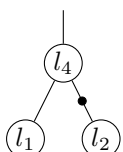
Dual Variables

[Kaufmann et al., 2022]

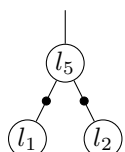
Provide a shorthand notation for inverters.



$$l_3 = l_1 \wedge l_2$$
$$-l_3 + l_1 l_2$$



$$l_4 = l_1 \wedge \neg l_2$$
$$-l_4 - l_1 l_2 + l_1$$



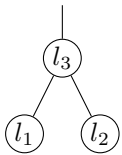
$$l_5 = \neg l_1 \wedge \neg l_2$$
$$-l_5 + l_1 l_2 - l_1 - l_2 + 1$$

Dual Variables

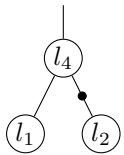
[Kaufmann et al., 2022]

Provide a shorthand notation for inverters.

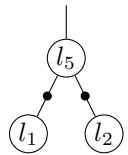
Dual variables.
Whenever two variables $l_i, f_i \in \{0, 1\}$ fulfill the relation $f_i = 1 - l_i$, we have $f_i = \mathrm{dual}(l_i)$.



$$l_3 = l_1 \wedge l_2$$
$$-l_3 + l_1 l_2$$
$$-l_3 + l_1 l_2$$



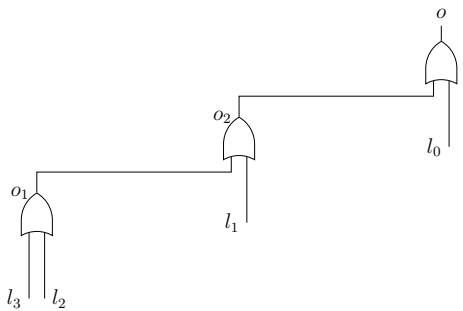
$$l_4 = l_1 \wedge \neg l_2$$
$$-l_4 - l_1 l_2 + l_1$$
$$-l_4 + l_1 f_2$$



$$l_5 = \neg l_1 \wedge \neg l_2$$
$$-l_5 + l_1 l_2 - l_1 - l_2 + 1$$
$$-l_5 + f_1 f_2$$

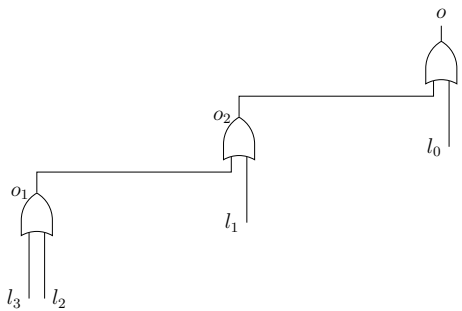
OR Gates

$$
\begin{aligned}
o = o_2 \vee x_0 \quad & -o + o_2 + l_0 - o_2 l_0, \\
o_2 = o_1 \vee l_1 \quad & -o_2 + o_1 + l_1 - o_1 l_1, \\
o_1 = l_3 \vee l_2 \quad & -o_1 + l_3 + l_2 - l_3 l_2
\end{aligned}
$$



OR Gates

$$
\begin{aligned}
o = o_2 \vee x_0 \quad & -o + o_2 + l_0 - o_2 l_0, \\
o_2 = o_1 \vee l_1 \quad & -o_2 + o_1 + l_1 - o_1 l_1, \\
o_1 = l_3 \vee l_2 \quad & -o_1 + l_3 + l_2 - l_3 l_2
\end{aligned}
$$



$$o = l_0 + l_1 - l_0 l_1 + l_2 - l_0 l_2 - l_1 l_2 + l_0 l_1 l_2 + l_3 - l_0 l_3 - l_1 l_3 + l_0 l_1 l_3 - l_2 l_3 + l_0 l_2 l_3 + l_1 l_2 l_3 - l_0 l_1 l_2 l_3$$

$$o = 1 - f_0 f_1 f_2 f_3$$

Practical Difficulty

Key Method for polynomial inference: Gröbner basis algorithm

Relies on a reduction method based on a fixed variable order that will immediately eliminate one of each pair of dual variables by re-expressing it using its partner.

Practice: During verification we always reduce the specification by the dual constraint $-f_i - l_i + 1$ of a gate variable l_i before reducing by its gate constraint. This has the effect that all occurrences of f_i in the specification will be flipped to l_i before reducing l_i.

Problem: Compact representation is unfolded.

Practical Difficulty

Key Method for polynomial inference: Gröbner basis algorithm

Relies on a reduction method based on a fixed variable order that will immediately eliminate one of each pair of dual variables by re-expressing it using its partner.

Practice: During verification we always reduce the specification by the dual constraint $-f_i - l_i + 1$ of a gate variable l_i before reducing by its gate constraint. This has the effect that all occurrences of f_i in the specification will be flipped to l_i before reducing l_i.

Problem: Compact representation is unfolded.

$\rightarrow$ We need dedicated preprocessing techniques to keep compact representation.

Challenges

How to integrate dual variables into polynomial reduction?

Calculate with Dual Variables

Proposition 1.
For all Boolean variables l_i and their dual representation $\mathrm{dual}(l_i) = f_i$ we have $l_i f_i = 0$.

"l_i and $\mathrm{dual}(l_i)$ cannot be 1 at the same time."

Proposition 2.
For all Boolean variables l_i and their dual representation $\mathrm{dual}(l_i) = f_i$ we have $l_i + f_i = 1$.

"l_i and $\mathrm{dual}(l_i)$ add up to 1."

Dual Mergeable

We call two monomials m_1 and m_2 **dual mergeable** iff $m_1 = cf_i\tau$ and $m_2 = cl_i\tau$ for c a constant, τ a term, and some index i. We call the monomial $\mathrm{dmerge}(m_1, m_2) = c\tau$ their **dual merge**.

Algorithm: Merging monomials(p)

Input : Polynomial p
Output: Simplified polynomial r

$q \leftarrow$ sort-degree-lex(p); $r \leftarrow 0$;
while $q \neq 0$ **do**
  $q_l \leftarrow \mathrm{lm}(q)$; $t \leftarrow \mathrm{tail}(q)$; simplify $\leftarrow \bot$;
  while $t \neq 0$ and $\deg(q_l) = \deg(\mathrm{lt}(t))$ and $\neg$ simplify **do**
    $q_t \leftarrow \mathrm{lt}(t)$;
    if q_l and q_t are dual mergeable **then**
      $q \leftarrow q - q_l - q_t + \mathrm{dmerge}(q_l, q_t)$;
      simplify $\leftarrow \top$;
    else $t \leftarrow t - q_t$;
  if $\neg$ simplify **then** $r \leftarrow r + q_l$, $q \leftarrow q - q_l$;
return sort-lex(r);

Dual Mergeable

Example

Let $p = l_1 f_2 f_3 + l_1 f_2 l_3 + l_1 l_2 f_3 + f_1 f_2 + l_2 \in \mathbb{Z}[l_1, l_2, l_3, f_1, f_2, f_3]$. We write q_i to denote the polynomial q after iteration i and indicate the dual merges.

$$
\begin{aligned}
q_0 &= l_1 f_2 f_3 + l_1 f_2 l_3 + l_1 l_2 f_3 + f_1 f_2 + l_2 && r = 0 \\
q_1 &= l_1 l_2 f_3 + f_1 f_2 + \boxed{l_1 f_2} + l_2 && r = 0 \\
q_2 &= f_1 f_2 + l_1 f_2 + l_2 && r = l_1 l_2 f_3 \\
q_3 &= \boxed{f_2} + l_2 && r = l_1 l_2 f_3 \\
q_4 &= \boxed{1} && r = l_1 l_2 f_3 \\
q_5 &= 0 && r = l_1 l_2 f_3 + 1
\end{aligned}
$$

Challenges

How to derive the most efficient representation of polynomials using dual variables?

How to apply other SAT concepts on polynomials?

Carry Rewriting

Goal: Rewrite encoding of carry look-ahead unit into a ripple-carry unit, which can easily be verified using computer algebra.

Algorithm: Carry-Rewriting

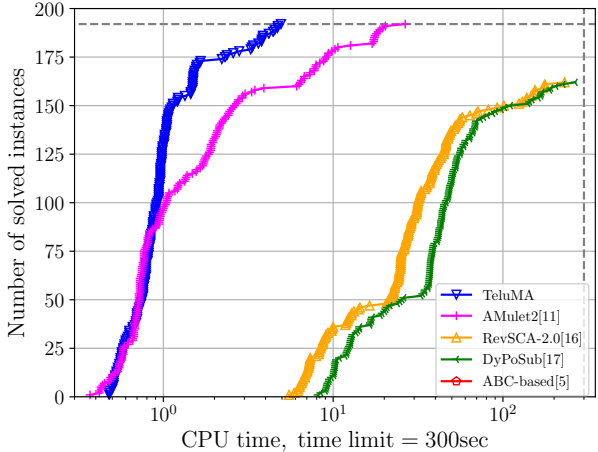
Input : Circuit C in AIG format

Output: Carry-rewritten Gröbner basis of C

1 $F \leftarrow$ Mark-final-stage-adder(C);
2 $G \leftarrow$ Dual-Polynomial-Encoding(F);
3 $H \leftarrow$ Polynomial-Encoding($C \setminus F$);
4 $G \leftarrow$ Eliminate-Pure-Positive-Variables(G);
5 $G \leftarrow$ Tail-Substitution(G);
6 $G \leftarrow$ Carry-Unfolding(G);
7 **return** $G \cup H$

Evaluation - Multiplier Verification

Verification of 192 unsigned 64-bit multipliers



Evaluation - Proof Certificates

architecture	n	[Kaufmann et al., 2019]			[Kaufmann et al., 2020a]		[Kaufmann et al., 2022]	
		DRUP	PAC	total (s)	PAC	total (s)	PAC	total (s)
		# rules	# rules		# rules		# rules	
sp-ar-cl	32	14 927	33 834	1	1 597 897	164	60 336	0
sp-bd-ks	32	17 528	34 958	1	817 956	28	54 116	0
sp-dt-lf	32	3 138	33 451	1	321 720	5	47 835	0
bp-ct-bk	32	2 276	27 312	1	217 128	3	36 356	0
bp-wt-cl	32	46 502	30 561	2	5 536 176	3 375	114 665	2
sp-ar-cl	64	65 317	139 338	8	-	TO	289 632	4
sp-bd-ks	64	44 921	142 138	6	1 440 943	74	214 378	3
sp-dt-lf	64	28 772	138 539	6	816 572	19	192 805	2
bp-ct-bk	64	19 891	105 579	5	459 262	15	136 703	2
bp-wt-cl	64	42 199	118 573	19	-	TO	774 044	24

All benchmarks are generated by the Arithmetic Model Generator [Homma et al., 2006]. TO = 3600 sec

Challenges

Proofs

- How to generalize **EXT** in a meaningful way?
- How to combine DRUP and PAC into a single proof?

Algebraic reasoning

- How to integrate dual variables into polynomial reduction?
- How to derive the most efficient representation of polynomials using dual variables?
- How to apply other SAT concepts on polynomials?

SATISFIABLE ALGEBRAIC CIRCUIT VERIFICATION

Daniela Kaufmann
Johannes Kepler University, Linz, Austria

Dagstuhl Seminar
New Perspectives in Symbolic Computation and Satisfiability Checking
Dagstuhl, Germany & online

February 16, 2022

✉ daniela.kaufmann@jku.at

References I

[Homma et al., 2006] Homma, N., Watanabe, Y., Aoki, T., and Higuchi, T. (2006).
Formal Design of Arithmetic Circuits Based on Arithmetic Description Language.
IEICE Transactions, 89-A(12):3500–3509.

[Kaufmann et al., 2022] Kaufmann, D., Beame, P., Biere, A., and Nordström, J. (2022).
Adding dual variables to algebraic reasoning for gate-level multiplier verification.
In *DATE*. IEEE.

[Kaufmann et al., 2019] Kaufmann, D., Biere, A., and Kauers, M. (2019).
Verifying Large Multipliers by Combining SAT and Computer Algebra.
In *FMCAD 2019*, pages 28–36. IEEE.

[Kaufmann et al., 2020a] Kaufmann, D., Biere, A., and Kauers, M. (2020a).
From DRUP to PAC and back.
In *DATE 2020*, pages 654–657. IEEE.

References II

[Kaufmann et al., 2020b] Kaufmann, D., Fleury, M., and Biere, A. (2020b).
Pacheck and Pastèque, Checking Practical Algebraic Calculus Proofs.
In *FMCAD 2020*, volume 1 of *FMCAD*, pages 264–269. TU Vienna Academic Press.

[Ritirc et al., 2018] Ritirc, D., Biere, A., and Kauers, M. (2018).
A Practical Polynomial Calculus for Arithmetic Circuit Verification.
In *SC2 2018*, pages 61–76. CEUR-WS.